Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Enhancing Visual SLAM for Autonomous Forest Navigation through Robust Feature Correspondence

Author:
Saifullah Ijaz

Supervisors:
Dr Bahadir Kocer
Dr Ronald Clark

Submitted in partial fulfilment of the requirements for the MEng degree in Electronic and Information Engineering of Imperial College London

June 2024

Abstract

Robotic systems are increasingly being used to reduce human workload for forestry tasks but autonomous forest navigation remains a significant challenge. Existing Visual Simultaneous Localisation and Mapping (V-SLAM) systems perform well in structured environments but falter in dynamic, unstructured forest settings due to their reliance on hand-crafted feature detection and correspondence methods.

This report focuses on analysing the effectiveness of incorporating the state-of-the-art deep learning based SuperPoint feature detector and SuperGlue feature matcher into a V-SLAM system for forest environments. We present Forest SLAM, a V-SLAM framework which leverages the pre-trained SuperPoint and SuperGlue models to achieve highly accurate relative pose estimations in challenging forest scenes. Our experimental results show that Forest SLAM is a robust SLAM framework for forest environments, with our visual-only stereo SLAM system achieving a Relative Pose Error of 4.718% on the BotanicGarden dataset leaderboard, outperforming the state-of-the-art visual-only ORB-SLAM3 stereo method.

Moreover, the stereo version of Forest SLAM generates 3D point cloud SLAM maps that are comparable with the ground truth. An ablation study further highlights the superiority of SuperPoint and SuperGlue over traditional ORB features and Brute Force matching, particularly under conditions of motion blur, demonstrating their robustness in challenging forest scenes.

Acknowledgments

I would like to thank my supervisors Dr Bahadir Kocer and Dr Ronald Clark for their invaluable support and insight throughout this project. Their guidance provided me with clear direction and their deep understanding of the subject matter helped me refine and validate my ideas.

I would also like to thank Thomas Pritchard, Marco Chan and Tommy Woodley for insightful discussions related to the project. Their advice encouraged me to find new solutions to the challenges I encountered in the practical implementation of my project.

Contents

Chapter 1

Introduction

1.1 Motivation

Forest environments form a large part of the natural terrain throughout many areas of the world. Ecological studies in forests are essential to monitor the changing biodiversity within these ecosystems. Typical forestry applications include forest monitoring, wildlife monitoring, early detection of wildfires and search and rescue operations [1]. Due to the large geographical areas covered by forests, it is often difficult for humans to perform monitoring tasks manually. There are also potential safety concerns for search and rescue operations that take place after natural disasters, where it is often unsafe for emergency response teams to carry out searches in areas prone to earthquakes and wildfires [2]. Autonomous robots have been investigated as a means to minimise direct human involvement for forestry related applications. However, dense canopy cover renders GPS ineffective [3], which poses a significant challenge for autonomous forest navigation, leading to SLAM [4] being explored as an alternative approach for robot navigation.

Visual SLAM (V-SLAM) systems employ cameras as the main sensors, utilising visual data to carry out localisation and mapping tasks. As the robot navigates its surroundings, it uses re-observed features to create a map of the environment whilst simultaneously determining its position within the map [5]. State-of-the-art V-SLAM systems achieve good localisation accuracy in structured environments like the ones present in general datasets such as KITTI, EUROC and TUM RGB-D [6, 7, 8]. Nevertheless, they rely on hand-crafted feature detection and correspondence techniques which prove ineffective when faced with the dynamic scenes present in forest environments. There is limited research on domain specific SLAM for unstructured environments [9] yet there is clearly a need for a robust feature correspondence method to reliably inform the pose estimation task in a V-SLAM system for forest environments.

1.2 Contributions

SuperPoint [10] and SuperGlue [11] are state-of-the-art deep learning based feature detection and correspondence methods that have been specifically designed as V-SLAM frontends. They have outperformed traditional hand-crafted approaches as well as other learning-based systems, achieving state-of-the-art results for pose estimation tasks on outdoor urban scenes. The objective of this report is to evaluate the effectiveness of the SuperPoint feature detector and SuperGlue feature matcher for robust localisation in the context of a V-SLAM system aimed at forest environments. For our evaluations, we utilise the BotanicGarden dataset [12] which contains a variety of scenes captured in unstructured natural environments. To this end, we have made the following contributions:

- We propose Forest SLAM, a V-SLAM framework for forest environments which incorporates the SuperPoint feature detector and SuperGlue feature correspondence methods. Monocular and stereo versions of our system are provided as Robot Operating System (ROS) [13] packages for future deployment as part of an autonomous navigation system on a physical robot.
- We evaluate Forest SLAM on the BotanicGarden dataset, with our stereo system achieving a Relative Pose Error of 4.718%, outperforming the state-of-the-art ORB-SLAM3 [14] stereo V-SLAM system on the BotanicGarden leaderboard.
- We generate 3D point cloud SLAM maps for the stereo version of Forest SLAM, comparing the generated maps with the ground truth maps from the BotanicGarden dataset for both short and long sequences.
- We perform an ablation study, substituting SuperPoint with ORB features and SuperGlue with a Brute Force (BF) matcher into our stereo method. Motion blur is artificially added to the input images to create challenging scenes and the matched features are visually compared for each method. The SuperPoint and SuperGlue combination achieves a lower RPE than the ORB and BF approach for all three levels of motion blur tested, highlighting the superior robustness of learning-based feature detectors and matchers for V-SLAM systems aimed at forest environments.

Chapter 2

Background

In this chapter, we:

- Provide an overview of both SLAM and V-SLAM, highlighting the different tasks that constitute each of their respective pipelines.
- Explain the challenges with using vision-based sensors in forest environments.
- Describe traditional feature description techniques, discussing the limitations of each method.
- Review previous work in which different feature correspondence techniques were compared in the context of a visual odometry system for forest environments. In particular, we focus on the state-of-the-art SuperGlue feature correspondence method.
- Explore ORB-SLAM3 [14] and VINS-MONO [15], two current state-of-the-art V-SLAM systems.

2.1 SLAM

2.1.1 Overview of SLAM

SLAM [4] is a problem in which a mobile robot traverses through an unknown environment to progressively construct a map of the environment whilst also self-localising with respect to the map at the same time. SLAM has been applied to several different types of mobile robots, including ground, underwater and aerial vehicles. The SLAM problem involves the robot estimating the position of other landmarks within the environment as well as its own position relative to those landmarks [16].

Mapping an unknown environment requires a detailed understanding of the position of landmarks. This then informs the localisation task, in which a robot performs pose estimations to predict its position and orientation to ultimately find its location within the map [17].

2.1.2 General SLAM Pipeline

Figure 2.1: SLAM pipeline [18].

Figure 2.1 shows the sequence of tasks that take place within a SLAM system. Practical implementations of SLAM on mobile robots usually consist of a combination of exteroceptive (outward-looking) sensors which capture distance measurements to other objects within the environment, and proprioceptive (inward-looking) sensors which capture data such as velocity and orientation of the robot, which are used in pose estimations [19]. Typical exteroceptive sensors used on robotic platforms include LiDAR, RADAR, acoustic or vision-based sensors, which are used to characterise the location of objects within the environment. Examples of proprioceptive sensors include Inertial Measurement Units (IMUs), accelerometers and gyroscopes [20].

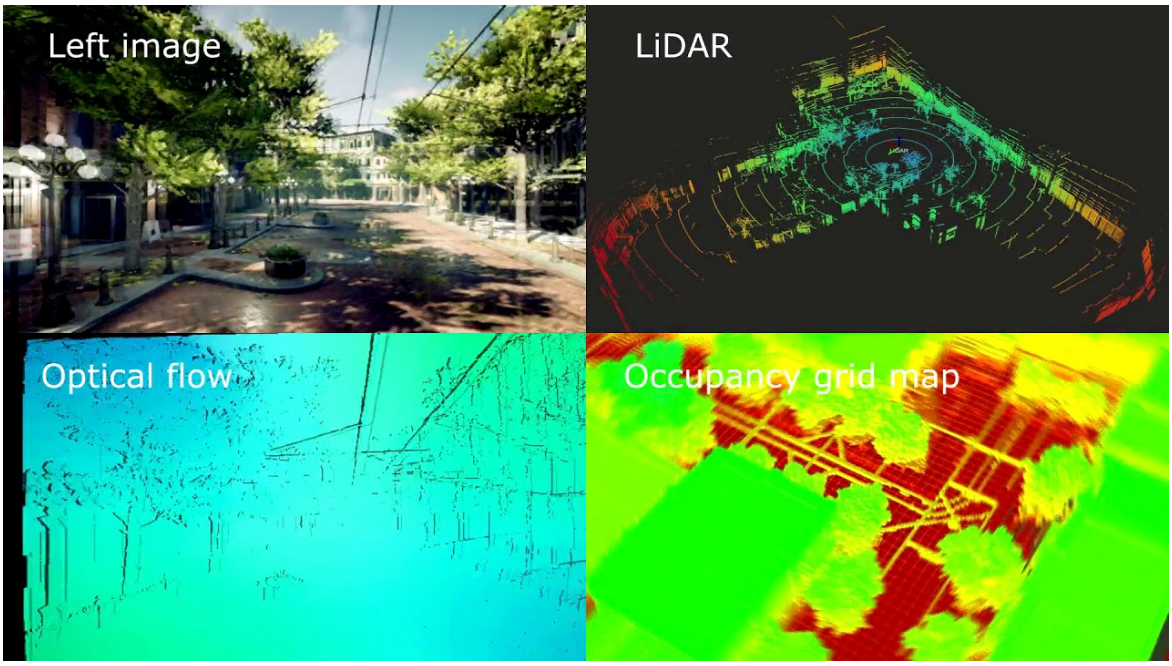

Figure 2.2: Examples of features used within SLAM [21].

Exteroceptive sensors are used to compute distances to features that are present in the environment. Features are discernible aspects of the environment that can be recognised from multiple viewpoints. Examples include wall segments and corners of objects for sensors such as LiDAR, or lines and points for vision-based sensors as shown in Figure 2.2. As the robot moves through the environment, new features will be observed, features may go out of view of the sensor, and other features that were previously seen might now be observed from a different point of view.

The terms data association, feature matching and feature correspondence are used interchangeably to represent the task of establishing if two features observed at different viewpoints represent the same object in the real-world. Correlating the newly observed information with the pre-established map information is crucial to acquiring an understanding of the spatial relationship between objects in the physical environment [22]. Hence, the quality of the maps generated by SLAM algorithms is directly influenced by how well the feature correspondence task is conducted.

The localisation task involves odometry [23], in which a robot makes predictions on where it is currently located on the map based on either proprioceptive or exteroceptive sensor measurements. To accurately determine its location, a robot needs to estimate its pose, which consists of its position and orientation. Similarly to the mapping aspect of SLAM, the localisation task involves taking observations of landmarks at various positions. As the robot travels through the unknown environment, measurements of features taken from different viewpoints in the environment will allow the robot to start to self-localise through place recognition as it remembers previously seen landmarks [24].

For small-scale SLAM systems, probabilistic methods [25] have been shown to be effective at reducing the uncertainty of the robot's predictions about its location on the map. However, odometry drift can occur due to the increased uncertainty of the robot's location at distances that can be considered far from the origin of the map. To counteract this, loop closure detection [26] is commonly employed in large-scale SLAM systems, in which re-observations of previously seen landmarks are used to correct the drift error from the sensor measurements. When a robot revisits a previously seen landmark, the uncertainty of its pose decreases due to prior knowledge about the locations of those landmarks on the map. The SLAM map is dynamically updated as additional sensor measurements are taken, with loops being closed within the map once the robot revisits a previously observed location.

2.2 Visual SLAM

2.2.1 Overview of Visual SLAM

V-SLAM involves the use of vision-based sensors such as monocular, stereo or RGB-D cameras [27]. Traditional SLAM approaches using sensors such as LiDAR or SONAR, either produce 2D maps or sparse 3D maps [28]. Visual data is especially useful for SLAM systems because it provides detailed information that is semantically useful for humans. This allows for the creation of dense 3D maps of physical environments that capture the visual experience of the human eye. Using visual data enables the implementation of visual odometry (VO) [29], where the camera frames are used to estimate the camera's pose. This, in turn, corresponds to the robot's pose, since the camera is attached to the robotic platform, thus allowing a robot to estimate its egomotion and localise itself in an unfamiliar environment using the visual input from the camera.

V-SLAM can employ either direct or indirect (feature-based) methods for both sparse and dense mapping. Direct methods use pixel intensities, for either the complete set of pixels in the frame or a sub-set of them. Whereas feature-based methods only consider pixels that correspond to features. Direct methods can result in higher density maps due to the increased amount of pixel information available. However, feature-based methods can improve both localisation and mapping accuracy since features are easily recognised and tracked between frames [30].

2.2.2 Visual SLAM Pipeline

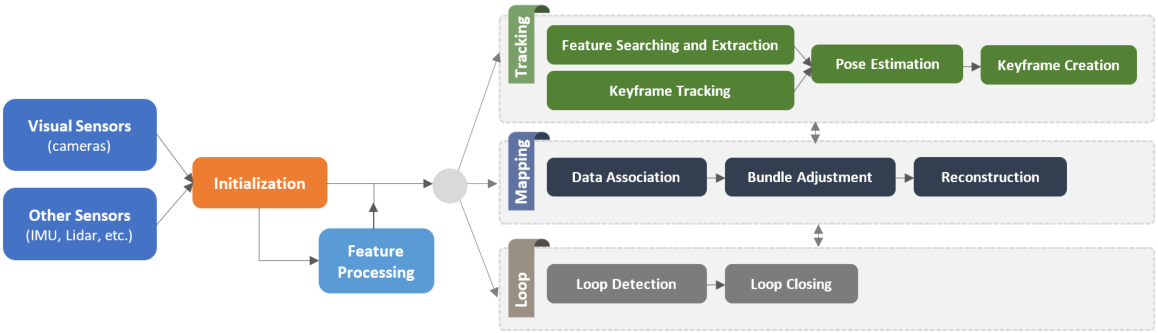

Figure 2.3: V-SLAM pipeline [31].

Figure 2.3 shows the sequence of tasks that take place within a V-SLAM system, although the exact tasks will vary depending on implementation choices, including whether direct or feature-based methods are used. The process will also be affected by the choice of proprioceptive sensors, since IMUs can be used alongside the vision-based sensor to implement visual inertial odometry (VIO) [32], in which the robot's egomotion is estimated using both the IMU and visual data. Once the initial data has been acquired from the sensors, three separate threads are used for tracking (localisation), mapping and loop closure respectively, which make up the V-SLAM system.

Tracking involves feature detection and the use of keyframes to estimate the camera pose. Traditionally, V-SLAM systems have employed filter-based methods like Kalman Filters (KF) or Extended Kalman Filters (EKF) [33, 34]. These methods establish a direct connection between localisation and mapping, assuming a direct correlation between camera pose and landmarks. However, this requires the estimations of the camera pose and locations of landmarks on the map to be updated for every camera frame which can be computationally intensive.

Modern V-SLAM systems use keyframe based approaches instead, which separate the localisation and mapping tasks. The camera pose estimates still take place over every frame for the localisation task, although this is restricted to a specific region of the map, as the robot can only occupy one location at any given time. The mapping task is carried out using only keyframes instead of every camera frame [35]. Keyframes are a sub-set of the frames captured by the camera which aim to maximise

the amount of useful visual information and minimise useless information, to reduce the amount of data required for mapping an unknown environment. Keyframes are usually frames in which there is a substantial change in visual data between one frame and the next, whereas frames that contain repeated visual information are usually ignored. Therefore, substantially less visual data is required to perform the mapping task in keyframe based V-SLAM approaches. This results in much more computationally efficient implementations that allow keyframe based V-SLAM approaches to outperform their filter-based counterparts [36].

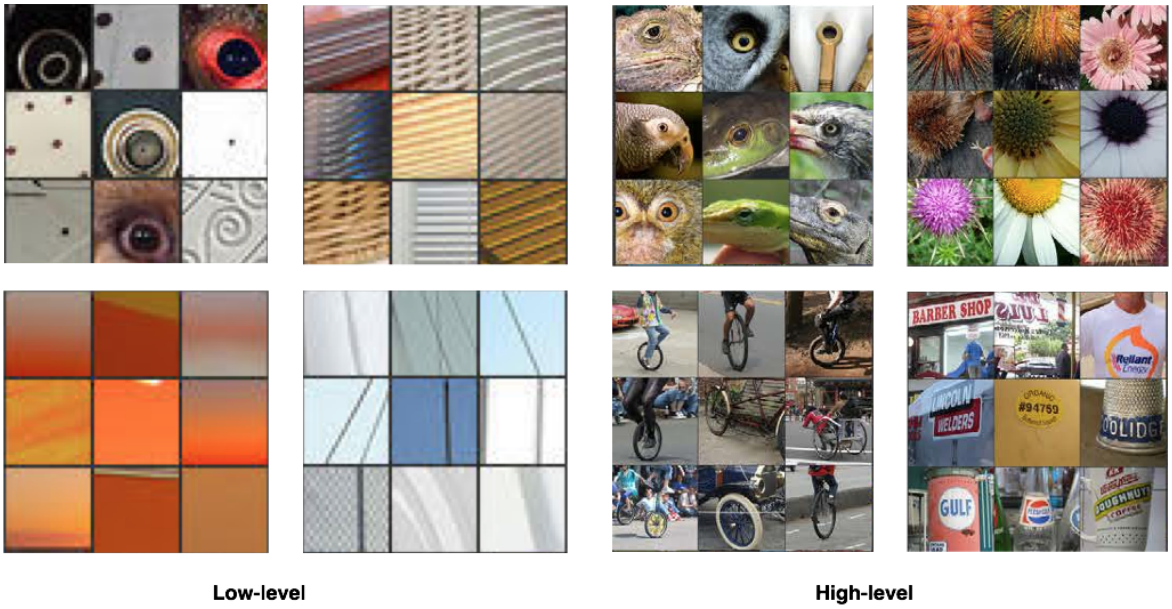

Figure 2.4: Low-level vs high-level features [37].

Mapping involves feature correspondence, bundle adjustment and reconstruction. Feature correspondence in V-SLAM requires matching low-level or high-level features between camera frames. Elements at a lower level such as points and lines concentrate on specific details including the fundamental geometric elements of objects within the scene and aspects such as textures. High-level features are semantically labelled objects that depict scenes in a manner that represents how humans perceive the environment [38], as shown in Figure 2.4.

The reconstruction task is use-case dependent and can be quite simple for basic 2D mapping. However, photorealistic 3D mapping requires reliable depth information to ensure that the landmarks are positioned correctly within 3D space. The depth data can be collected directly to generate 3D point clouds such as in RGB-D or LiDAR based SLAM approaches. If depth information is not directly available from exteroceptive sensors, more complex computer vision techniques [39] will need to be used to infer the depth information required to accurately reconstruct the 3D scene. Additionally, computer graphics techniques will need to be employed to ensure that textures are rendered correctly.

Bundle adjustment is an optimisation technique which is used to refine the visual reconstruction of the SLAM map. The estimates of the 3D feature coordinates and camera poses are optimised to minimise reprojection errors based on a cost function. However, the exact cost function used will be implementation dependent [40]. Loop closure is used to build accurate SLAM maps and to provide a mechanism that allows the robot to re-localise itself if it loses track of its current position. In V-SLAM systems, similarity metrics are used to compare the features in the current frame with those that have been observed in the past. This requires accurate image matching and is usually performed by a convolutional neural network (CNN) [41].

2.3 Traditional Feature Descriptors

2.3.1 Overview of Feature Descriptors

Feature description techniques aim to identify areas of an image that can be easily re-identified from different camera viewpoints. The first step involves extracting the points of interest from the image. The objective is to identify features that are robust to rotation, scale and changes of illumination in order to allow feature correspondence algorithms to accurately track features between frames. Feature description involves creating vectors or matrices to store information about the features such as the grey level of the pixels or the textures of the surrounding pixels. Descriptors can be local gradient-based, image intensity-based or learning-based [42].

2.3.2 Harris Corner Detector

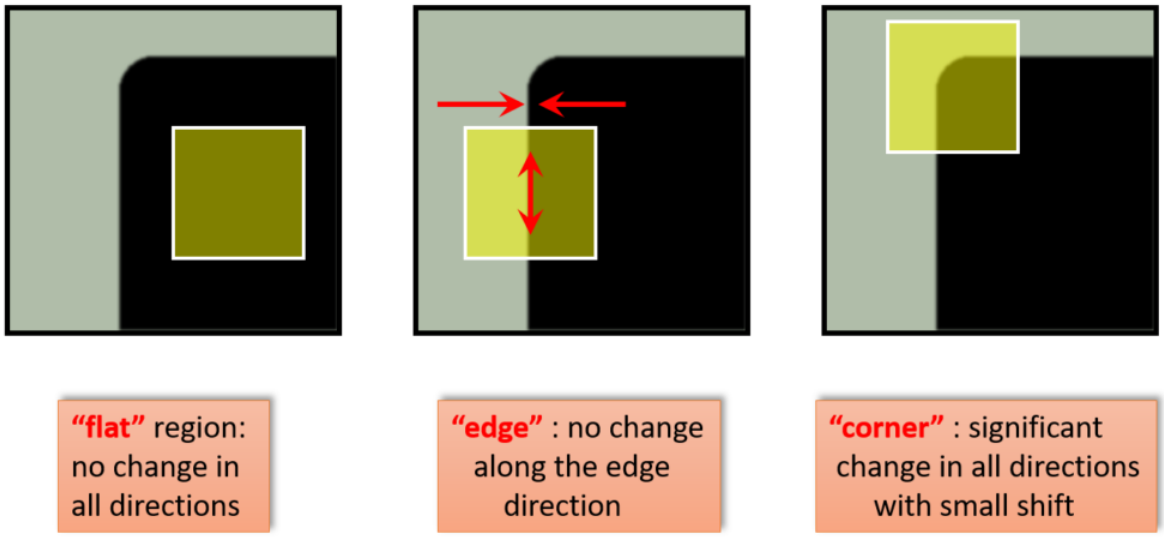

Figure 2.5: Harris corner detector [43].

Harris et al. proposed the Harris corner detector [44] shown in Figure 2.5 that identifies corners of objects as useful points, since they are invariant to rotation and can easily be tracked between frames.

$$E(u,v) = \sum_{(x,y)\in W} \underbrace{w(x,y)}_{\text{window function}} \Big[\underbrace{I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}\Big]^2 \tag{2.1}$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.2}$$

Equation 2.1 shows how the window is shifted across the pixels in the image in both horizontal and vertical directions which are given by (u, v). Rectangular or Gaussian functions can be used for the window function. The aim of the Harris corner detector is to maximise $E(u, v)$ which is the difference between the original pixel intensities and the pixel intensities in the shifted window. This is achieved using the first-order approximation of the Taylor expansion of equation 2.1, the results of which can be seen in equation 2.2 [45].

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{2.3}$$

$$R = \det(M) - k\,(trace(M))^2 \tag{2.4}$$

$$\det(M) = \lambda_1 \lambda_2 \tag{2.5}$$

$$trace(M) = \lambda_1 + \lambda_2 \tag{2.6}$$

Equation 2.3 shows that M is a 2x2 matrix which is computed using the products of the image derivatives I_x and I_y. Harris et al. described a cornerness function R as shown in equation 2.4. This can be computed using the determinant and trace of M as given by equations 2.5 and 2.6 respectively, where λ_1 and λ_2 are the eigenvalues of M. The function R can be used to determine if the change in intensities represents a flat region, an edge or a corner.

If λ_1 and λ_2 are both small, $|R|$ will also be small, which means that the region is flat since there is no change in pixel intensities in any direction. Edges only occur when $\lambda_1 >> \lambda_2$ or vice-versa, so $R < 0$ which represents a change in only one direction. Corners are only present when λ_1 and λ_2 are both large and $\lambda_1 \approx \lambda_2$ which causes R to be large as there is a significant change in intensities regardless of the direction in which the window was shifted.

The Harris corner detector has proven to be useful to detect corners which are invariant to rotation. Shifting a window along a rotated direction will still produce the same difference in intensities for a corner. However, the Harris corner detector is not invariant to scale. Hence, other feature description methods have been explored as alternative options.

2.3.3 SIFT

Scale-Invariant Feature Transform (SIFT) [46] is a feature description method that identifies points of interest within an image which are characterised by feature vectors. SIFT uses local features which are both scale and rotation invariant. SIFT features are also robust to variations in light intensity within an image.

Scale-space extrema detection is used to consider different scales via a difference of Gaussians (DoG) filter [47]. Keypoints are localised at potential features to improve the estimates for the location and scale. Orientations are then assigned to the features using the directions of neighbouring points which form a histogram of gradients for each sub-region within an area of the image. The size of the SIFT feature descriptor vector will be determined by the number of sub-regions as well as the number of bins used in the histograms. To maintain rotation invariant features, the dominant orientation in the histogram will be selected based on the gradient magnitude which receives the majority vote from the pixels in the neighbourhood.

However, there are several calculations required to determine the gradient magnitudes and orientations for SIFT descriptors. This means that actual implementations of SIFT are often quite slow and therefore unsuitable for SLAM applications, in which real-time behaviour is desirable.

2.3.4 SURF

Speeded Up Robust Features (SURF) [48] is a feature description technique designed to improve the speed of SIFT. Instead of calculating gradient magnitudes in multiple orientations as seen in the SIFT algorithm, SURF calculates gradients solely in the x and y directions through the use of Haar wavelets as shown in Fig 2.6.

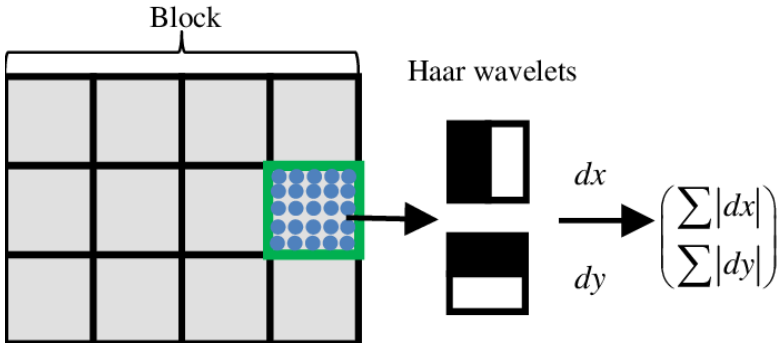

Figure 2.6: Haar wavelets in SURF feature descriptor [49].

Since there are only two orientations to consider, horizontal and vertical, pixel intensities are weighted with either $+1$ or -1. Summing these is therefore much faster for a given sub-region, thus allowing SURF to be significantly faster than SIFT. Despite this, SURF feature descriptor vectors are still quite large and feature correspondence techniques often struggle to match SURF features when there are large angles of rotation involved.

2.3.5 BRIEF

Binary Robust Independent Elementary Features (BRIEF) [50] is a feature description technique designed to improve upon the performance of SURF to achieve even faster speeds. Image patches are used to obtain binary strings as the features, which can be efficiently matched using the Hamming distance [51], which measures the number of places in which the binary strings differ in values.

$$\tau(p; x, y) = \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

The smoothed intensities $p(x)$ and $p(y)$ of randomly sampled pixels within a patch p are compared against each other to result in a binary output as shown in equation 2.7. n_d pairs of points x and y are randomly sampled to lead to an n_d-dimensional bitstring, where n_d is typically 128, 256 or 512.

Nevertheless, despite the accelerated computation, BRIEF descriptors are neither scale nor rotation invariant. Robotic platforms moving within forest environments will inevitably rotate in order to avoid obstacles and navigate difficult terrain. Different scales will also be present in forest scenes, since background foliage will be much further away than trees that are close to the camera. As such, BRIEF descriptors are not suitable to use as features within a V-SLAM system designed to operate within forest environments.

2.3.6 FAST

Features from Accelerated Segment Test (FAST) [52] is a feature description method aimed at improving the speed of the Harris corner detector. Corners are identified based on the brightness of 16 pixels that form a ring around a pixel p that could be a potential corner within a given image patch. If the intensities of n pixels surrounding p are brighter than $I(p) + t$ or darker than $I(p) - t$ where t is a threshold, p can be considered a corner.

The top, bottom, left and rightmost pixels within the ring are evaluated first to quickly reject non-corners. If more than one of these pixels do not meet the brightness criteria, the candidate is not a corner. However, the high-speed test struggles when $n < 12$ and also depends on the distribution of corners. Hence, a machine learning model was used in order to increase reliability, in which a decision tree was trained to successfully classify corners. FAST converts the decision tree into if-then-else statements in C to identify corners quickly.

Although FAST exhibits the real-time performance required to include it as part of a SLAM system, the use of corners as features is sub-optimal for forest scenes. Forest environments rarely contain objects with well defined corners that can easily be identified and tracked between frames. Therefore, other types of features have been explored as more robust alternatives for forest environments.

2.4 Challenges with Visual SLAM in Forests

2.4.1 Camera Model

Figure 2.7: RGB camera model [53].

RGB cameras consist of a lens, Bayer filter, image sensor and Image Signal Processor (ISP) as shown in Figure 2.7. Light passes through the lens onto the Bayer filter where pixels are filtered to represent either red, green or blue, as this is required for colour images. The main types of image sensors are Charge Coupled Device (CCD) and Complementary Metal Oxide Semi-conductor (CMOS). The image sensor contains receivers that convert photons of light into electrical signals. These are then discretised into numerical values relating to the pixel intensities. Finally, the ISP performs tasks such as denoising, lens distortion correction and encoding to achieve the final image [53].

2.4.2 Dynamic Lighting Conditions

Lighting conditions are especially variable in forests due to dense vegetation. Tree canopies often block much of the direct sunlight causing the forest floor to be darker than other areas. Some areas within the forest might have more vegetation and other areas might have less. This results in rapidly changing light and dark areas which can be problematic for vision sensors [54]. High dynamic range (HDR) is usually required to ensure that the camera captures image details in the bright areas in the environment as well as the shadows. White balance and ISO also need to be managed correctly to ensure that the resulting image is exposed correctly.

2.4.3 Monotonous Textures

Vision sensors interpret textures as local spatial variations within an image, including variations in colour and pixel intensities [55]. Forest scenes often contain monotonous textures, particularly when the ground is blanketed in indistinguishable grass or leaves which results in a lack of visual diversity. This means it is difficult for traditional feature correspondence methods to distinguish between identified features in areas of the images that look the same.

2.4.4 Motion Blur

Atmospheric conditions, like the wind, cause movement among objects in forest landscapes, such as swaying tree branches and fluttering leaves. In addition to this, the camera itself will be moving relative to the environment during SLAM applications. Factors such as the speed of the camera movement and whether the camera platform is stabilised will affect the quality of the footage captured. This could result in undesirable motion blur, in which areas of individual frames might appear blurry with a noticeable lack of detail [56]. Robust feature correspondence relies on sufficient detail being available in each frame to estimate the camera pose and locate the position of objects within the scene. Motion artefacts can inhibit the ability of the feature correspondence method to locate the new positions of features once they have moved. As a result, forest landscapes prove to be challenging environments for V-SLAM systems to work effectively.

2.5 Related Work

2.5.1 Tree Parameter Extraction Model

Mapping a forest environment relies heavily on accurate feature correspondence which is why Benny explored the use of tree trunks as robust features that can easily be tracked between frames [57]. Tree trunks are stationary landmarks in dynamic forest scenes, which makes them ideal to use as robust features. The Tree Parameter Extraction Model (TPEM) is a machine learning model that was trained to identify tree trunk diameters, inclination of trees and felling cut positions as distinctive landmarks to result in reliable keypoints along the tree trunks.

ResNet-50, ResNet-101 and ResNeXt-101 [58, 59] were compared as the backbone architectures for the TPEM. To reduce overfitting, models were pre-trained on a synthetic dataset (SynthTree43k) before being fine-tuned on the CanaTree100 dataset [60]. However, all three models exhibited better performance without pre-training on the synthetic dataset first, and ResNeXt-101 had the highest average precision (AP) score so it was selected as the backbone architecture of choice.

Each identified keypoint has a corresponding x and y coordinate which is converted into an OpenCV keypoint object. BRIEF descriptors are assigned to these tree trunk keypoints to allow tracking across consecutive frames as shown in Figure 2.8.

However, the TPEM uses the base of the tree trunks to determine the inclination parameter. If the base is occluded, keypoints can be positioned incorrectly, which results in incorrect feature correspondences. Incorrect positioning of keypoints on the edges of the tree trunks can cause background and foreground elements of the scene to be confused with each other, especially when depth values are included. As such, Benny also included ORB features to improve robustness.

Figure 2.8: TPEM feature correspondence in forest environment [57].

2.5.2 ORB

Oriented FAST and Rotated BRIEF (ORB) [61] features combine the advantages of the speeds of FAST and BRIEF with the added benefit of rotation invariance. FAST is used to identify keypoints, with the N best keypoints being selected based on the responses from the Harris corner detector. To ensure that features are considered at multiple scales, a scale pyramid of the image is used.

ORB then employs the use of intensity centroids to consider orientations, as this is not done by FAST. An image patch is considered around a corner point, where the moments of the patch are calculated using Equation 2.8.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.8}$$

These moments are then used to compute the centroid as shown in Equation 2.9.

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{2.9}$$

A vector is formed from the centre of the corner to the centroid. The orientation can therefore be determined by the direction of this vector as given by Equation 2.10.

$$\theta = atan2(m_{01}, m_{10}) \tag{2.10}$$

BRIEF feature descriptors are then used, although traditional BRIEF does not perform well when there is rotation involved. Therefore, ORB steers the BRIEF descriptors relative to the orientation of the keypoints. ORB feature correspondence involves multi-probe Locality Sensitive Hashing (LSH) [62] where hash tables are used to store hashed feature points in distinct buckets. During feature correspondence, neighbouring buckets are selected and the elements are compared using brute-force matching to find the features. Figure 2.9 shows ORB feature correspondence being used to match features in a forest scene.

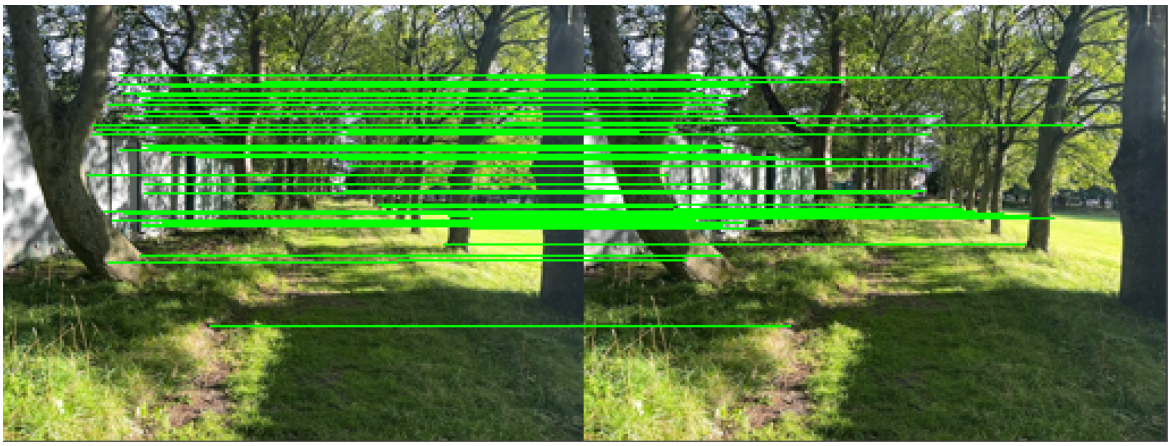

Figure 2.9: ORB feature correspondence in forest environment [57].

Benny experimented with solely using ORB features as well as ORB features in conjunction with the features from the TPEM. It was shown that the inclusion of the tree keypoints reduced both the translation and rotation errors of the camera pose estimates, since the tree trunks provided reliable landmarks. However, the state-of-the-art SuperGlue feature correspondence technique outperformed both methods.

2.5.3 SuperGlue

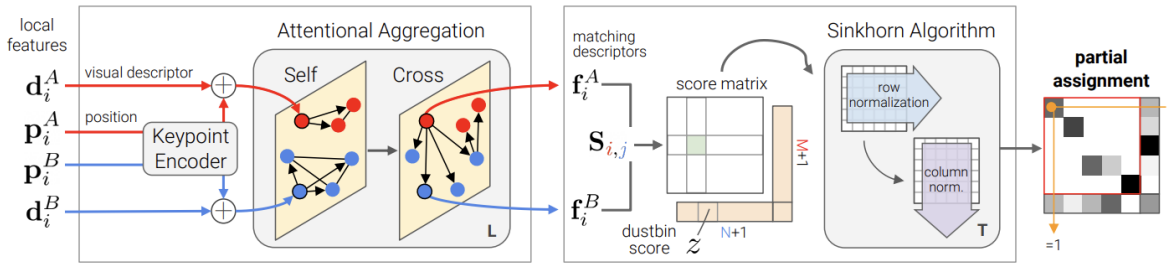

Figure 2.10: SuperGlue architecture [11].

Sarlin et al. proposed SuperGlue [11], a feature correspondence technique that differs greatly from traditional methods. It attempts to understand the geometric transformations of the underlying 3D scene, which makes it ideal for challenging outdoor scenarios such as forest environments. SuperGlue is based on a graph neural network architecture which learns to correctly match features and reject any points that cannot be matched. It can match many different types of low-level features but is typically used to match SuperPoint features [10], which are interest points that have been identified via a self-supervised framework. SuperPoint uses Homographic Adaptation (HA) to consistently identify a more extensive range of interest points, providing pixel-wise locations and descriptors for each identified keypoint and outperforming traditional hand-crafted feature detectors.

SuperGlue consists of two main aspects, an attentional graph neural network (GNN) and an optimal matching layer as shown in Fig 2.10. A Multi-Layer Perceptron (MLP) is used to form a high-dimensional vector containing the x and y position coordinates of the keypoints as well as their descriptors. The use of self and cross attention enables the graph neural network to interpret both the appearance and position of the keypoints. Attentional aggregation is used to build a graph between the keypoints that form the nodes. This allows SuperGlue to focus on specific keypoints for certain attributes and to consider other similar keypoints nearby. This learning based approach is particularly effective, which is why SuperGlue outperforms traditional feature correspondence algorithms, since they cannot exhibit the same type of behaviour to understand the underlying scene.

The second part of SuperGlue is used to solve an optimal transport problem where the scores for the predictions of matching descriptors are maximised. Unmatched keypoints are discarded into dustbins, which allows SuperGlue to perform better in scenarios that involve occlusions. This is particularly useful in forest environments, since there might be occlusions at certain camera angles due to dense foliage. Optimisation of the score matrix is implemented through the use of the Sinkhorn algorithm. This allows SuperGlue to perform real-time feature correspondences, since all components of the SuperGlue architecture are differentiable, which means computations can be done efficiently using GPUs. This makes SuperGlue a suitable choice as a feature correspondence technique for use within a V-SLAM system, since real-time behaviour is desirable.

Figure 2.11 shows the SuperGlue feature correspondences in a forest scene. When comparing the different feature correspondence methods on the same forest scene, it is clear that SuperGlue outperforms both the TPEM and ORB. The distribution of keypoints in the scene is far more spread out in Figure 2.11 compared to both Figures 2.8 and 2.9. This shows that SuperGlue is able to understand the underlying features of the scene in a much more comprehensive way.

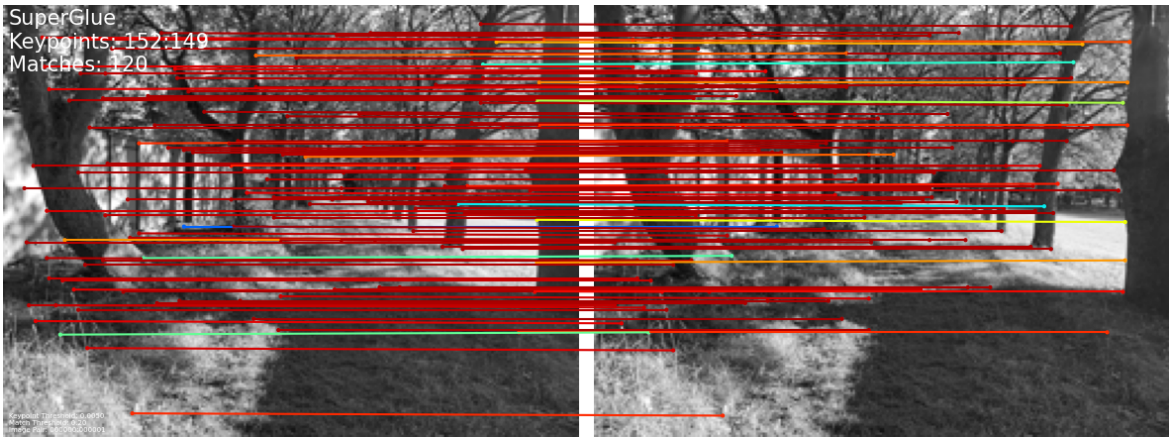

Figure 2.11: SuperGlue feature correspondence in forest environment [57].

Since the TPEM focuses solely on tree trunks, often there are very few keypoints to match, and frames in which no tree trunks are present will result in no feature correspondences. ORB feature correspondences tend to focus on more discernable aspects of the environment such as the trees and the wall. Feature correspondences that are close together in one part of the frame can be problematic, especially if that part of the frame is occluded in subsequent frames.

SuperGlue matches points not only on the trees and the wall, but also on the ground. Several feature points on the blades of grass were successfully matched by SuperGlue but ignored by the other feature correspondence methods. In forest environments, it is important to have feature correspondences that cover the entire frame. Since the scenes are very dynamic and can change dramatically from one frame to the next, it is unreliable to focus on one specific part of the scene.

Figure 2.12 shows the absolute trajectory errors (ATE) calculated for each of the feature correspondence methods that Benny tested. SuperGlue had the lowest RMSE error for both translation and rotation. However, since the data collection setup involved the use of the iPhone Stray Scanner app, the accuracies of the ground truth measurements are unclear. Nevertheless, the relative performance of SuperGlue has shown it to be the best feature correspondence method out of the ones which were tested.

	Translation Error (m)			**Rotation Error (deg)**		
	Min	Max	RMSE	Min	Max	RMSE
ORB	0.383	3.471	2.016	7.775	21.211	14.142
SuperGlue	0.275	3.638	1.871	7.618	17.051	11.749
Tree Parameters: TPEM + ORB	0.812	7.712	3.944	6.840	73.124	38.570
Tree Parameters: ORB	0.419	13.609	6.381	8.229	83.135	45.405

Figure 2.12: Trajectory errors for different feature correspondence methods [57].

2.5.4 ORB-SLAM3

Campos et al. proposed ORB-SLAM3 [14], a state-of-the-art V-SLAM system based on the ORB feature detector, integrating visual, visual-inertial and multi-map capabilities. ORB-SLAM3 uses a bag-of-words representation to store the ORB keypoints and descriptors, allowing for efficient retrieval of similar keyframes for matching. Keyframes are analysed geometrically within an image window to verify if the ORB features match, which is achieved by computing the Hamming distance between descriptors. RANSAC [63] and bundle adjustment is then applied to minimise the reprojection error between the current keypoints and the aligned map points.

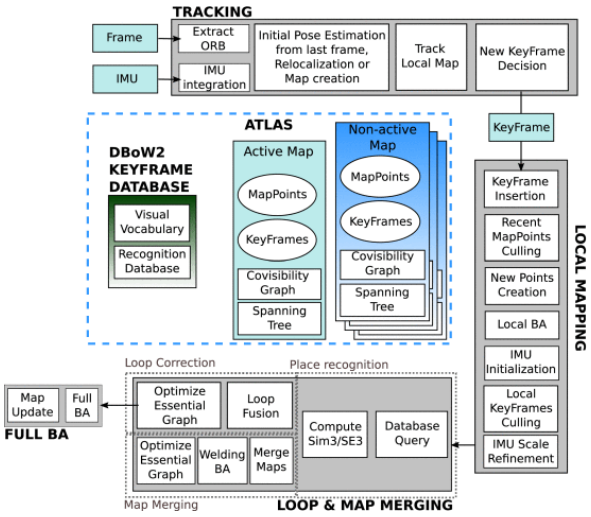

Figure 2.13: ORB-SLAM3 architecture [14].

Figure 2.13 shows the ORB-SLAM3 system architecture. It is highly accurate as shown by its low RMS-ATE scores on the trajectories in the EUROC and TUM-VI benchmarks. Nevertheless, Campos et al. identified low-texture environments as the main failure case for ORB-SLAM3. Forest scenes typically include repetitive features like grass or leaves which can provide challenging conditions for the ORB features to be detected and matched correctly. Swaying tree branches and changes in illumination also contribute to this, resulting in ORB-SLAM3 performing less well in forest environments. This highlights the need for a more robust feature correspondence method to deal with these challenges in order to provide reliable information to the pose estimation aspect of a V-SLAM system for forest environments.

2.5.5 VINS-Mono

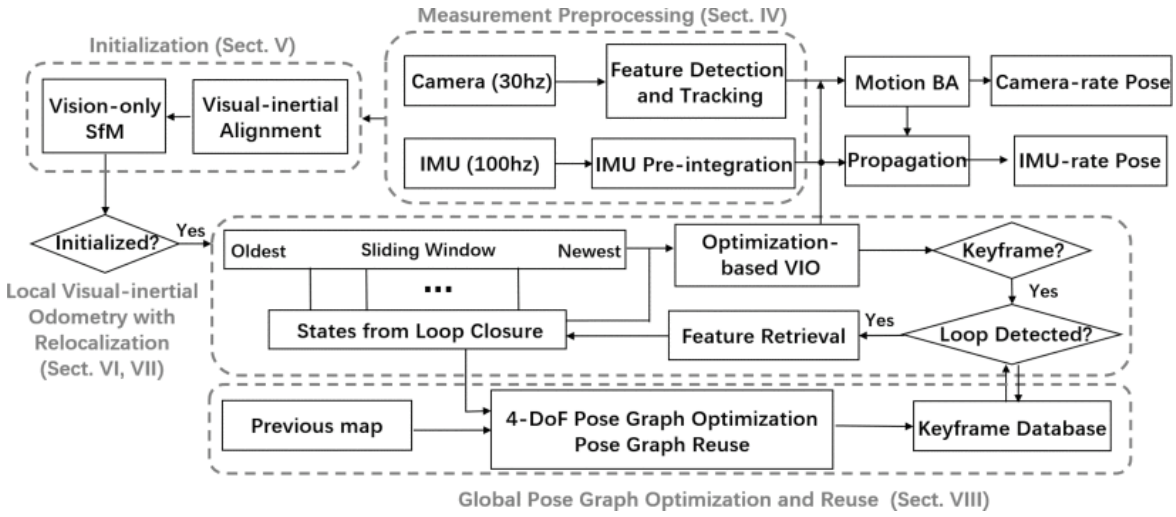

Figure 2.14: VINS-Mono architecture [15].

Qin et al. proposed VINS-Mono [15], shown in Figure 2.14, a state-of-the-art visual-inertial (VI) SLAM system that combines data from a monocular camera with an IMU for accurate state estimation. The exact feature detector used is not stated in the paper, however it is mentioned that an optic flow method is used to match features between frames. VINS-Mono uses a robust initialisation process that can start from unknown initial states and includes both bundle adjustment and loop closure detection for accurate mapping and localisation.

It performs very well on the EUROC dataset and is capable of being part of an on-board autonomous flight system. However, pre-integrating the IMU data can result in incorrect pose estimations since the IMU data can be noisy. Additionally, optic flow can fail when there are large changes in motion between frames and can be easily confused by swaying branches in forest scenes.

State-of-the-art V-SLAM systems like ORB-SLAM3 and VINS-Mono do not perform as well on sequences involving unstructured environments such as forests. Current feature detectors and matchers including ORB and optic flow methods are not robust enough to deal with dynamic objects, illumination changes and monotonous textures, all of which are abundant in forest scenes. Therefore, we incorporate SuperPoint and SuperGlue as robust feature detection and correspondence methods for our Forest SLAM system.

Chapter 3

Dataset

3.1 Requirements

In order to evaluate a SLAM system designed for forest environments, we require a dataset that contains scenes with natural foliage like trees and plants as opposed to the structured urban scenes typically present in most available SLAM datasets. The absence of well-defined features in unstructured environments creates difficult conditions for existing V-SLAM approaches to match features across frames and perform accurate localisation. However, there is a distinct lack of forestry datasets available that are suitable for evaluating SLAM systems. Datasets like FinnWoodlands [64] are aimed at machine learning models for semantic segmentation since they supply ground truth manual annotations for different types of trees and other objects. However, they do not offer ground truth pose information, which is necessary to evaluate the localisation accuracy of a SLAM system. FinnForest [65] is a dataset aimed at V-SLAM systems which contains sequences recorded by sensors attached to a car which was driven through roads in the forests of Tampere, Finland. Despite containing rural scenes and providing ground truth poses, the fact that reliable GNSS data was collected means that this dataset is not representative of typical forestry scenarios in which GPS data cannot be reliably obtained under the tree canopy. Synthetic datasets [21] also contain highly accurate ground truth poses but they lack the necessary photorealism to evaluate a SLAM system designed for real-world forest scenarios.

3.2 BotanicGarden

Liu et al. curated the BotanicGarden dataset [12] to address the challenges of SLAM in unstructured natural environments. It contains data captured by cameras, LiDAR and IMU sensors on a ground robot moving through a botanical garden of over 48000m^2 at Shanghai Jiao Tong University as shown in Figure 3.1. While not exactly the same, the thick woods and narrow trail scenes in the BotanicGarden dataset are analogous to those found in forest environments. The primary reason that we select the BotanicGarden dataset to evaluate our Forest SLAM system is because of the highly accurate ground truth poses with ~ 1cm accuracy.

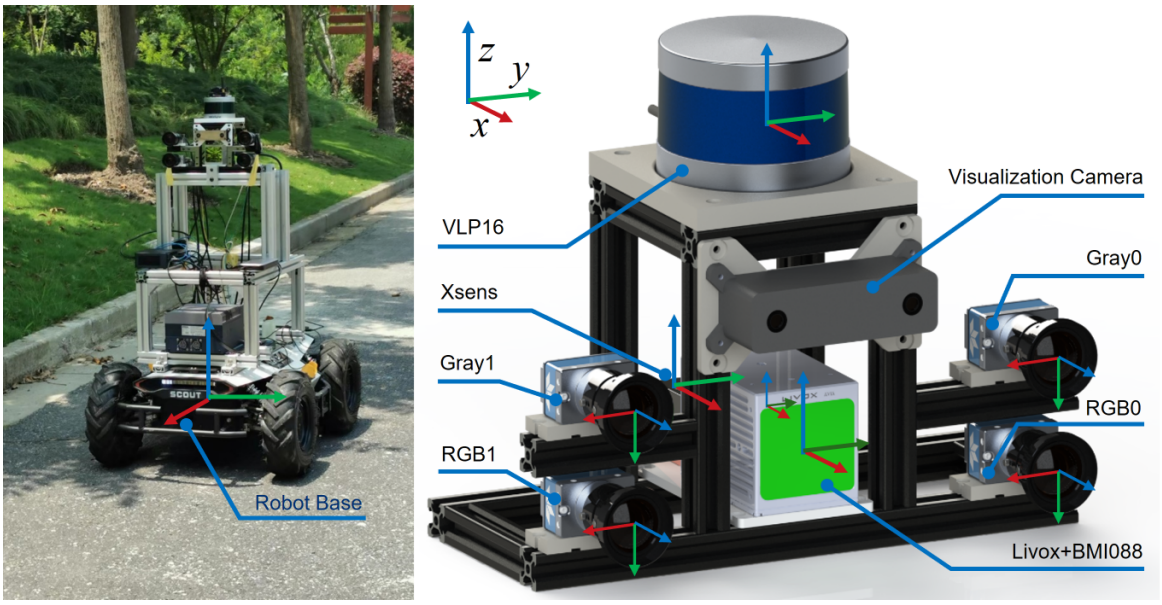

Figure 3.1: Sensor setup for the BotanicGarden dataset robot [12].

Data is provided in the form of rosbags with separate ROS topics [13] for each sensor used on the robot. We tailor our system to read data directly from these rosbags to more closely simulate how a SLAM system would be deployed on a physical robot. Currently, only seven recorded sequences are publicly available which contain a mixture of long and short sequences. Liu et al. also provide a leaderboard comparing the performance of current state-of-the-art SLAM systems on these seven sequences.

Table 3.1: Maximum number of adjacent image pairs in each sequence in the BotanicGarden dataset.

Sequence	Number of Pairs	Length (m)
1005-00	5645	601.603
1005-01	4325	479.729
1005-07	5183	591.040
1006-01	6976	765.801
1008-03	5980	750.087
1018-00	963	115.240
1018-13	1598	201.234
Total	30670	3504.734

We form adjacent image pairs from the image topics in the rosbags for feature correspondence since frames need to be co-visible with the same features being present in both frames in order for the feature matcher to function correctly. Table 3.1 shows the length of each recorded sequence along with the maximum number of adjacent image pairs which can be formed for feature matching. We use the rosbags containing downsampled vision data (960x600@10Hz).

Chapter 4

Forest SLAM

In this chapter, we:

- Provide an overview of the Forest SLAM system architecture.
- Describe the configuration used to determine the optimal settings for the SuperPoint and SuperGlue pre-trained models.
- Detail the mechanisms involved in our monocular and stereo methods.

4.1 System Architecture

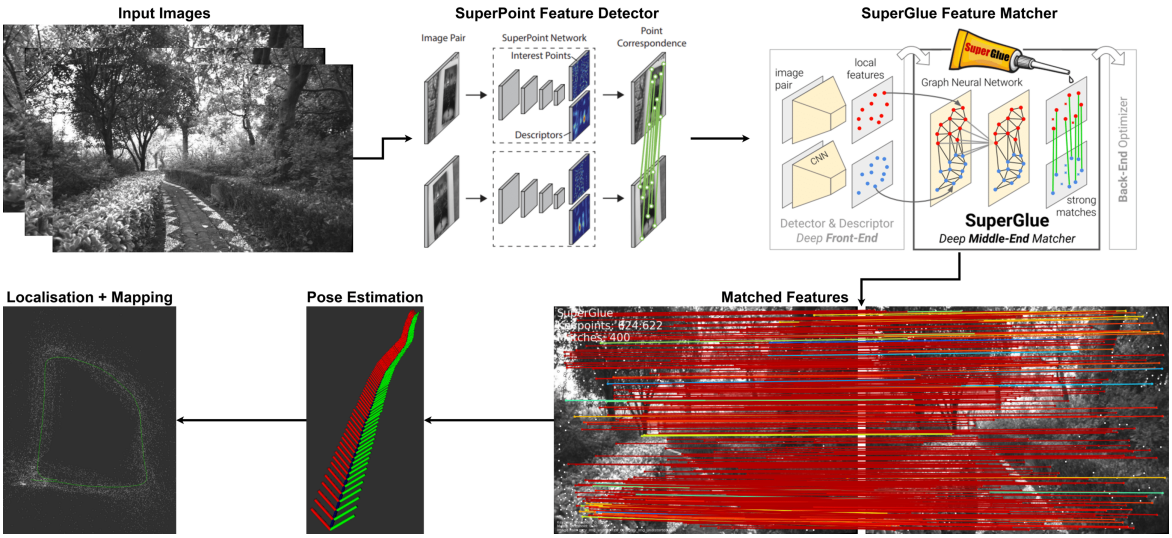

Figure 4.1: Forest SLAM system architecture.

Figure 4.1 shows the architecture for the Forest SLAM system, which we create from scratch. We extract input images from the BotanicGarden dataset rosbag sequences and form adjacent image pairs which are then passed through the default pre-trained SuperPoint and SuperGlue models. The matched features inform pose estimations, forming a cumulative trajectory by combining relative poses which are also used to transform the point clouds generated for each frame into a global map.

4.2 SuperPoint and SuperGlue Settings

The pre-trained weights for both the indoor and outdoor SuperGlue models are available. To ensure fair comparisons, we only consider the performance of the outdoor model, since all of the input images in the BotanicGarden dataset are of outdoor nature scenes. We test using both the default and recommended settings from Sarlin et al. [66] to get an accurate representation of SuperPoint and SuperGlue's performance on the BotanicGarden dataset. The recommended settings involve decreasing the image width to 960 pixels, increasing the maximum number of keypoints from 1024 to 2048 and reducing the non-maximum suppression radius from 4 to 3.

Since the ground truth pose information is available, epipolar lines can be drawn from the features in the first frame, which show where the features should lie in the second frame based on the geometric relation between frames. The matched features in the second frame can be deemed to be correct if they lie closely to the epipolar line. We evaluate SuperGlue feature correspondences across pairs of images using the ground truth poses for the seven available rosbag sequences in the BotanicGarden dataset. We align the */dalsa_rgb/left/image_raw* and */gt_poses* topics using their timestamps to ensure that the poses accurately match the images. We provide image pairs to SuperGlue in sequential order as opposed to randomly selected pairs to follow the expected sequence of inputs encountered in a real-world SLAM scenario. This ensures that the same features can be seen across both frames.

The SuperGlue matches are evaluated using both precision and matching score which are given by Equations 4.1 and 4.2 respectively.

$$\text{Precision} = \frac{\text{Number of Correct Matches}}{\text{Total Number of Estimated Matches}} \tag{4.1}$$

$$\text{Matching Score} = \frac{\text{Number of Correct Matches}}{\text{Total Number of Detected Keypoints}} \tag{4.2}$$

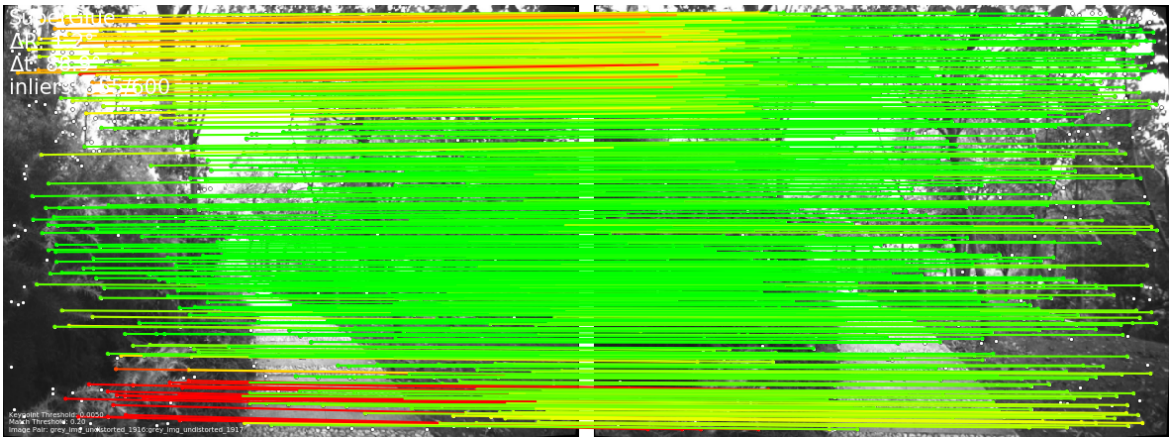

Figure 4.2: Epipolar errors for the SuperGlue matches using the default settings. Green matches indicate lower epipolar error whilst red matches indicate higher epipolar error.

Figure 4.2 shows the epipolar errors visualised for the SuperGlue outdoor model with the default settings. Most of the feature matches are green which indicates that the matches lie closely to the epipolar lines and are matched correctly.

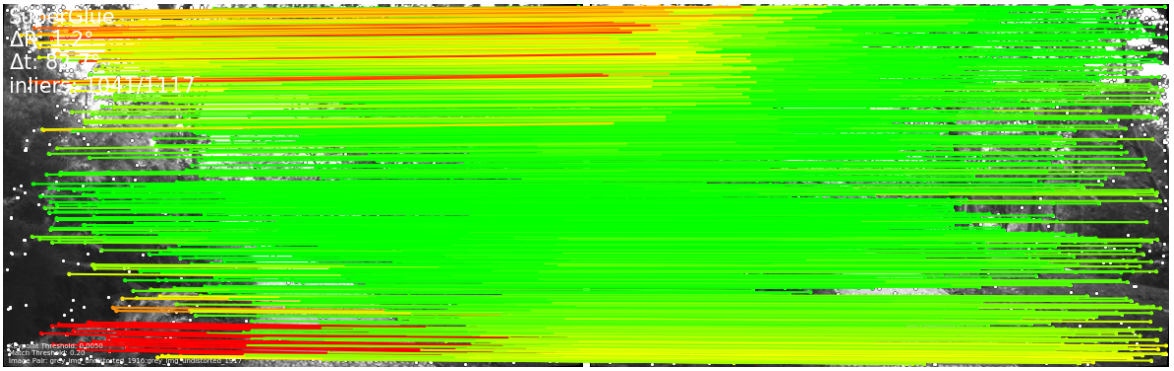

Figure 4.3: Epipolar errors for the SuperGlue matches using the recommended settings. Green matches indicate lower epipolar error whilst red matches indicate higher epipolar error.

Figure 4.3 shows the epipolar errors visualised for the SuperGlue outdoor model with the recommended settings. There are more incorrect matches compared to the default settings as shown by the increase in red and yellow lines in the bottom left and top left corners of the frame.

We average both precision and matching score across all seven available sequences to compare the overall performance of the pre-trained SuperPoint and SuperGlue models on the BotanicGarden dataset. Using the recommended settings for the pre-trained models yielded no improvement over the default settings and actually resulted in worse performance for both precision and matching score as shown in tables 4.1 and 4.2. This could be due to the fact that Sarlin et al. focused on urban outdoor scenes which means the recommended settings might not offer better performance when deploying SuperPoint and SuperGlue in unstructured natural environments. Therefore, we use the default settings when incorporating the pre-trained SuperPoint and SuperGlue models into the Forest SLAM framework.

Table 4.1: SuperGlue precision on the BotanicGarden dataset for each model averaged across the number of pairs in each input sequence.

Sequence	default	recommended
1005-00	**81.90**	81.81
1005-01	**88.24**	88.16
1005-07	78.97	**78.98**
1006-01	76.78	**76.87**
1008-03	**75.17**	75.16
1018-00	**83.13**	83.03
1018-13	**79.22**	78.94
Average	**80.49**	80.42

Table 4.2: SuperGlue matching score on the BotanicGarden dataset for each model averaged across the number of pairs in each input sequence.

Sequence	default	recommended
1005-00	**61.24**	56.44
1005-01	**67.09**	61.52
1005-07	**57.41**	52.62
1006-01	**56.52**	51.93
1008-03	**54.71**	50.22
1018-00	**63.76**	58.80
1018-13	**57.84**	53.10
Average	**59.80**	54.95

4.3 Monocular Forest SLAM

Monocular SLAM involves using only a single camera to estimate the trajectory of the robot. The principle behind this is based on the pinhole camera model as shown in Figure 4.4. The 3D point in a scene with respect to the world coordinate frame, P_w, is projected onto the image plane to get the corresponding 2D pixel p using a perspective transformation based on the intrinsic parameters of the camera [67].

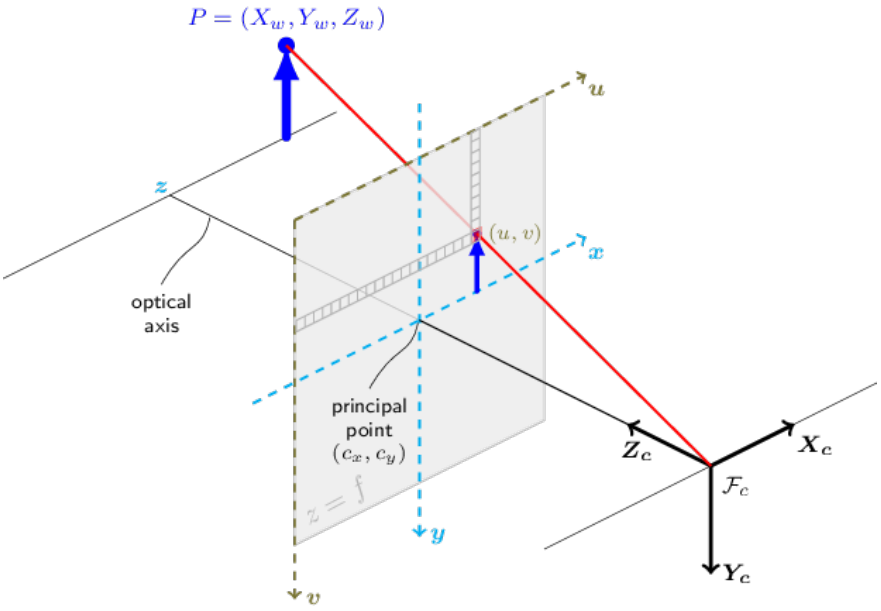

Figure 4.4: Pinhole camera model [67].

We begin by initialising a variable to store the previous image frame. We iterate through the rosbag selecting the current frame from the */dalsa_rgb/left/image_raw* topic. Since the images are stored as messages in the rosbag, we convert the image message to an OpenCV image, correcting any distortion as shown in Figure 4.5. We use the *cv2.undistort()* function which takes in both the intrinsic parameters and distortion coefficients from the camera model which are provided by Liu et al.

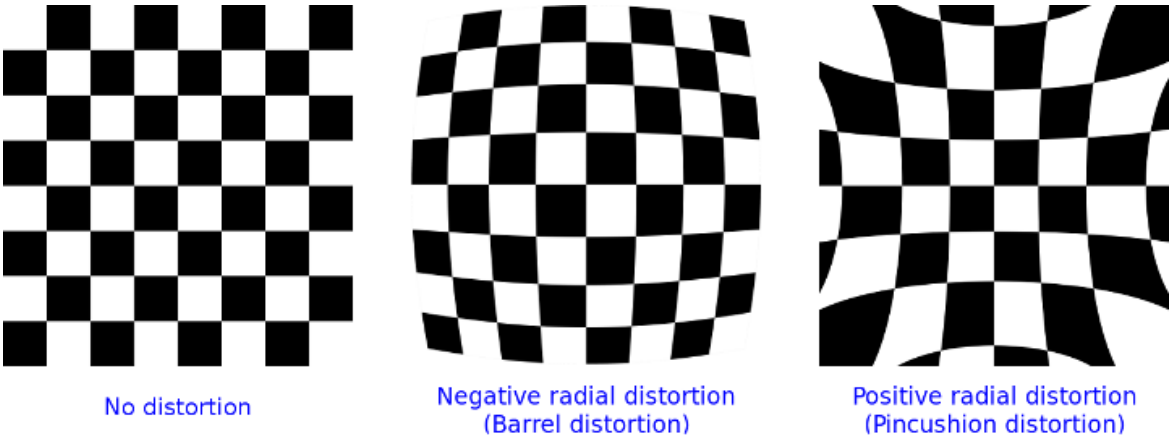

Figure 4.5: Image distortion examples [67].

The intrinsic parameters of the camera are usually expressed as a 3×3 matrix as shown in Equation 4.3 containing the focal lengths f_x and f_y given in pixels as well as the principle point (c_x, c_y) which denotes the optical centre of the image. We use the same intrinsic parameters K_0 and K_1 for the first and second frame being matched in each image pair since they have been captured by the same camera. The distortion coefficients contain the radial coefficients k_1, k_2 and k_3 as well as the tangential coefficients p_1 and p_2 as shown in Equation 4.4 [67].

$$K_0 = K_1 = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$\text{distortion coefficients} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{4.4}$$

After correcting image distortion, we convert the images to greyscale and move them to PyTorch [68] tensors before normalisation is applied to ensure the images can be used with the pre-trained SuperPoint and SuperGlue models. After passing the image pair through the models we obtain the matched keypoints in each frame which are given as (x, y) pairs denoting their locations within the image. We filter out any keypoints that are only seen in one frame and not the other, leaving only the correctly matched keypoints.

After obtaining the matched keypoints, the next step is to find the essential matrix. This encapsulates the geometric pose relation between two camera views of the same scene. Since we are using a monocular setup, the two views are obtained once the robot has moved and is viewing the same scene from a slightly different position. The fundamental matrix is similar to the essential matrix, but it assumes an uncalibrated camera in which the intrinsic parameters are not known. The fundamental matrix is typically found using the eight-point algorithm proposed by Hartley [69]. However, since we have the intrinsic parameters of the camera, we only need to find the essential matrix so we use the *cv2.findEssentialMat()* function which is based on the adapted five-point algorithm proposed by Nister [70].

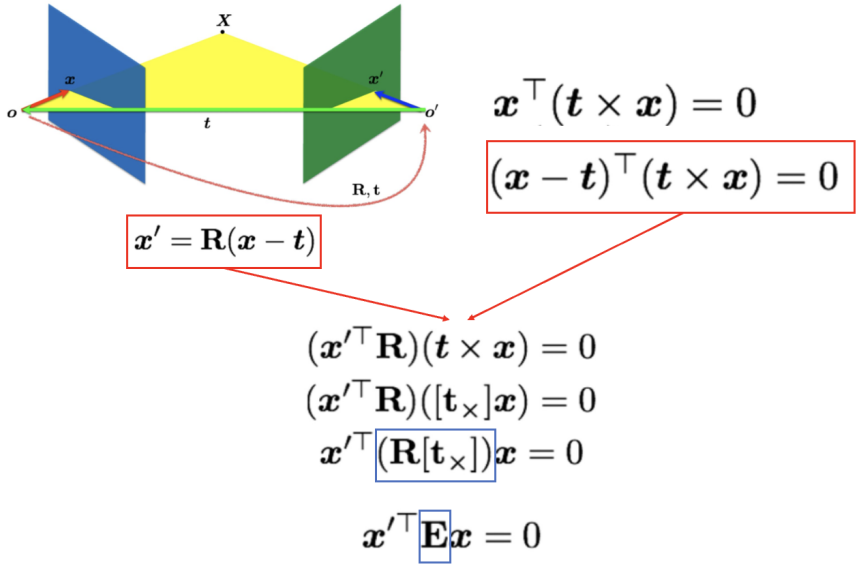

Figure 4.6: Essential matrix computation [71].

If x and x' are homogeneous normalised image coordinates in image 1 and 2 respectively, then x and x' correspond to the same 3D point in the scene if $x'^T E x = 0$, where E is the essential matrix as shown in Figure 4.6. The 3×3 essential matrix is a combination of the translation and rotation relating the same points between two images. It is shown in Equation 4.5, where R represents the rotation matrix and $[t_\times]$ is the matrix representation of the cross-product of t.

$$E = R[t_\times] \tag{4.5}$$

We use RANSAC [63] to reject outliers when using matched keypoints to estimate the essential matrix. We set the *threshold* to one pixel, so that any point which is further than one pixel from the epipolar line is rejected and not used as part of the essential matrix computation. Additionally, we set the *confidence* to 0.999 which is the probability that the essential matrix has been computed correctly.

Once we obtain the essential matrix, we use the *cv2.recoverPose()* function, which takes in the essential matrix, matched keypoints from both frames and intrinsic parameters as input. This returns the rotation as a 3×3 matrix R with a determinant of one, where α, β and γ represent the roll, pitch and yaw angles respectively, as shown in Equation 4.6.

$$R = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\ \sin(\alpha)\cos(\beta) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma) \end{bmatrix} \tag{4.6}$$

The translation is given as a 1×3 vector t containing the translations in the x, y and z axes respectively, as shown in Equation 4.7.

$$t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix} \tag{4.7}$$

$$T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

We combine the translation and rotation into a 4×4 transformation matrix T which gives the relative pose transformation between two frames as shown in Equation 4.8. The bottom row is set to $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ to ensure T is homogeneous.

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

Having initialised the starting pose to be a 4×4 identity matrix I as shown in Equation 4.9, we obtain the next pose by computing the dot product between I and the first T. Subsequently, the current pose is computed as the dot product between the previous pose and the current relative pose T. This results in cumulative pose estimations that together form the entire trajectory for the sequence.

We publish the current pose as a message on the *Path* rostopic to visualise the trajectories in real time using RViz [72] as shown in Figure 4.7. The *est_trajectory* message consists of the current timestamp, the frame_id, the positions and orientations. We set the frame_id to *map* which is the default used by RViz. The positions are provided as x, y and z coordinates. The rotations are provided as quaternions in the form x, y, z, w which we obtain using the *quaternion_from_matrix()* function from the *tf.transformations* ROS package.

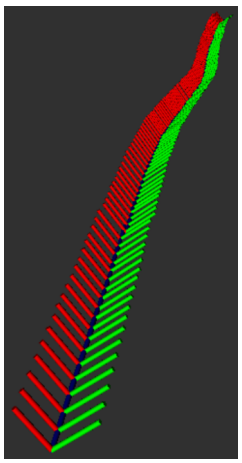

Figure 4.7: Cumulative poses visualised in RViz.

Due to the inherent limitations of monocular SLAM, it is not possible to directly recover the true scale of the trajectories using only a single camera. Therefore, we also investigate the stereo SLAM method.

4.4 Stereo Forest SLAM

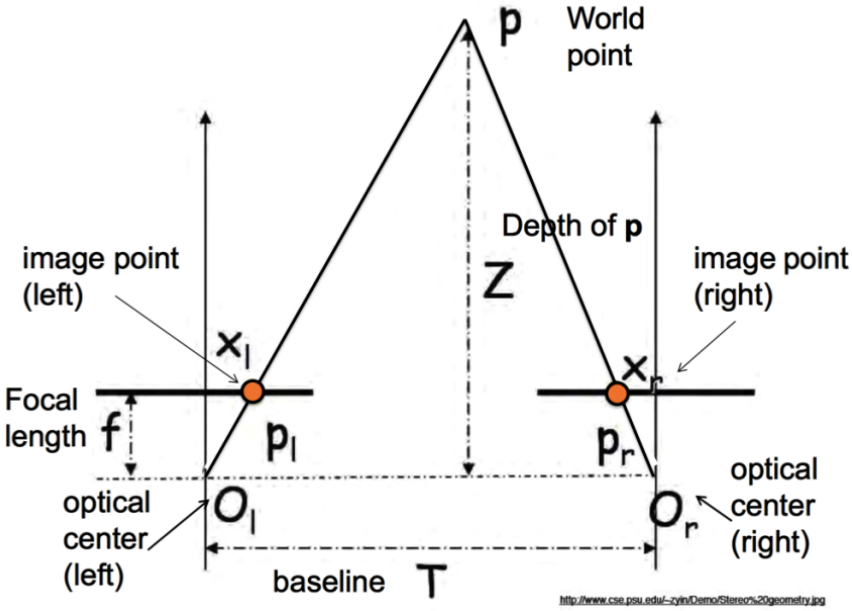

Figure 4.8: Stereo vision [73].

Stereo SLAM uses two cameras to view the same scene from different perspectives as shown in Figure 4.8, akin to human vision instead of a single camera as used in the monocular approach. In our case, we use the images captured by the left camera on the robot, */dalsa_rgb/left/image_raw* and the images captured by the right camera on the robot, */dalsa_rgb/right/image_raw*. As in the monocular approach, we retrieve the camera intrinsic parameters and distortion coefficients, although we use different ones for each camera now since they are no longer the same. Additionally, we require information about the transformation between the left and right camera coordinate frames, which is given by the 4×4 transformation matrix T_left_right for the robot used in the BotanicGarden dataset, as shown in Equation 4.10.

$$T\_left\_right = \begin{bmatrix} 0.999994564612669 & -0.00327143011166783 & -0.000410475508767800 & 0.253736175410149 \\ 0.00326819763481066 & 0.999965451959397 & -0.00764289028177120 & -0.000362553856124796 \\ 0.000435464509051199 & 0.00764150722461529 & 0.999970708440001 & -0.000621002717451192 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.10}$$

This enables understanding of the spatial relationship between the left and right cameras in the real-world. Therefore, stereo methods can use this information to recover the true scale of the trajectories by estimating the depth of the matched keypoints which cannot be done directly using only a single camera. We use $T\_left\_right[:3, 3]$ as the baseline T shown in Figure 4.8 which relates the physical translation offset between the left and right cameras. Estimating the depth of the real-world point P can be done using similar triangles as shown in Figure 4.9, resulting in Equation 4.11.

$$\frac{T}{Z} = \frac{T + x_r - x_l}{Z - f} \tag{4.11}$$

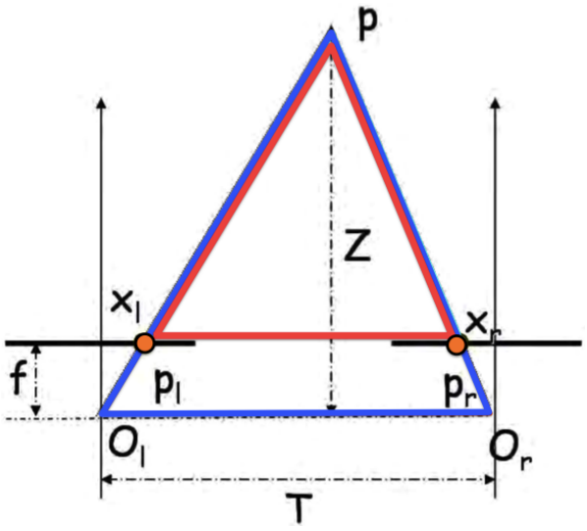

Figure 4.9: Similar triangles are used to compute the depth of point P [73].

$$Z = \frac{f \cdot T}{x_l - x_r} \tag{4.12}$$

Equation 4.11 can be rearranged to obtain the depth Z of point P as shown in Equation 4.12, where f is the focal length, T is the baseline and $x_l - x_r$ is the disparity. The disparity refers to the difference in horizontal positions (x-coordinates) of the corresponding pixels in two stereo images. In practice, we obtain the disparity map by passing the left and right images into the *cv2.StereoSGBM_create()* function. We set *numDisparities* to 96 which is the difference between the maximum and minimum disparity and needs to be divisible by 16. Since we set *minDisparity* to zero, *numDisparities* corresponds to the maximum disparity. We use a *blockSize* of seven which is the size of the window over which pixel disparities are considered and needs to be an odd number, usually three, five or seven. The two parameters which control the smoothness of the disparity, *P1* and *P2* are set to 392 and 1568 respectively. Larger values result in increased smoothness, with *P1* referring to the penalty on the disparity change by plus or minus one between neighbouring pixels. *P2* refers to the penalty on the disparity change by more than one between neighbouring pixels and $P2 > P1$. We specify the *mode* as *cv2.STEREO_SGBM_MODE_SGBM_3WAY* which uses three paths instead of five in the SGBM algorithm which increases speed at the expense of being slightly less accurate. To avoid division by zero errors as well as numerical instability, any values in the disparity map which are zero or minus one are set to 0.1.

We only use the right image to obtain the depth for the matched keypoints in the left image. Once we have the 3D x, y and z coordinates, we filter any extreme depth values and ensure there are at least six valid keypoints before performing pose estimation using only the matched keypoints from the left image. We use the *cv2.solvePnPRansac()* function which takes in the 3D points, the matched keypoints in the current left image as well as the intrinsic parameters and distortion coefficients for the left camera. We use a *confidence* of 0.99 and set the *iterationsCount* to 1000.

We specify the *reprojectionError* to be 1.0, which is the maximum permitted distance between the observed and computed point projections to consider it an inlier. We use the *cv2.SOLVEPNP_ITERATIVE* method which is based on Levenberg-Marquardt optimisation [74]. This identifies a pose that minimises the reprojection error which is the sum of the squared distances between the observed projections and the projected points. The estimated translations and rotations are provided as vectors, so we convert the rotation vector to a matrix using the *cv2.Rodrigues()* function. As before, we use the translation vector and rotation matrix to form a 4x4 transformation matrix and update the current pose by computing the dot product with the previous pose.

Since the stereo method allows us to obtain 3D points for the matched keypoints, we also perform mapping each time the pose is updated. At the start of the SLAM algorithm, we initialise a rospy publisher called *slam_map* which publishes to the *PointCloud2* topic. We publish the generated map as a 3D point cloud which we initialise as an empty list. At each iteration, after performing pose estimation, we use the estimated current pose to transform the 3D points to the global map coordinate frame. This is computed as a matrix multiplication between the transpose of the 3D homogeneous coordinates and the current pose transformation matrix. The converted points are stored in the list before converting the entire map to a point cloud message using the *PointField()* function from the *sensor_msgs.msg* ROS package. The updated map is published at each iteration and visualised using RViz. The map can be viewed alongside the trajectory as shown in Figure 4.10.

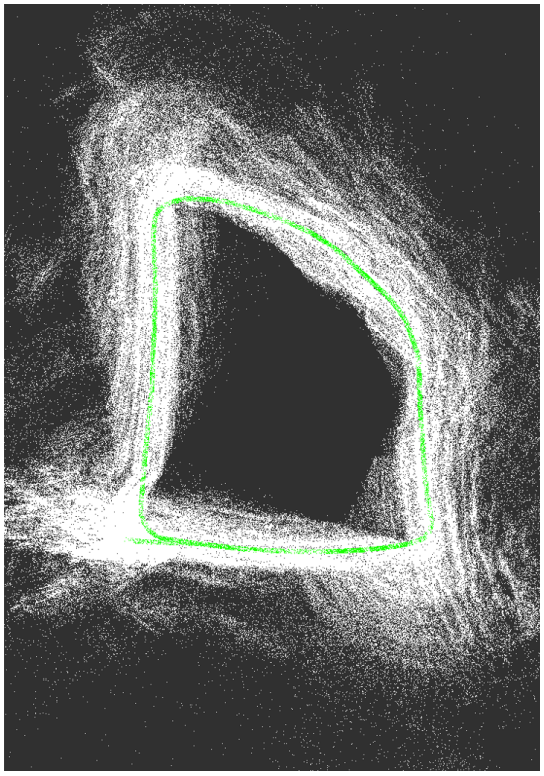

Figure 4.10: Generated point cloud SLAM map visualised alongside trajectory in RViz.

Chapter 5

Evaluation

In this chapter, we:

- Explain the optimal frame intervals used when evaluating Forest SLAM.
- Evaluate Forest SLAM against the current state-of-the-art on the BotanicGarden leaderboard using the Relative Pose Error (RPE) metric.
- Compare our monocular and stereo Forest SLAM approaches using RPE, XYZ errors, estimated velocities and ATE.
- Showcase the SLAM maps generated by our stereo Forest SLAM method.
- Perform an ablation study, substituting SuperPoint with ORB features and SuperGlue with a Brute Force (BF) matcher. We artificially add motion blur to the input images to establish SuperPoint and SuperGlue as the most robust combination in the context of a SLAM system for dynamic environments.

5.1 Forest SLAM Frame Intervals

Table 5.1: Frame intervals for monocular and stereo Forest SLAM.

Sequence	Monocular	Stereo
1005-00	15	10
1005-01	10	5
1005-07	5	5
1006-01	10	5
1008-03	5	5
1018-00	5	1
1018-13	10	1

We find that setting the frame intervals as a tunable parameter leads to better localisation when evaluating Forest SLAM on the BotanicGarden dataset. This is perhaps due to the fact that the robot moved at different speeds when the data was being collected. Using only adjacent frames means that the relative translations and

rotations are very small if there is little to no movement between frames. In addition, at the start of each recording, the robot is stationary for several frames before it begins its course. This could potentially cause the pose estimations to fail very quickly if the same features are seen in multiple frames without moving to different pixel locations. Table 5.1 shows the optimal settings for the frame intervals for both the monocular (mono) and stereo methods. The stereo method generally requires smaller frame intervals compared to the mono approach as it handles smaller translations and rotations better.

5.2 BotanicGarden Leaderboard

To compare the localisation capabilities of our mono and stereo Forest SLAM systems against the state-of-the-art (SOTA) on the BotanicGarden leaderboard, we use the open source evaluation tool Evo [75], as suggested by Liu et al. We only compare against the SOTA approaches using the Relative Pose Error (RPE) metric. RPE is a good metric to compare the localisation accuracy of a SLAM system as it remains unaffected to the length of the trajectory and is robust to the effects of occasional distance variations and drift accumulation. As performed by Liu et al. we compute the RPE over fixed-length segments of 20m for sequences 1018-00 and 1018-13 since they are less than 300m long, and 100m for all other sequences. In the Evo implementation, the RPE is the relative error between the estimated and ground truth relative poses across fixed-length segments expressed as a percentage. We use the median RPE for each sequence to reduce sensitivity to outliers.

The BotanicGarden leaderboard displays the median RPE of each system averaged across seven available sequences, the lengths of which can be seen in Table 3.1. We omit results for LiDAR based SLAM approaches and only compare against V-SLAM systems. Our stereo Forest SLAM system outperforms the stereo visual-only ORB-SLAM3 counterpart with respect to the RPE metric. This indicates that using SuperPoint and SuperGlue for the feature detector and matcher in a SLAM system results in more accurate relative pose estimations. However, our mono approach is clearly not suitable with a much higher RPE. The two visual-inertial methods result in the lowest RPE, highlighting the usefulness of IMU data to aid pose estimation accuracy.

Table 5.2: BotanicGarden leaderboard showing the RPE for different SLAM systems averaged across seven sample sequences expressed as a percentage.

Rank	Method	Setting	Relative Pose Error
1	VINS-Mono	(mono)Visual-Inertial	3.643 %
2	ORB-SLAM3	(stereo)Visual-Inertial	4.272 %
3	**Ours**	**(stereo)Visual-Only**	**4.718 %**
4	ORB-SLAM3	(stereo)Visual-Only	4.880 %
5	**Ours**	**(mono)Visual-Only**	**12.506 %**

5.3 Localisation

5.3.1 Generated Trajectories

We compare the trajectories generated by our mono and stereo methods on all seven sequences of the BotanicGarden dataset. However, since our focus is on robust relative pose estimations, we have not included any bundle adjustment or loop closure detection into our systems. Therefore, we only include the generated trajectories for our systems and omit results for SOTA approaches and we have already compared against these using the RPE metric which is the best metric for comparing relative poses. We only provide the generated trajectories for our system to showcase the benefits of a stereo method over a mono approach. When using Evo to generate trajectories from the estimated poses, we scale and align with the ground truth using Umeyama alignment since the mono approach cannot recover the real-world scale of the trajectories.

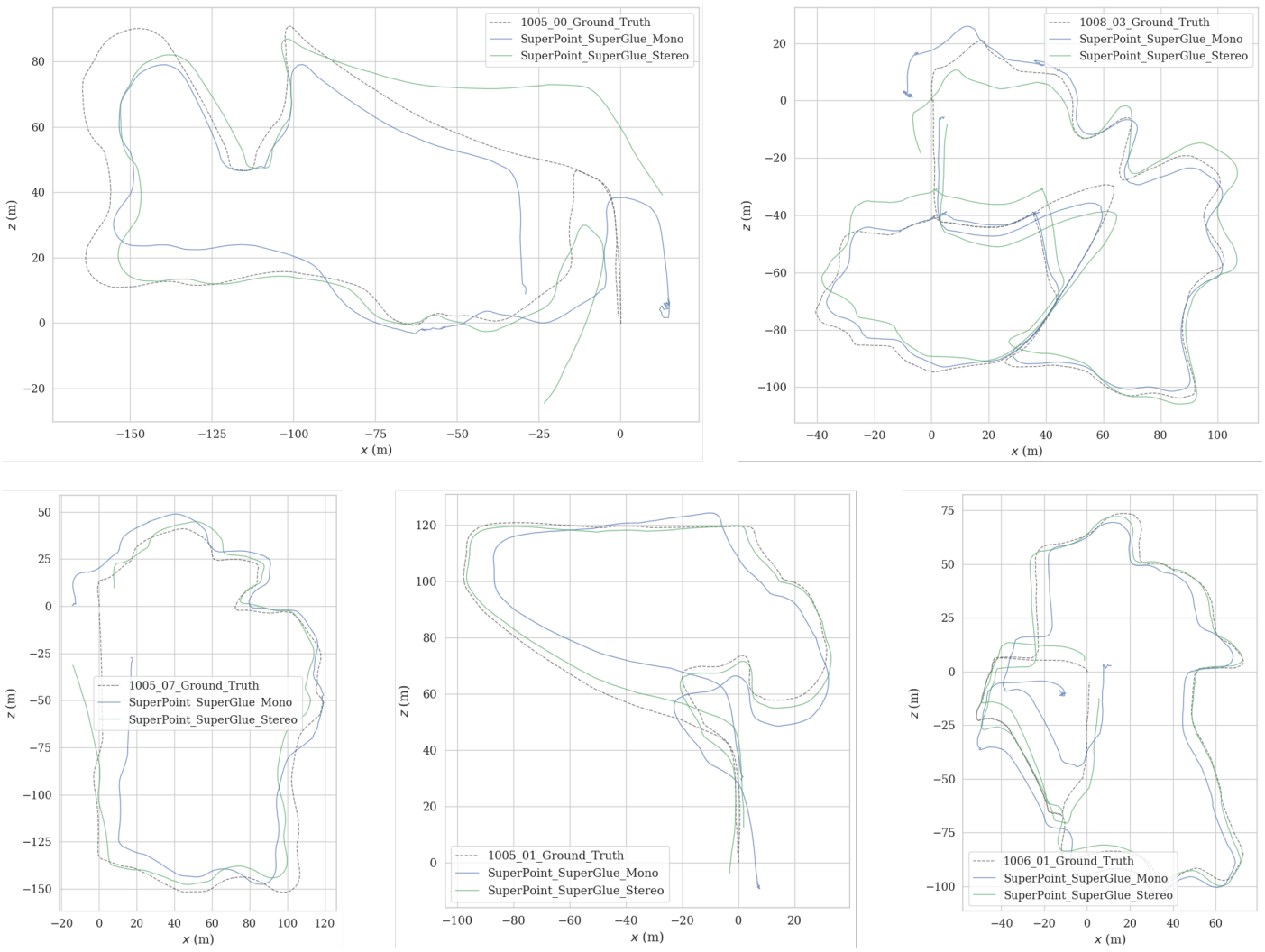

Figure 5.1: Generated trajectories for mono and stereo Forest SLAM using SuperPoint and SuperGlue on sequences longer than 300m in the BotanicGarden dataset.

Figure 5.1 shows the generated trajectories for the sequences longer than 300m in the BotanicGarden dataset. Generally, the stereo method follows the ground truth trajectory more closely, although the mono trajectory is better in sequence 1008-03. In sequences 1005-00 and 1005-07, both methods fail towards the end of the trajectory which is most likely due to a lack of loop closure detection.

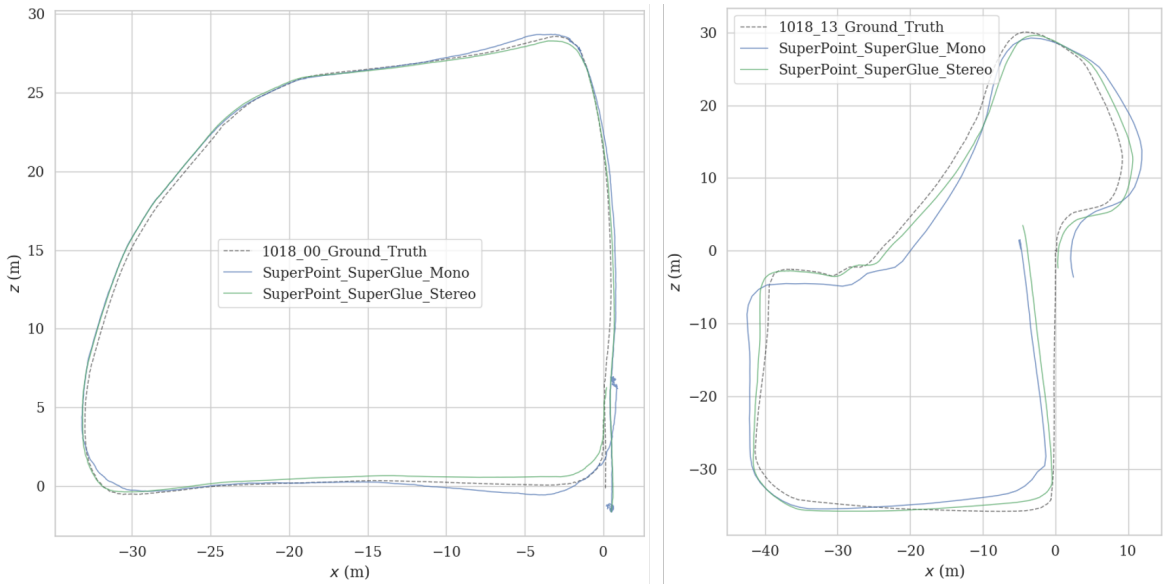

Figure 5.2: Generated trajectories for mono and stereo Forest SLAM using SuperPoint and SuperGlue on sequences less than 300m in the BotanicGarden dataset.

Figure 5.2 shows the generated trajectories for the sequences less than 300m in the BotanicGarden dataset. Both methods perform much better on these shorter sequences, with trajectories that align very closely with the ground truth. The stereo method still performs the best, although the mono approach is still suitable on these shorter sequences. There is still some minor drift on sequence 1018-13 but far less than on the longer sequences.

5.3.2 Relative Pose Error

Figures 5.3 and 5.4 show the RPE for the monocular and stereo methods respectively on sequence 1005-00. The stereo method clearly has a much lower root mean squared error (RMSE), mean and median RPE, although it does have a larger standard deviation compared to the mono method. Similar trends are seen in the shorter sequences as well which use smaller fixed-length segments of 20m instead of 100m to calculate the RPE. It is interesting to compare the RPE plots shown in Figures 5.5 and 5.6, since sequence 1008-03 contains lots of small turns instead of long straights which indicates why both methods showcase similar performance. Tighter turns result in larger motion changes between frames which could make pose estimation easier. Despite the mono approach having a slightly lower median RPE of 5.089% compared to 5.165% for the stereo method, the graphs actually show the stereo method to have a lower mean and RMSE.

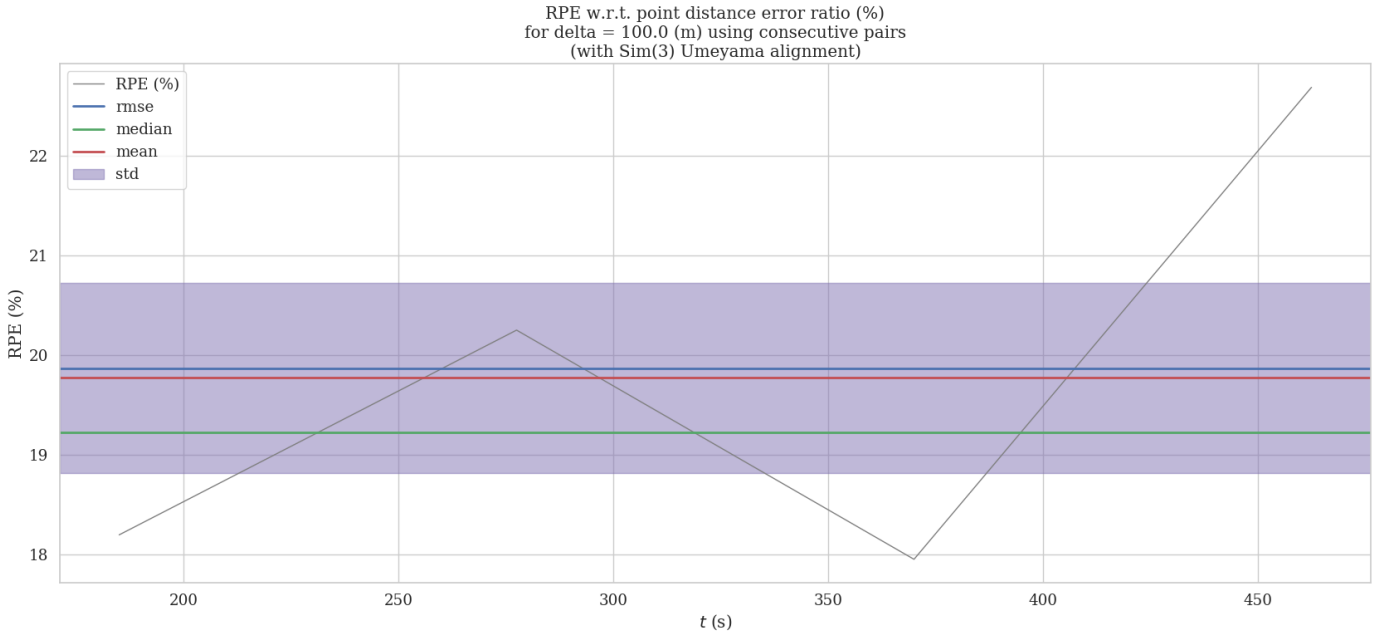

Figure 5.3: Mono Forest SLAM RPE on sequence 1005-07.

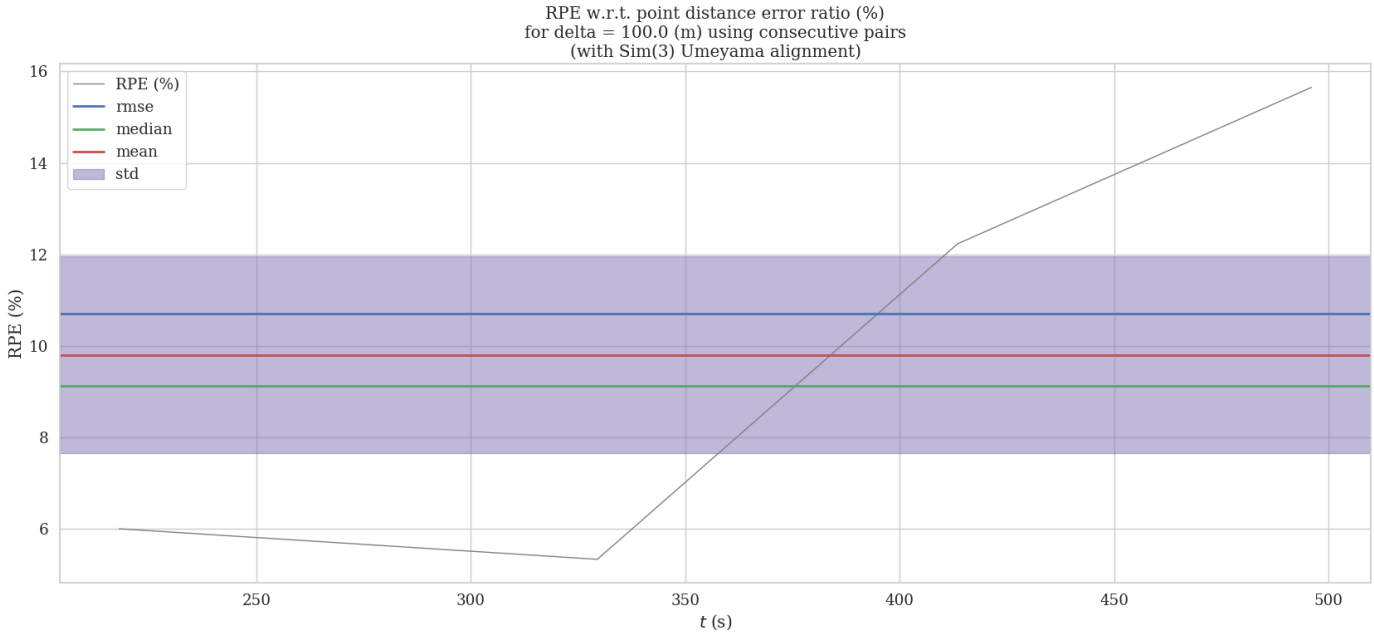

Figure 5.4: Stereo Forest SLAM RPE on sequence 1005-07.

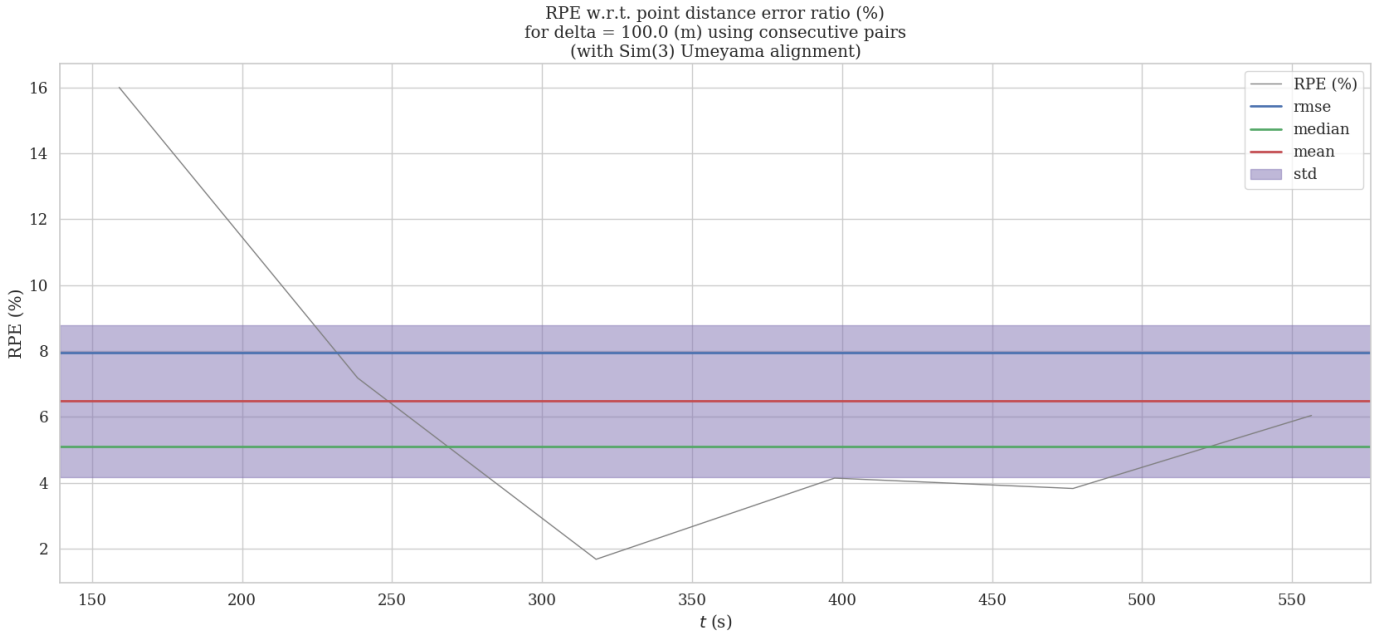

Figure 5.5: Mono Forest SLAM RPE on sequence 1008-03.

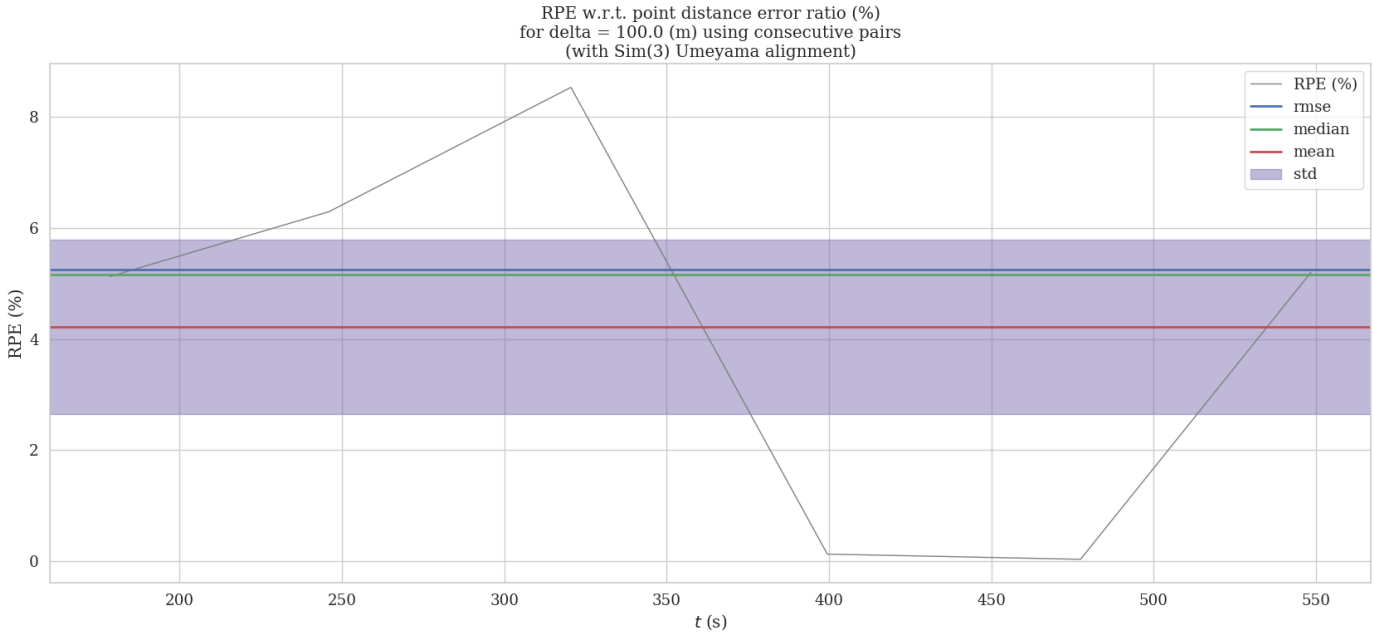

Figure 5.6: Stereo Forest SLAM RPE on sequence 1008-03.

Table 5.3: Median RPE as a percentage for each sequence in the BotanicGarden dataset for mono and stereo Forest SLAM.

Sequence	Monocular	Stereo
1005-00	27.797 %	**12.453 %**
1005-01	8.680 %	**2.680 %**
1005-07	19.221 %	**9.115 %**
1006-01	15.941 %	**0.770 %**
1008-03	**5.089 %**	5.165 %
1018-00	5.823 %	**0.823 %**
1018-13	4.991 %	**2.022 %**
Avg	12.506	**4.718**

Table 5.3 compares the median RPE for our mono and stereo Forest SLAM methods on each of the seven BotanicGarden dataset sequences. It is clear that the stereo method is superior to the mono approach with the stereo system achieving a lower RPE on all sequences except 1008-03 which is still very close. The average RPE for the stereo system is far lower than the mono method. In particular, the stereo system actually performed the best on the longest sequence, 1006-01, which means it is suitable for both short and long sequences. Only on sequences 1005-00 and 1005-07 does it fail significantly, which is possibly due to the complex nature of those trajectories, although it still outperforms the mono system on those sequences. On the shorter sequences, the stereo method achieves an RPE comparable with the SOTA systems in Table 5.2.

5.3.3 XYZ Errors and Estimated Velocities

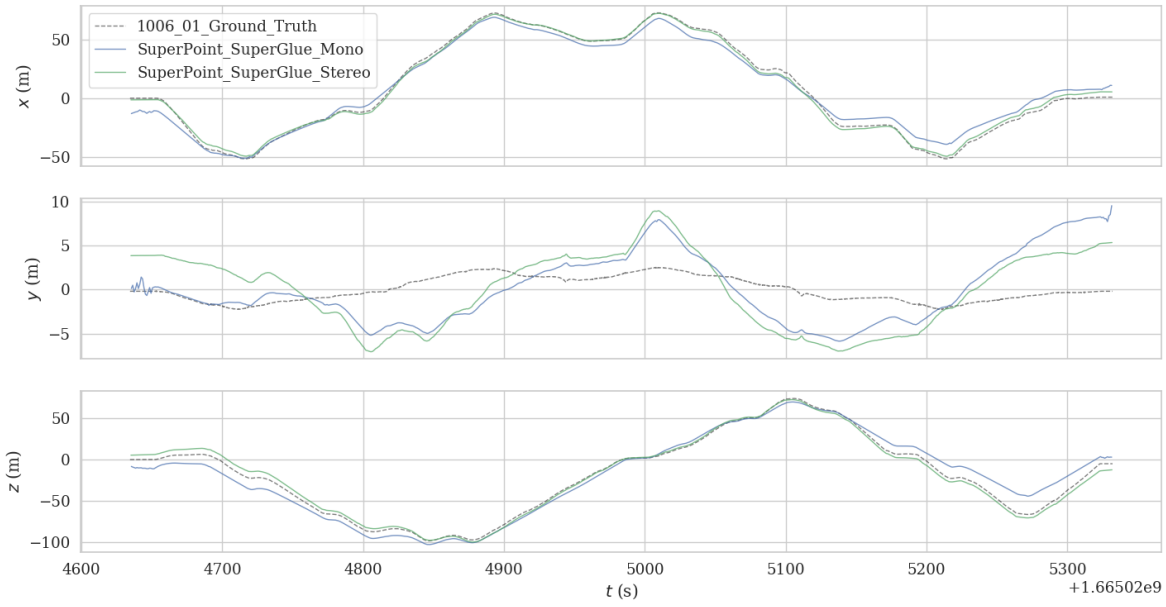

Figure 5.7: Estimated and ground truth trajectories for each axis on sequence 1006-01.

Figure 5.7 compares the estimated and ground truth trajectories generated by the mono and stereo methods for the x, y and z axes. The time is given in seconds and utilises the rosbag timings from the 1006-01 sequence. The Evo tool swaps the y and z axes so the x and z axes in Figure 5.7 refer to the 2D motion and the y axis represents the depth or vertical motion of the trajectory. As shown by the errors, the 2D motion is generally tracked quite well, with both mono and stereo methods closely following the ground truth, albeit the stereo method still performing better. However, both methods seem to struggle when estimating the vertical motion as shown by the large deviation from the ground truth in the y axis in Figure 5.7. Since the data in the BotanicGarden dataset was collected using a ground robot, we expect the trajectories to mainly be flat with small deviations in height based on the slope of the ground. This is reflected by the scales of the plots in which the 2D motion has much larger absolute differences between each pose. When performing pose estimation using the matched keypoints between frames, it is often difficult to reliably estimate small changes in height if there is little to no vertical motion between each frame. Nevertheless, we expect our approach to perform better when dealing with larger height differences such as scenarios involving aerial robots moving through 3D space.

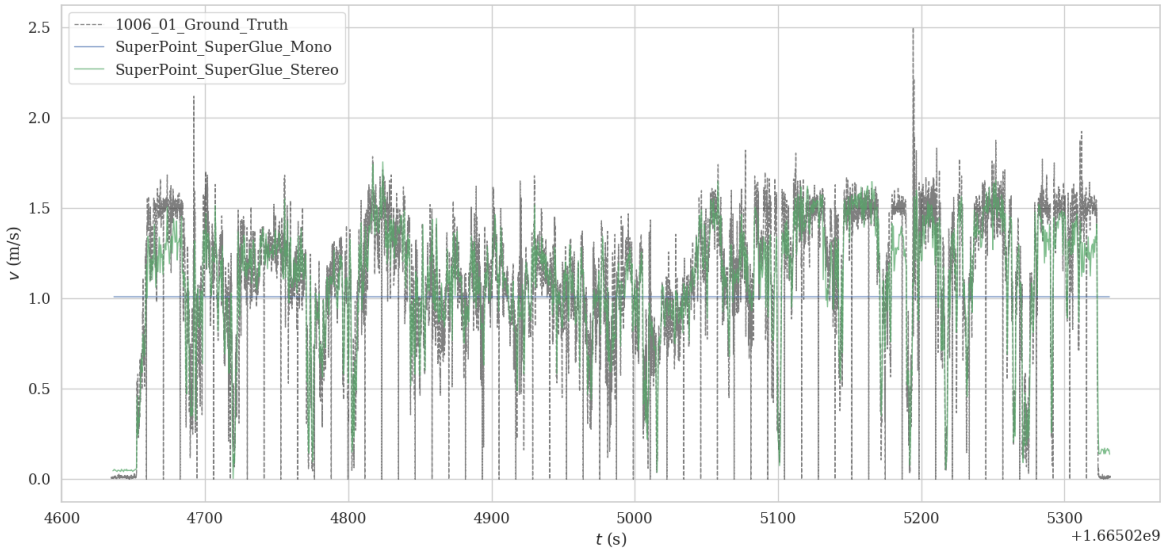

Figure 5.8: Estimated and ground truth velocities for each axis on sequence 1006-01.

Figure 5.8 shows the estimated and ground truth velocities for different parts of the 1006-01 recorded trajectory for both the mono and stereo methods. From the graph it is clear to see that the stereo approach is able to accurately estimate the speed that the camera is moving through the environment, with the estimated velocities closely aligning with the ground truth. The mono approach however, assumes a constant velocity throughout the entire sequence, indicating that it thinks the camera is moving at the same speed between every frame. This is seen in every sequence and is due to the fact that a single camera approach cannot recover the distance between the camera and objects in the scene to be able to estimate the speed.

5.3.4 Absolute Trajectory Error

The Absolute Trajectory Error (ATE) serves as a metric to evaluate the global consistency of the estimated trajectory with the ground truth. By default, the Evo tool calculates the Absolute Pose Error (APE) which takes into account both translation and rotation. However, we configure it to only use the translation part of the poses so that it is equivalent to the ATE which is the more common metric used when evaluating SLAM trajectories. We display each trajectory as a coloured heatmap with blue colours indicating smaller errors and red colours indicating larger errors. The stereo method performs much better overall, although both methods have significant errors at the ends of the trajectories as shown by the red colours.

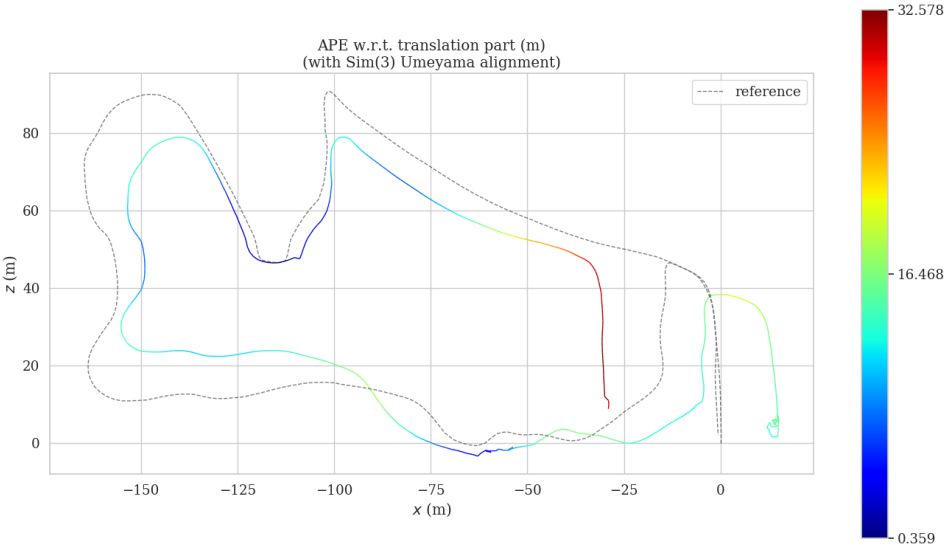

Figure 5.9: Mono Forest SLAM ATE heatmap on sequence 1005-00.

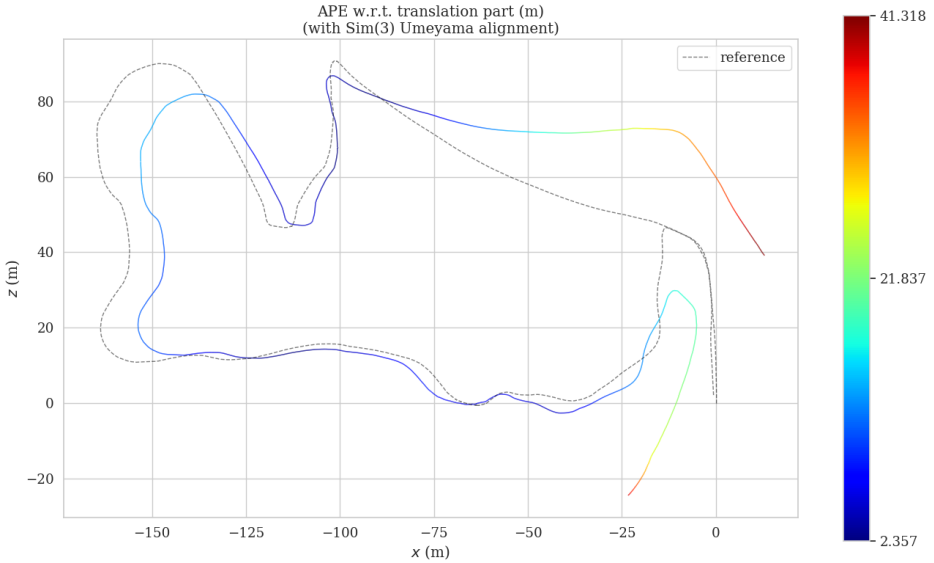

Figure 5.10: Stereo Forest SLAM ATE heatmap on sequence 1005-00.

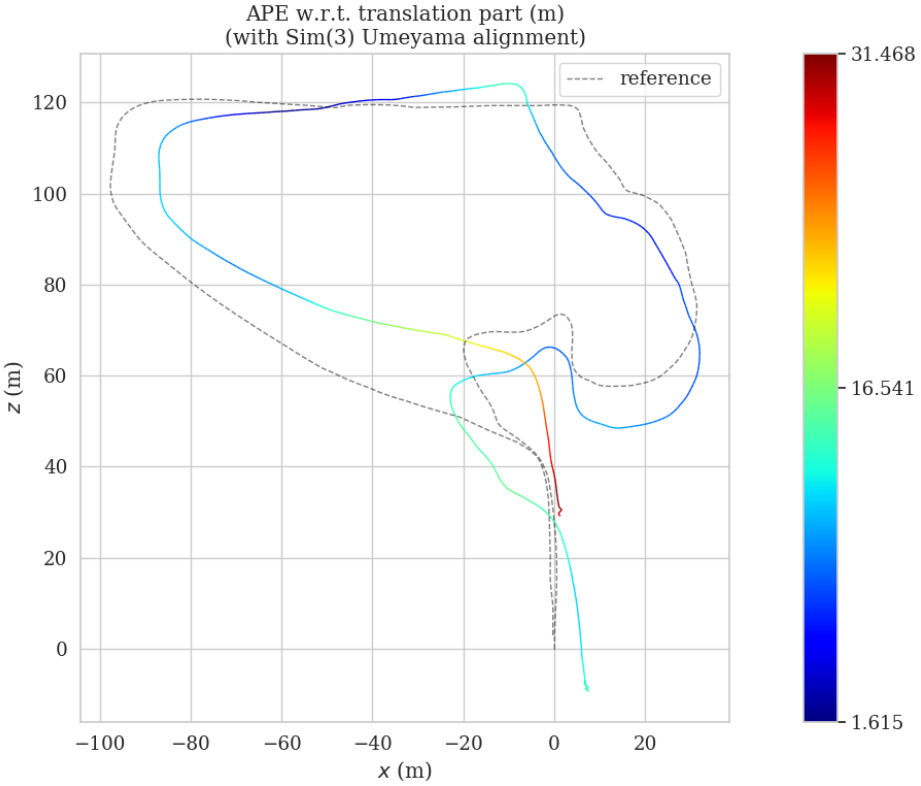

Figure 5.11: Mono Forest SLAM ATE heatmap on sequence 1005-01.

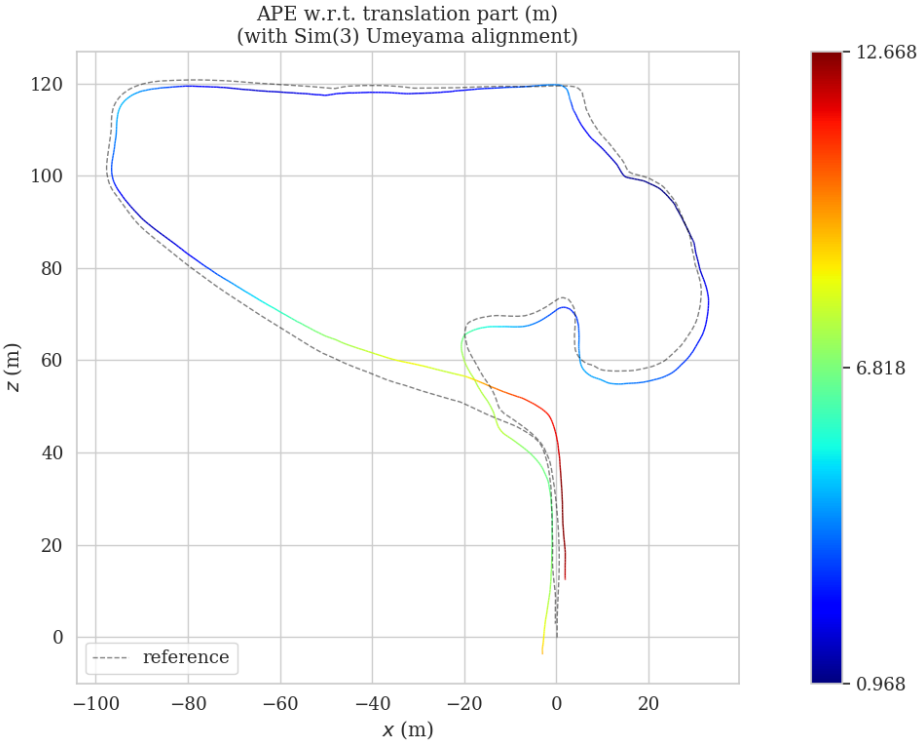

Figure 5.12: Stereo Forest SLAM ATE heatmap on sequence 1005-01.

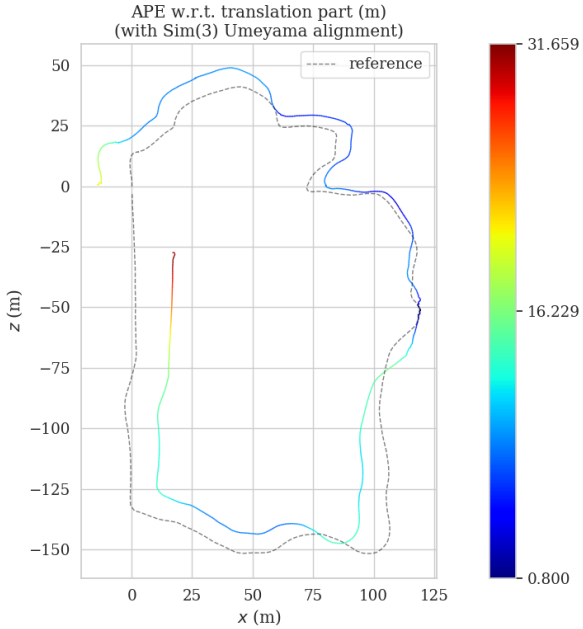

Figure 5.13: Mono Forest SLAM ATE heatmap on sequence 1005-07.

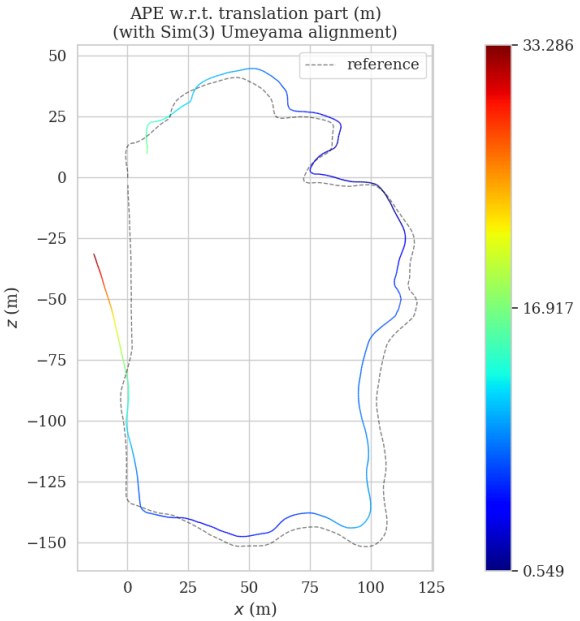

Figure 5.14: Stereo Forest SLAM ATE heatmap on sequence 1005-07.

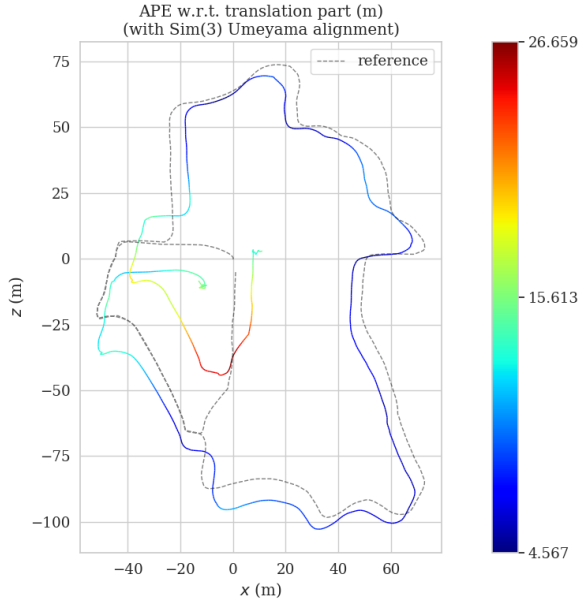

Figure 5.15: Mono Forest SLAM ATE heatmap on sequence 1006-01.

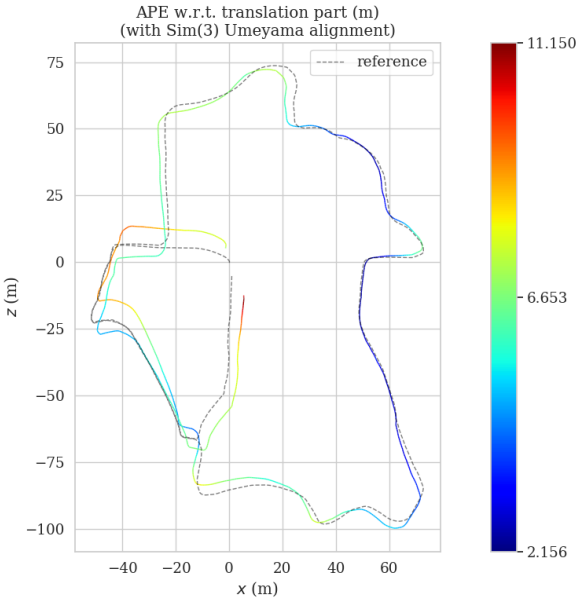

Figure 5.16: Stereo Forest SLAM ATE heatmap on sequence 1006-01.

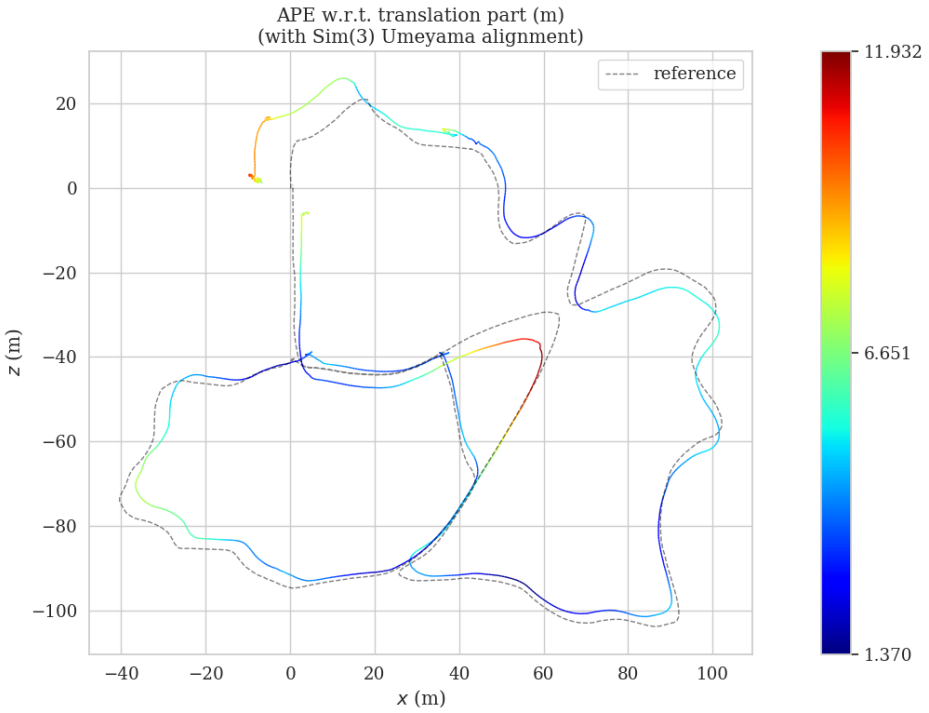

Figure 5.17: Mono Forest SLAM ATE heatmap on sequence 1008-03.

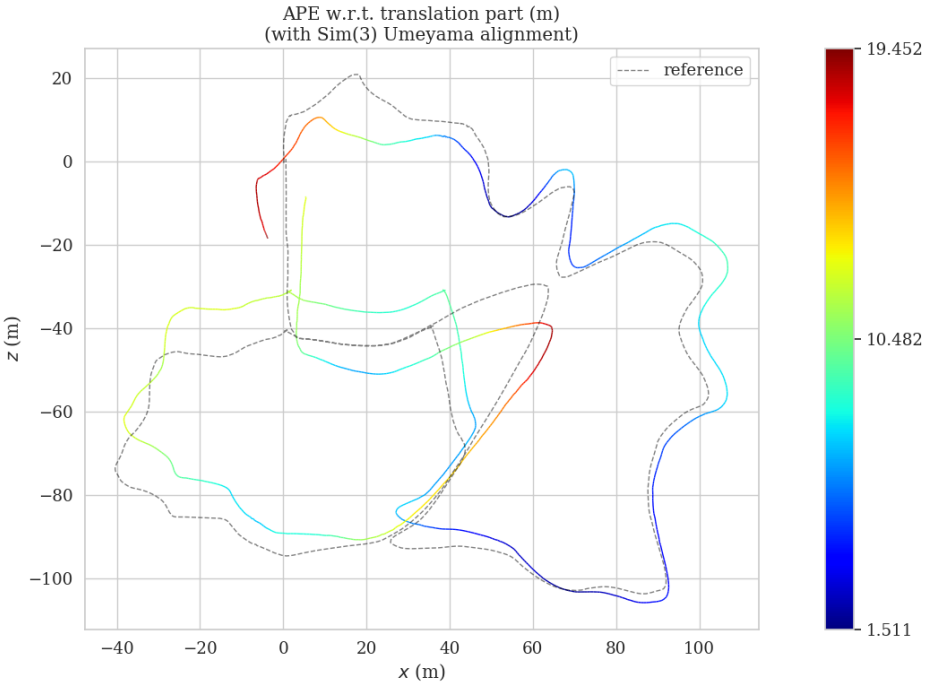

Figure 5.18: Stereo Forest SLAM ATE heatmap on sequence 1008-03.

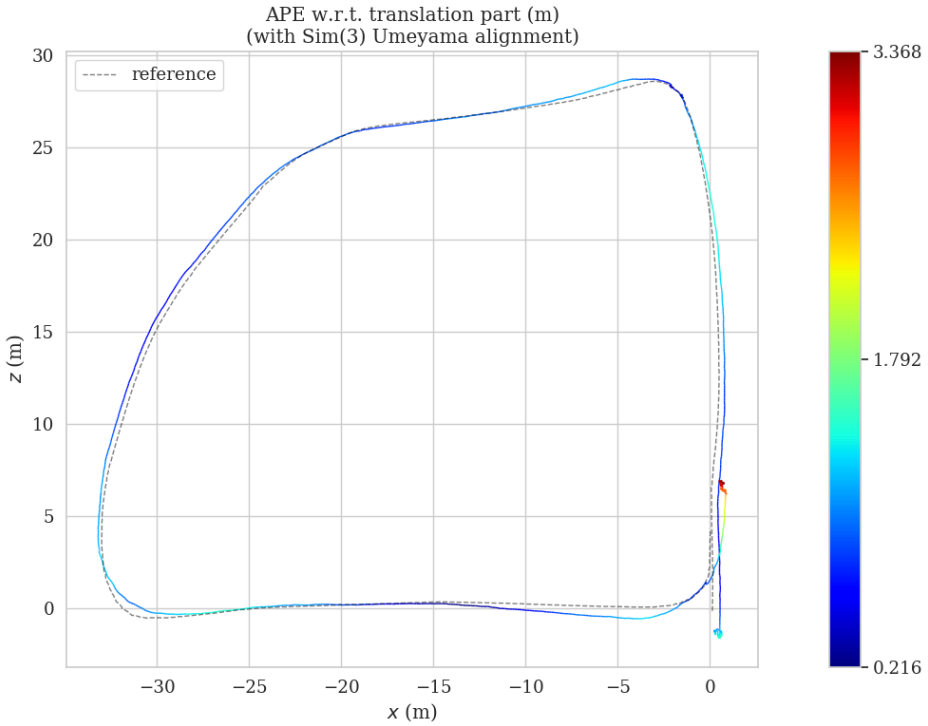

Figure 5.19: Mono Forest SLAM ATE heatmap on sequence 1018-00.

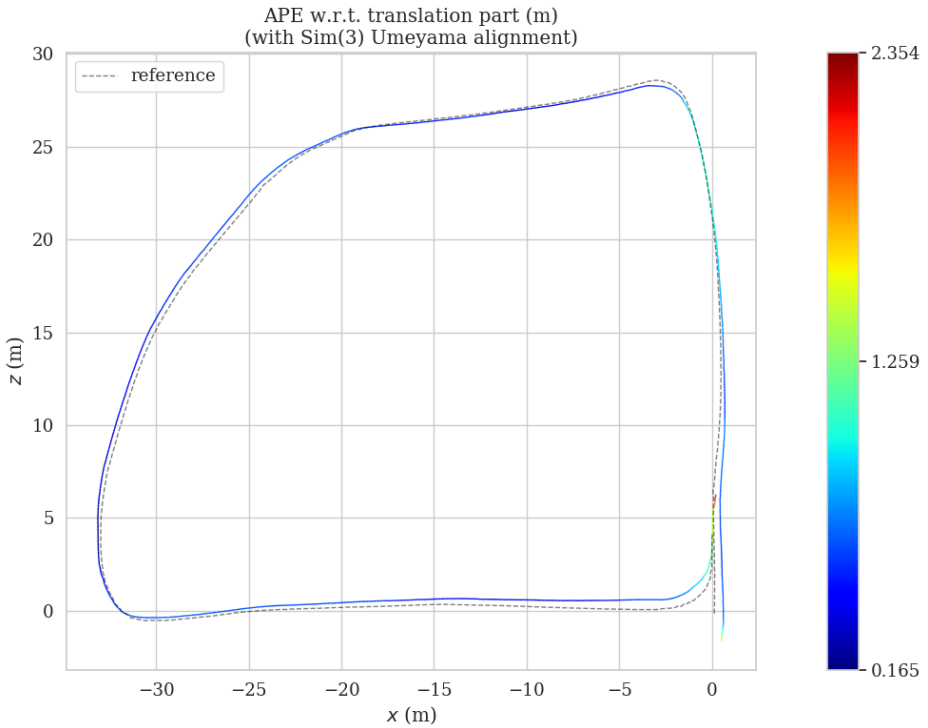

Figure 5.20: Stereo Forest SLAM ATE heatmap on sequence 1018-00.

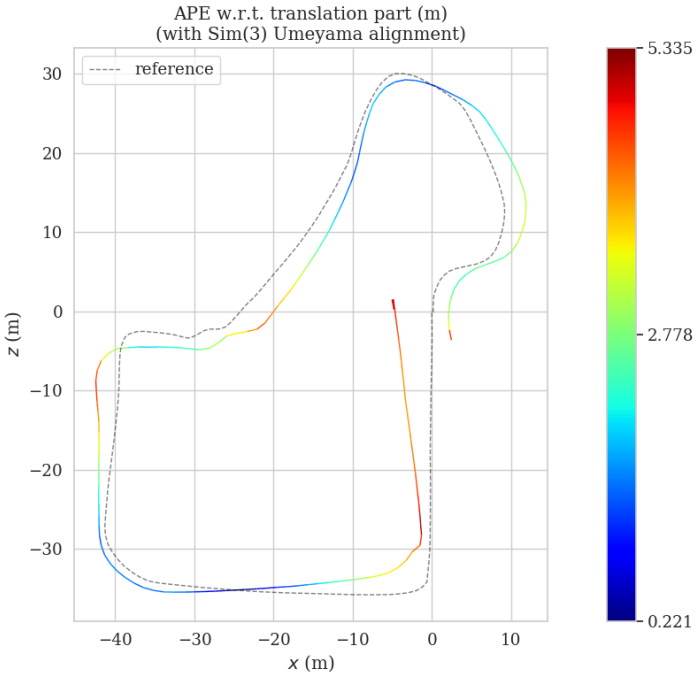

Figure 5.21: Mono Forest SLAM ATE heatmap on sequence 1018-13.

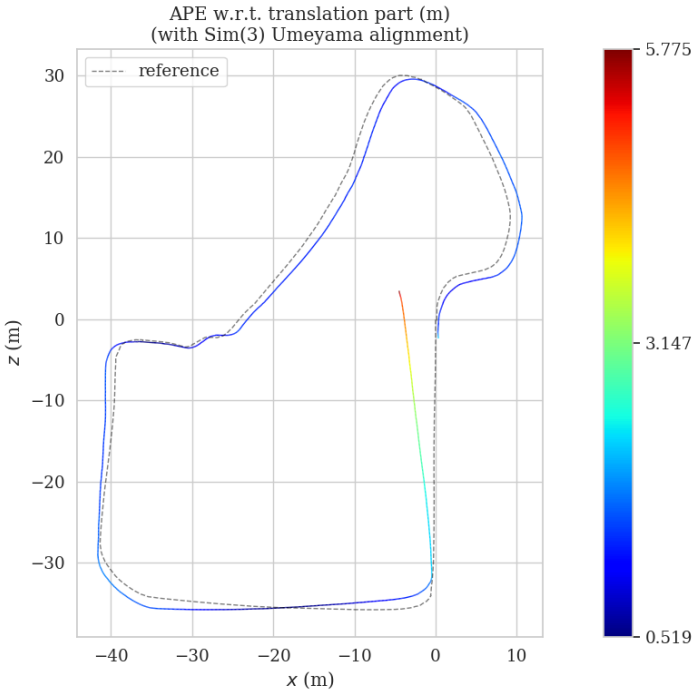

Figure 5.22: Stereo Forest SLAM ATE heatmap on sequence 1018-13.

5.4 Mapping

Although the main focus of our SLAM system is on robust relative pose estimations, our Forest SLAM system is still able to generate dense 3D point cloud maps. Since our mono approach is unable to estimate the depths of keypoints directly, we transform the LiDAR point clouds from the */velodyne_points* topic using the estimated poses to generate a 3D SLAM map. However, since we recognise that not all robotic systems have access to onboard LiDAR, our mapping results are focused on the stereo method, which is able to generate dense 3D point cloud maps without the use of additional sensors. This is due to the fact that the stereo method is able to directly estimate the depth of the matched keypoints in 3D space, allowing the points to be projected onto the map-frame using the estimated relative poses.

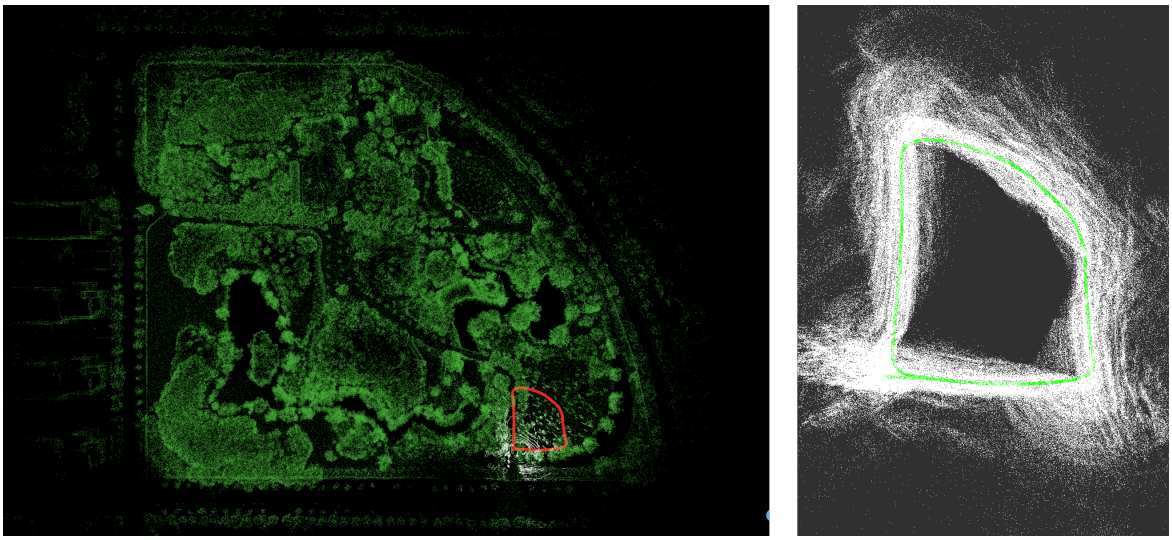

Figure 5.23: Ground truth and generated maps for sequence 1018-00.

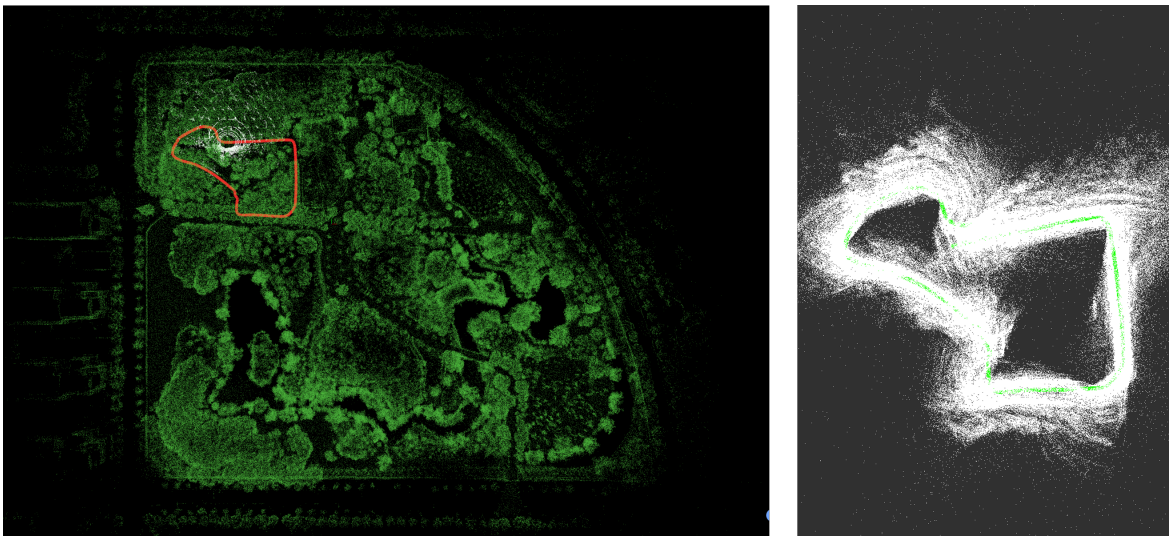

Figure 5.24: Ground truth and generated maps for sequence 1018-13.

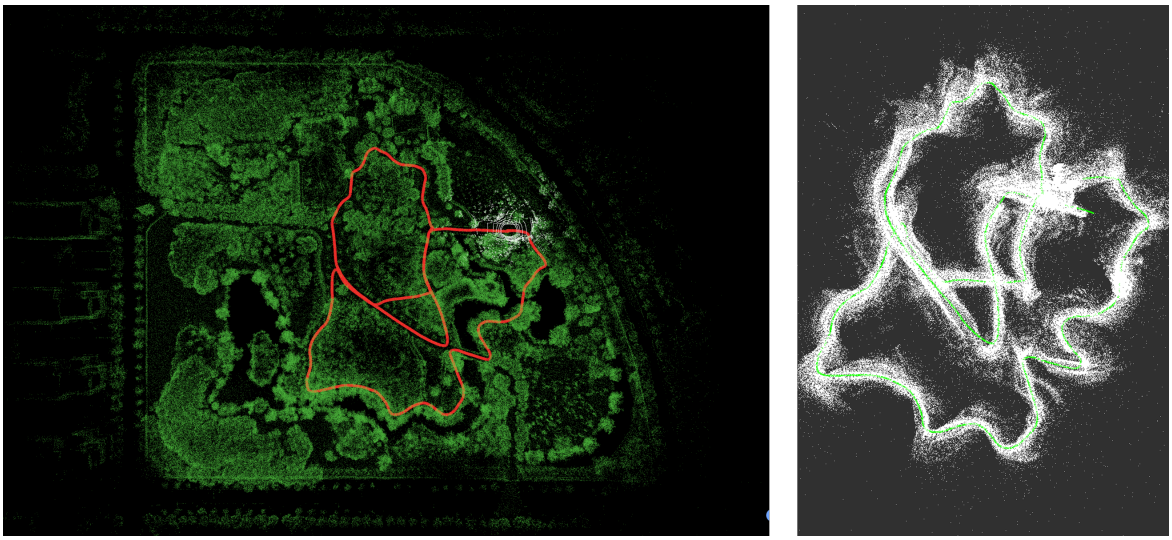

Figure 5.25: Ground truth and generated maps for sequence 1008-03.

We showcase the mapping results of our stereo system on sequences 1018-00, 1018-13 and 1008-03 as shown in Figures 5.23, 5.24 and 5.25 respectively. The ground truth maps are provided as images of the entire botanical garden which was surveyed using a Leica RTC360 industrial 3D laser scanner. This results in highly accurate ground truth maps with millimetre accuracy. The RTC360 has a very large scan radius of 130m and was moved to several different locations so that the entire botanical garden could be mapped. However, this makes it difficult to compare the mapping accuracy of our stereo Forest SLAM system, since our maps are restricted to the areas in which the cameras were able to view the scene in each recorded sequence.

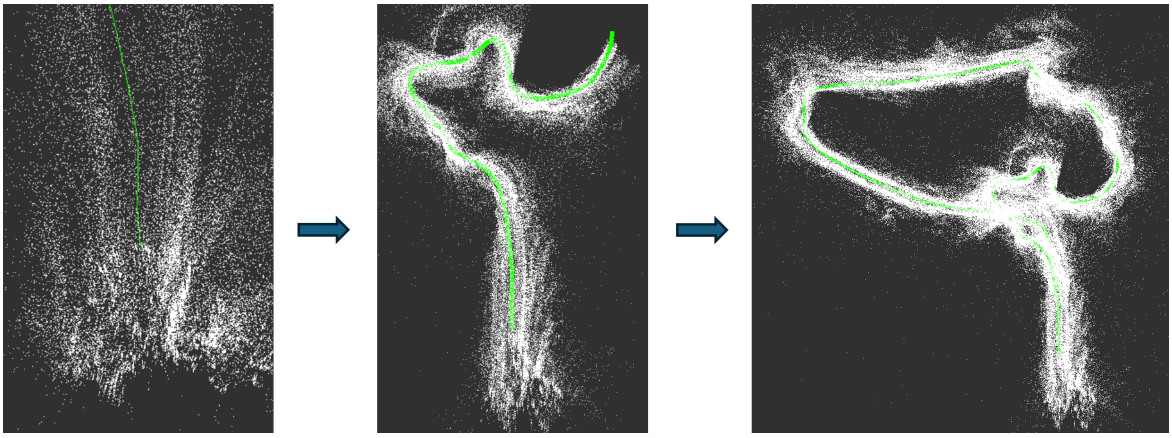

Figure 5.26: Mapping progression for sequence 1005-01.

Figure 5.26 showcases how the maps are incrementally generated, with the localisation and mapping tasks closely interlinked. At each iteration of the SLAM algorithm, the map is updated using the relative pose estimations to culminate in a global 3D point cloud SLAM map.

5.5 Ablation Study

To understand the importance of robust feature detection and correspondence within a SLAM system for forest environments, we perform an ablation study analysing the impact of different feature detectors and matchers. In this we utilise our stereo Forest SLAM system, substituting SuperPoint with ORB features and SuperGlue with a Brute Force (BF) matcher, comparing the pose estimation performance of each.

Although the BotanicGarden dataset contains unstructured natural scenes, the robot motion is relatively slow, resulting in little motion blur. To truly evaluate a SLAM system for dynamic forest scenes, we artificially add motion blur to each frame in sequence 1018-00. We randomly select pixels which make up 10% of the image before applying different levels of blur. This is carried out using a motion blur kernel K, which when convolved with the input image, results in the effect of motion blur in a specific direction. K is initialised as a diagonal matrix of ones to simulate motion blur along a particular direction, which cannot be achieved using traditional Gaussian blurring as it is non-directional. We create a rotation matrix M which is used to rotate the kernel so that blur can be achieved in different directions. However, in our implementation, we set the angle to zero, which results in purely horizontal motion blur. K is normalised to ensure the same level of brightness is used as in the original image. It is applied to the input image using the *cv2.filter2D()* function which performs the convolution and outputs the blurred image.

Figure 5.27: Motion blur levels for frames in sequence 1018-00.

Figure 5.27 shows the different levels of artificial motion blur being applied to the frames in sequence 1018-00. The left-most image is the original image containing no additional motion blur. A kernel size of 10 was used to achieve the motion blur in the centre image. The right-most image contains the result of using a kernel size of 20.

When performing this ablation study, we utilise the *cv2.ORB_create()* function to instantiate the ORB feature detector. To ensure a fair comparison, we use the optimal settings for the BF matcher when paired with the ORB feature detector. This involves setting the distance metric to *cv2.NORM_HAMMING* and ensuring *crossCheck=True*. As we are comparing different feature detectors and matchers on the stereo method, we ensure that the disparity map is calculated using the blurred images for all methods which are tested to maintain fair comparisons.

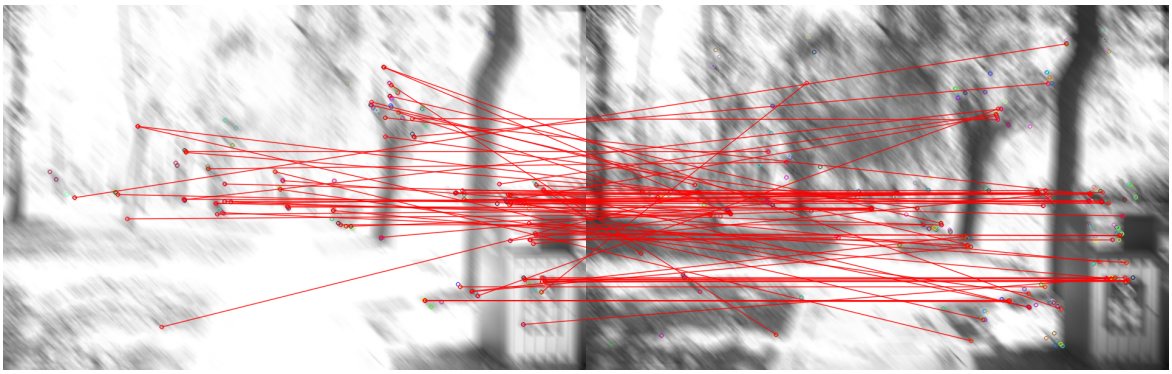

Figure 5.28: ORB feature matches with $K = 20$.

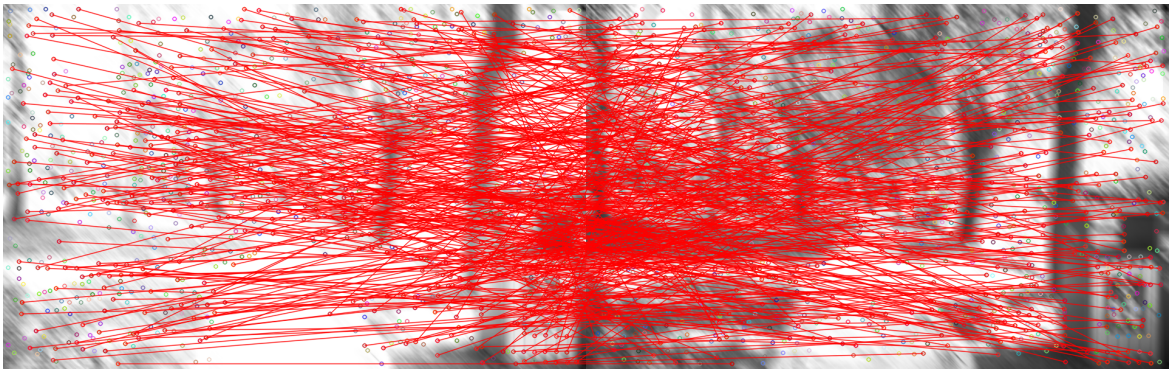

Figure 5.29: SuperGlue feature matches with $K = 20$.

Figure 5.28 shows the ORB feature matches on a pair of frames from sequence 1018-00. This particular set of images provides a particularly challenging scenario for feature correspondence. The dynamic lighting conditions result in overexposed areas of the image where there are a lack of well defined points that can be identified as features. Additional motion blur is also added using a kernel size of 20 to create very difficult conditions for feature matching. As a result, there are very few ORB features that are detected, with most being found on the edges of the trees. With only a few feature matches there is limited information to inform the pose estimation task in a SLAM system.

Figure 5.29 shows the SuperPoint features being matched by SuperGlue on the same scene. It is apparent that there are far more features being detected and matched. The SuperPoint features are spread out across the entire frame unlike the ORB features, which are concentrated in a small area. The learning-based methods are clearly more robust to the changes in illumination as well, given that features are being detected and matched even in the overexposed areas of the image. Although the added motion blur represents an extreme case, it highlights the need for robust feature correspondence methods in forest environments since the scenes can change vastly from one frame to the next. SuperPoint and SuperGlue are much more robust to the challenges faced in forest environments compared to the traditional ORB and BF approach. We further validate this by testing in the context of Forest SLAM.

Table 5.4: Median RPE as a percentage for ORB+BF and SuperPoint+SuperGlue stereo Forest SLAM for different levels of motion blur on sequence 1018-00.

Blur Level (Kernel Size)	ORB+BF	SuperPoint+SuperGlue
0	3.304 %	**0.823 %**
10	17.984 %	**1.740 %**
20	58.302 %	**1.741 %**

Table 5.4 compares the RPE for the traditional ORB and BF method with the SuperPoint and SuperGlue learning based approach. It is clear that the learning based method is far more robust when dealing with challenging dynamic scenes, with the SuperPoint and SuperGlue combination achieving a lower RPE than the traditional counterpart for every blur level tested. The learning based approach is still able to result in accurate relative pose estimations even when large amounts of artificial motion blur are added, with the increase in RPE being quite small. On the other hand, the traditional method results in extremely large increases in RPE, indicating that the feature detection and correspondence have led to incorrect pose estimations. It is especially interesting to note that even the worse performance of the SuperPoint and SuperGlue combination still outperforms the best performance of the ORB and BF approach.

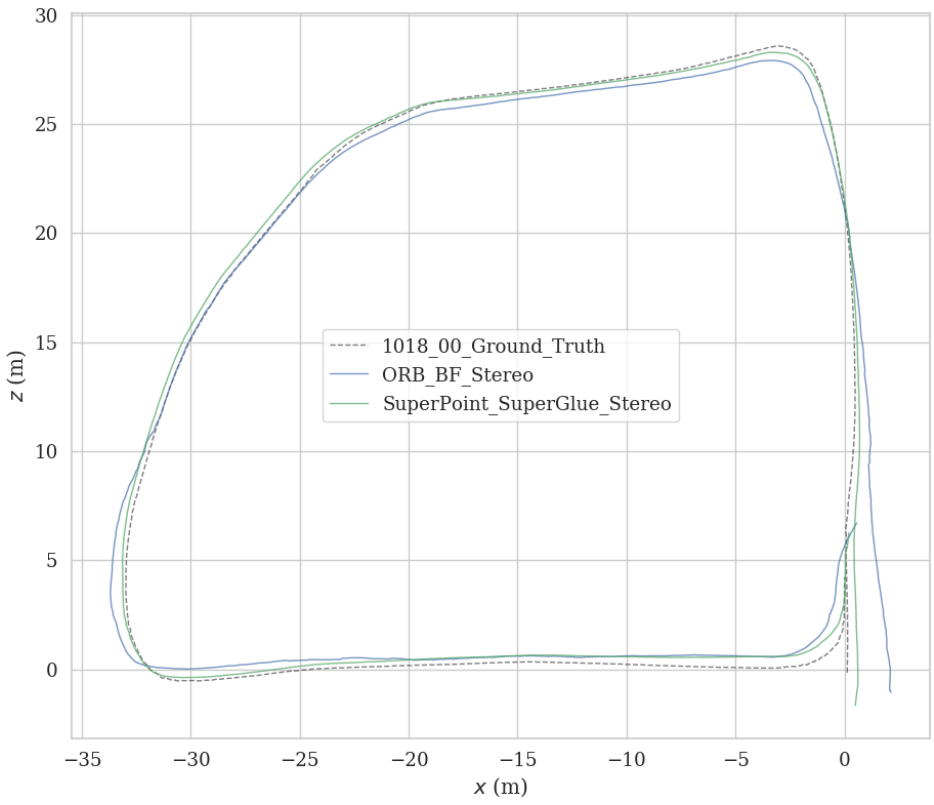

Figure 5.30: ORB+BF and SuperPoint+SuperGlue trajectories on sequence 1018-00 with no additional motion blur.

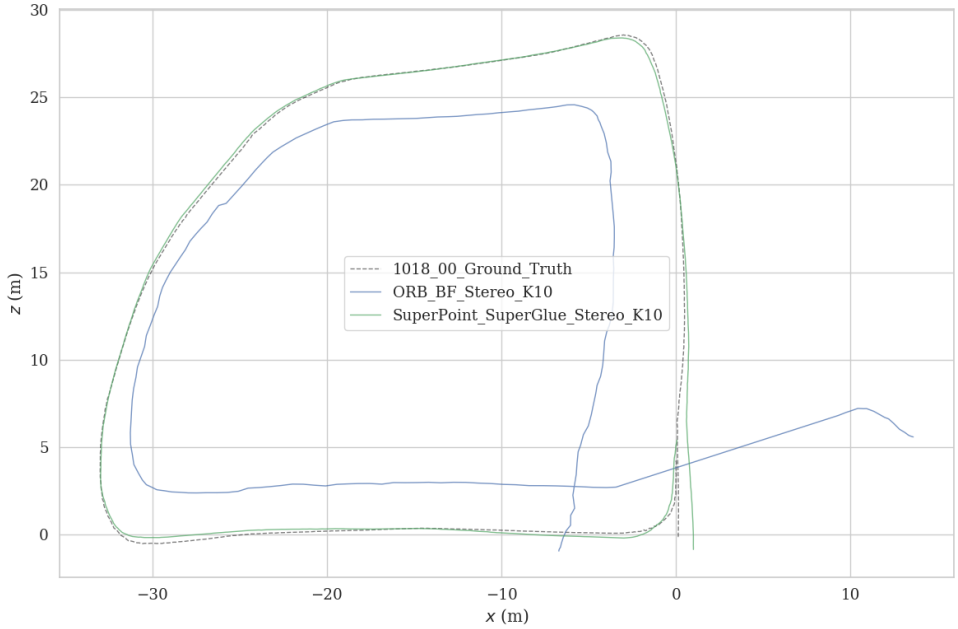

Figure 5.31: ORB+BF and SuperPoint+SuperGlue trajectories on sequence 1018-00 with $K = 10$.

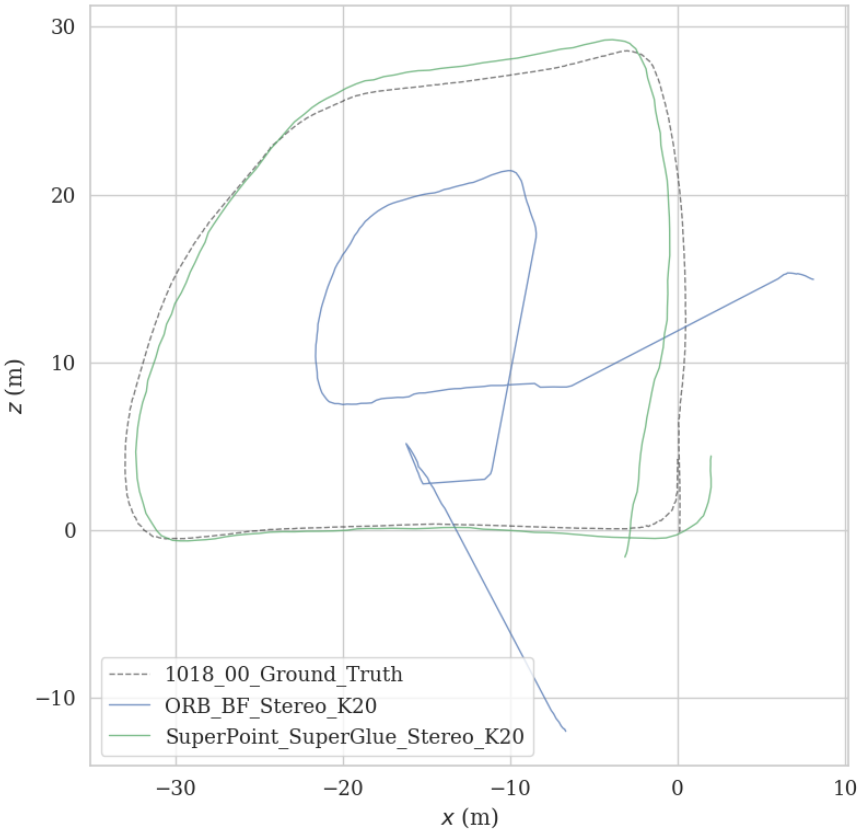

Figure 5.32: ORB+BF and SuperPoint+SuperGlue trajectories on sequence 1018-00 with $K = 20$.

Figure 5.30 shows the generated trajectories for the stereo Forest SLAM system incorporating the different feature detection and correspondence methods on sequence 1018-00 when no additional motion blur is introduced. Both methods achieve estimated trajectories that closely follow the ground truth, although the SuperPoint and SuperGlue combination has less deviations in its path. When motion blur is introduced through kernel sizes of 10 and 20, the ORB and BF approach breaks down, with the trajectories failing significantly as shown in Figures 5.31 and 5.32 respectively. Meanwhile, the learning based feature detector and matcher approach results in useable trajectories for all three levels of motion blur. Even when a kernel size of 20 is used, there are only minor deviations from the ground truth trajectory, as shown in Figure 5.32. This illustrates why the SuperPoint feature detector and SuperGlue feature correspondence technique are robust methods to inform the relative pose estimation aspect of a V-SLAM system.

5.6 Conclusion

In this report we have investigated the effectiveness of the state-of-the-art SuperPoint feature detector and SuperGlue feature matcher as part of a V-SLAM framework for forest environments. We have demonstrated the performance of both our monocular and stereo Forest SLAM methods on real-world unstructured natural scenes from the BotanicGarden dataset, showcasing the superior nature of stereo approaches in V-SLAM. Our stereo method achieves state-of-the-art relative pose estimation accuracy with an average RPE of 4.718%, outperforming the stereo visual-only ORB-SLAM3 system.

From the results obtained, it is clear that robust feature detection and correspondence methods are required for accurate relative pose estimations in the context of a V-SLAM system for forest environments. We demonstrate the superior performance of learning-based approaches compared to their traditional counterparts through an ablation study analysing the impact of dynamic scenes on pose estimation accuracy. SuperPoint and SuperGlue are robust techniques that demonstrate the need for learning-based feature detectors and matchers in SLAM systems.

Nevertheless, we acknowledge that there are improvements that can be made to ensure Forest SLAM is a highly robust V-SLAM system for forest environments. Although our method achieves highly accurate relative pose estimations as shown by its RPE scores, the overall trajectories can be improved through the inclusion of global optimisation techniques such as bundle adjustment as well as loop closure detection. This would help to counteract any odometry drift accumulated which is especially important for longer sequences. Even though the pre-trained SuperGlue outdoor model outperformed the traditional methods when evaluating the pose estimation accuracy on the BotanicGarden dataset, it is limited in its capabilities due to it mainly being trained on urban outdoor scenes. There is the possibility that a feature detector and feature matcher that are specifically trained on forest scenes would perform better when incorporated into a V-SLAM system for forest environments.

Chapter 6

Discussion

6.1 Future Work

We recognise that domain specific SLAM is an emerging field, so new SLAM datasets targeted specifically at challenging forest environments would provide a useful way to benchmark the performance of current state-of-the-art SLAM systems on forest scenes. These would also provide relevant examples to re-train the SuperPoint and SuperGlue models on forest specific data which could improve pose estimations.

Recently, Huber et al. from the Technical University of Munich have explored a new learning-based feature correspondence method called DynamicGlue [76]. It is similar to SuperGlue, but trained specifically on dynamic scenes. Re-training DynamicGlue on forest scenes could result in even better relative pose estimations when incorporated into Forest SLAM.

Deep learning SLAM models are now being explored and have shown to outperform their geometric counterparts in certain cases, but are typically trained on general SLAM datasets. Additionally, these models are usually trained to perform V-SLAM tasks in an end-to-end manner, directly regressing the relative camera pose from the input images. However, with a robust pre-trained feature detector and feature correspondence model, there is the option of training a forest specific deep learning SLAM model to perform pose estimations using the matched keypoints directly.

6.2 Ethical Considerations

There are several ethical issues which need to be considered regarding the potential use cases for such technology. A forest-based SLAM approach could be used for autonomous drone navigation in military applications. It could also be used in surveillance applications where targets are tracked without their knowledge which could be a breach of privacy. However, the intention is that this technology is to be used solely for civilian applications in order to help humans carry out labour intensive tasks such as forest monitoring, or dangerous tasks such as search and rescue missions in order to minimise the risk exposed to humans.

Bibliography

[1] Barbara Koch, Matthias Dees, Jo Van Brusselen, H Leblon, and R Nilsson. Forestry applications. In *Advances in Photogrammetry, Remote Sensing and Spatial Information Sciences: 2008 ISPRS Congress Book*, pages 439–465. July 2008. ISBN 978-0-415-47805-2. doi: 10.1201/9780203888445.ch32. Journal Abbreviation: Advances in Photogrammetry, Remote Sensing and Spatial Information Sciences: 2008 ISPRS Congress Book. pages 1

[2] S. Karma, E. Zorba, G. C. Pallis, G. Statheropoulos, I. Balta, K. Mikedi, J. Vamvakari, A. Pappa, M. Chalaris, G. Xanthopoulos, and M. Statheropoulos. Use of unmanned vehicles in search and rescue operations in forest fires: Advantages and limitations observed in a field trial. *International Journal of Disaster Risk Reduction*, 13:307–312, September 2015. ISSN 2212-4209. doi: 10.1016/j.ijdrr.2015.07.009. URL https://www.sciencedirect.com/science/article/pii/S2212420915300364. pages 1

[3] Alex Souza Bastos and Hisashi Hasegawa. Behavior of GPS Signal Interruption Probability under Tree Canopies in Different Forest Conditions. *European Journal of Remote Sensing*, 46(1):613–622, January 2013. ISSN 2279-7254. doi: 10.5721/EuJRS20134636. URL https://www.tandfonline.com/doi/full/10.5721/EuJRS20134636. pages 1

[4] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006. ISSN 1558-223X. doi: 10.1109/MRA.2006.1638022. URL https://ieeexplore.ieee.org/document/1638022. Conference Name: IEEE Robotics & Automation Magazine. pages 1, 3

[5] Weifeng Chen, Guangtao Shang, Aihong Ji, Chengjun Zhou, Xiyang Wang, Chonghui Xu, Zhenxiong Li, and Kai Hu. An Overview on Visual SLAM: From Tradition to Semantic. *Remote Sensing*, 14(13):3010, January 2022. ISSN 2072-4292. doi: 10.3390/rs14133010. URL https://www.mdpi.com/2072-4292/14/13/3010. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute. pages 1

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, Providence, RI, June

2012. IEEE. ISBN 978-1-4673-1228-8 978-1-4673-1226-4 978-1-4673-1227-1. doi: 10.1109/CVPR.2012.6248074. URL http://ieeexplore.ieee.org/document/6248074/. pages 1

[7] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, September 2016. ISSN 0278-3649. doi: 10.1177/0278364915620033. URL https://doi.org/10.1177/0278364915620033. Publisher: SAGE Publications Ltd STM. pages 1

[8] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, October 2012. doi: 10.1109/IROS.2012.6385773. URL https://ieeexplore.ieee.org/document/6385773. ISSN: 2153-0866. pages 1

[9] Riccardo Giubilato, Wolfgang Sturzl, Armin Wedler, and Rudolph Triebel. Challenges of SLAM in Extremely Unstructured Environments: The DLR Planetary Stereo, Solid-State LiDAR, Inertial Dataset. *IEEE Robotics and Automation Letters*, 7(4):8721–8728, October 2022. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2022.3188118. URL https://ieeexplore.ieee.org/document/9813579/. pages 1

[10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 337–33712, Salt Lake City, UT, USA, June 2018. IEEE. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00060. URL https://ieeexplore.ieee.org/document/8575521/. pages 2, 15

[11] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning Feature Matching With Graph Neural Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4937–4946, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.00499. URL https://ieeexplore.ieee.org/document/9157489/. pages 2, 15

[12] Yuanzhi Liu, Yujia Fu, Minghui Qin, Yufeng Xu, Baoxin Xu, Fengdong Chen, Bart Goossens, Poly Z.H. Sun, Hongwei Yu, Chun Liu, Long Chen, Wei Tao, and Hui Zhao. BotanicGarden: A High-Quality Dataset for Robot Navigation in Unstructured Natural Environments. *IEEE Robotics and Automation Letters*, 9(3):2798–2805, March 2024. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2024.3359548. URL https://ieeexplore.ieee.org/document/10415477/. pages 2, 20, 21

[13] ROS: Home, . URL https://ros.org/. pages 2, 21

[14] 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 Multi-Map SLAM. *IEEE Transactions on Robotics*, 37(6): 1874–1890, December 2021. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2021.3075644. URL http://arxiv.org/abs/2007.11898. arXiv:2007.11898 [cs]. pages 2, 3, 17, 18

[15] 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, August 2018. ISSN 1941-0468. doi: 10.1109/TRO.2018.2853729. URL https://ieeexplore.ieee.org/document/8421746. Conference Name: IEEE Transactions on Robotics. pages 3, 18, 19

[16] Fabio T. Ramos, Juan Nieto, and Hugh F. Durrant-Whyte. Recognising and Modelling Landmarks to Close Loops in Outdoor SLAM. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2036–2041, April 2007. doi: 10.1109/ROBOT.2007.363621. URL https://ieeexplore.ieee.org/document/4209385. ISSN: 1050-4729. pages 3

[17] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 1 edition, July 2017. ISBN 978-1-107-15939-6 978-1-316-67152-8. doi: 10.1017/9781316671528. URL https://www.cambridge.org/core/product/identifier/9781316671528/type/book. pages 3

[18] Iman Abaspur Kazerouni, Luke Fitzgerald, Gerard Dooly, and Daniel Toal. A survey of state-of-the-art on visual SLAM. *Expert Systems with Applications*, 205:117734, November 2022. ISSN 0957-4174. doi: 10.1016/j.eswa.2022.117734. URL https://www.sciencedirect.com/science/article/pii/S0957417422010156. pages 4

[19] Fernando Molano Ortiz, Matteo Sammarco, Luís Henrique M. K. Costa, and Marcin Detyniecki. Applications and Services Using Vehicular Exteroceptive Sensors: A Survey. *IEEE Transactions on Intelligent Vehicles*, 8(1):949–969, January 2023. ISSN 2379-8904. doi: 10.1109/TIV.2022.3182218. URL https://ieeexplore.ieee.org/document/9795135. Conference Name: IEEE Transactions on Intelligent Vehicles. pages 4

[20] Ilham Faisal, Tito Purboyo, and Anton Ansori. A Review of Accelerometer Sensor and Gyroscope Sensor in IMU Sensors on Motion Capture. *Journal of Engineering and Applied Sciences*, 15:826–829, November 2019. doi: 10.36478/jeasci.2020.826.829. pages 4

[21] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. 2020. pages 4, 20

[22] Wu Zhou, E Shiju, Zhenxin Cao, and Ying Dong. Review of SLAM Data Association Study. 2016. URL https://www.atlantis-press.com/article/25859609.pdf. pages 5

[23] Mordechai Ben-Ari and Francesco Mondada. Robotic Motion and Odometry. pages 63–93. January 2018. ISBN 978-3-319-62532-4. doi: 10.1007/978-3-319-62533-1_5. pages 5

[24] Viorela Ila, Josep M. Porta, and Juan Andrade-Cetto. Information-Based Compact Pose SLAM. *IEEE Transactions on Robotics*, 26(1):78–93, February 2010. ISSN 1941-0468. doi: 10.1109/TRO.2009.2034435. URL https://ieeexplore.ieee.org/abstract/document/5325904. Conference Name: IEEE Transactions on Robotics. pages 5

[25] Andrii Kudriashov, Tomasz Buratowski, Mariusz Giergiel, and Piotr Małka. SLAM as Probabilistic Robotics Framework Approach. In Andrii Kudriashov, Tomasz Buratowski, Mariusz Giergiel, and Piotr Małka, editors, *SLAM Techniques Application for Mobile Robot in Rough Terrain*, Mechanisms and Machine Science, pages 39–64. Springer International Publishing, Cham, 2020. ISBN 978-3-030-48981-6. doi: 10.1007/978-3-030-48981-6_3. URL https://doi.org/10.1007/978-3-030-48981-6_3. pages 5

[26] Aritra Mukherjee, Satyaki Chakraborty, and Sanjoy Kumar Saha. Detection of loop closure in SLAM: A DeconvNet based approach. *Applied Soft Computing*, 80:650–656, July 2019. ISSN 1568-4946. doi: 10.1016/j.asoc.2019.04.041. URL https://www.sciencedirect.com/science/article/pii/S1568494619302339. pages 5

[27] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. In *Advances in Neural Information Processing Systems*, volume 34, pages 16558–16569. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/89fcd07f20b6785b92134bd6c1d0fa42-Abstract.html. pages 5

[28] P Sankalprajan, Thrilochan Sharma, Hamsa Datta Perur, and Prithvi Sekhar Pagala. Comparative analysis of ROS based 2D and 3D SLAM algorithms for Autonomous Ground Vehicles. In *2020 International Conference for Emerging Technology (INCET)*, pages 1–6, June 2020. doi: 10.1109/INCET49848.2020.9154101. URL https://ieeexplore.ieee.org/document/9154101. pages 5

[29] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems*, 1(4):289–311, December 2015. ISSN 2199-854X. doi: 10.1007/s40903-015-0032-7. URL https://doi.org/10.1007/s40903-015-0032-7. pages 5

[30] Rana Azzam, Tarek Taha, Shoudong Huang, and Yahya Zweiri. Feature-based visual simultaneous localization and mapping: a survey. *SN Applied Sciences*, 2(2):224, February 2020. ISSN 2523-3963, 2523-3971. doi: 10.1007/s42452-020-2001-3. URL http://link.springer.com/10.1007/s42452-020-2001-3. pages 6

[31] Ali Tourani, Hriday Bavle, Jose Luis Sanchez-Lopez, and Holger Voos. Visual SLAM: What Are the Current Trends and What to Expect? *Sensors (Basel, Switzerland)*, 22(23):9297, November 2022. ISSN 1424-8220. doi: 10.3390/s22239297. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9735432/. pages 6

[32] Zhengdong Zhang, Amr Suleiman, Luca Carlone, Vivienne Sze, and Sertac Karaman. Visual-Inertial Odometry on Chip: An Algorithm-and-Hardware Co-design Approach. In *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, July 2017. ISBN 978-0-9923747-3-0. doi: 10.15607/RSS.2017.XIII.028. URL http://www.roboticsproceedings.org/rss13/p28.pdf. pages 6

[33] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. Kalman Filter and Its Application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 74–77, November 2015. doi: 10.1109/ICINIS.2015.35. URL https://ieeexplore.ieee.org/document/7528889. pages 6

[34] Simon J Julier and Jeffrey K Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. 1997. URL https://www.cs.unc.edu/~welch/kalman/media/pdf/Julier1997_SPIE_KF.pdf. pages 6

[35] Georges Younes, Daniel Asmar, Elie Shammas, and John Zelek. Keyframe-based monocular SLAM: design, survey, and future directions. *Robotics and Autonomous Systems*, 98:67–88, December 2017. ISSN 0921-8890. doi: 10.1016/j.robot.2017.09.010. URL https://www.sciencedirect.com/science/article/pii/S0921889017300647. pages 6

[36] Nigel Joseph Bandeira Dias, Gustavo Teodoro Laureano, and Ronaldo Martins Da Costa. Keyframe Selection for Visual Localization and Mapping Tasks: A Systematic Literature Review. *Robotics*, 12(3):88, June 2023. ISSN 2218-6581. doi: 10.3390/robotics12030088. URL https://www.mdpi.com/2218-6581/12/3/88. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute. pages 7

[37] Deep Learning for Computer Vision with Caffe and cuDNN, October 2014. URL https://developer.nvidia.com/blog/deep-learning-computer-vision-caffe-cudnn/. pages 7

[38] Zewen Xu, Zheng Rong, and Yihong Wu. A survey: which features are required for dynamic visual simultaneous localization and mapping? *Visual Computing for Industry, Biomedicine, and Art*, 4(1):20, July 2021. ISSN 2524-4442. doi: 10.1186/s42492-021-00086-w. URL https://doi.org/10.1186/s42492-021-00086-w. pages 7

[39] Ambroise Moreau, Matei Mancas, and Thierry Dutoit. Depth prediction from 2D images: A taxonomy and an evaluation study. *Image and Vision Computing*, 93:103825, January 2020. ISSN 0262-8856. doi: 10.1016/j.imavis.

2019.11.003. URL https://www.sciencedirect.com/science/article/pii/S0262885619304184. pages 7

[40] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment — A Modern Synthesis. In Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883, pages 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. ISBN 978-3-540-67973-8 978-3-540-44480-0. doi: 10.1007/3-540-44480-7_21. URL https://link.springer.com/10.1007/3-540-44480-7_21. Series Title: Lecture Notes in Computer Science. pages 8

[41] Xiwu Zhang, Yan Su, and Xinhua Zhu. Loop closure detection for visual SLAM systems using convolutional neural network. In *2017 23rd International Conference on Automation and Computing (ICAC)*, pages 1–6, September 2017. doi: 10.23919/IConAC.2017.8082072. URL https://ieeexplore.ieee.org/document/8082072. pages 8

[42] Wen Liu, Shuo Wang, Zhongliang Deng, and Hong Chen. A Review of Image Feature Descriptors in Visual Positioning. 2021. pages 8

[43] datahacker.rs. OpenCV #013 Harris Corner Detector - Theory, July 2019. URL https://datahacker.rs/opencv-harris-corner-detector-part1/. pages 8

[44] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of the Alvey Vision Conference 1988*, pages 23.1–23.6, Manchester, 1988. Alvey Vision Club. doi: 10.5244/C.2.23. URL http://www.bmva.org/bmvc/1988/avc-88-023.html. pages 8

[45] OpenCV: Harris Corner Detection, . URL https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html. pages 9

[46] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://link.springer.com/10.1023/B:VISI.0000029664.99615.94. pages 10

[47] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157 vol.2, Kerkyra, Greece, 1999. IEEE. ISBN 978-0-7695-0164-2. doi: 10.1109/ICCV.1999.790410. URL http://ieeexplore.ieee.org/document/790410/. pages 10

[48] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, Lecture Notes in Computer Science, pages 404–417, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-33833-8. doi: 10.1007/11744023_32. pages 10

[49] Yuanxin Ye, Lorenzo Bruzzone, Jie Shan, Francesca Bovolo, and Qing Zhu. *A Fast and Robust Matching Framework for Multimodal Remote Sensing Image Registration*. August 2018. pages 10

[50] David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314, pages 778–792. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-15560-4 978-3-642-15561-1. doi: 10.1007/978-3-642-15561-1_56. URL http://link.springer.com/10.1007/978-3-642-15561-1_56. Series Title: Lecture Notes in Computer Science. pages 11

[51] Mohammad Norouzi, David J Fleet, and Ruslan Salakhutdinov. Hamming Distance Metric Learning. pages 11

[52] Edward Rosten and Tom Drummond. Machine Learning for High-Speed Corner Detection. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, Lecture Notes in Computer Science, pages 430–443, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-33833-8. doi: 10.1007/11744023_34. pages 11

[53] Francesco Secci and Andrea Ceccarelli. On failures of RGB cameras and their effects in autonomous driving applications. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 13–24, October 2020. doi: 10.1109/ISSRE5003.2020.00011. URL https://ieeexplore.ieee.org/document/9251080. ISSN: 2332-6549. pages 12

[54] Karen De Pauw, Pieter Sanczuk, Camille Meeussen, Leen Depauw, Emiel De Lombaerde, Sanne Govaert, Thomas Vanneste, Jörg Brunet, Sara A. O. Cousins, Cristina Gasperini, Per-Ola Hedwall, Giovanni Iacopetti, Jonathan Lenoir, Jan Plue, Federico Selvi, Fabien Spicher, Jaime Uria-Diez, Kris Verheyen, Pieter Vangansbeke, and Pieter De Frenne. Forest understorey communities respond strongly to light in interaction with forest structure, but not to microclimate warming. *New Phytologist*, 233 (1):219–235, 2022. ISSN 1469-8137. doi: 10.1111/nph.17803. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/nph.17803. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/nph.17803. pages 12

[55] Omar Elezabi, Sebastien Guesney-Bodet, and Jean-Baptiste Thomas. Impact of Exposure and Illumination on Texture Classification Based on Raw Spectral Filter Array Images. *Sensors (Basel, Switzerland)*, 23(12):5443, June 2023. ISSN 1424-8220. doi: 10.3390/s23125443. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10302469/. pages 12

[56] Tim Brooks and Jonathan T. Barron. Learning to Synthesize Motion Blur. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6833–6841, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00700. URL https://ieeexplore.ieee.org/document/8953368/. pages 13

[57] Basil Benny, Bahadir Kocer, and Ronald Clark. Tracking and Mapping Forest Environments with Tree Trunk Parameter Estimations. September 2023. pages 13, 14, 15, 16, 17

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459/. pages 13

[59] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks, April 2017. URL http://arxiv.org/abs/1611.05431. arXiv:1611.05431 [cs]. pages 13

[60] Vincent Grondin, Jean-Michel Fortin, François Pomerleau, and Philippe Giguère. Tree detection and diameter estimation based on deep learning. *Forestry: An International Journal of Forest Research*, 96(2):264–276, April 2023. ISSN 0015-752X. doi: 10.1093/forestry/cpac043. URL https://doi.org/10.1093/forestry/cpac043. pages 13

[61] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011. doi: 10.1109/ICCV.2011.6126544. URL https://ieeexplore.ieee.org/document/6126544. ISSN: 2380-7504. pages 14

[62] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. 1999. URL https://www.vldb.org/conf/1999/P49.pdf. pages 14

[63] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL https://dl.acm.org/doi/10.1145/358669.358692. pages 17, 27

[64] Juan Lagos, Urho Lempiö, and Esa Rahtu. FinnWoodlands Dataset. In Rikke Gade, Michael Felsberg, and Joni-Kristian Kämäräinen, editors, *Image Analysis*, Lecture Notes in Computer Science, pages 95–110, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-31435-3. doi: 10.1007/978-3-031-31435-3_7. pages 20

[65] Ihtisham Ali, Ahmed Durmush, Olli Suominen, Jari Yli-Hietanen, Sari Peltonen, Jussi Collin, and Atanas Gotchev. FinnForest dataset: A forest landscape for visual SLAM. *Robotics and Autonomous Systems*, 132:103610, October 2020. ISSN 0921-8890. doi: 10.1016/j.robot.2020.103610. URL https://www.sciencedirect.com/science/article/pii/S0921889020304504. pages 20

[66] magicleap/SuperGluePretrainedNetwork, 2020. URL https://github.com/magicleap/SuperGluePretrainedNetwork. original-date: 2020-03-17T19:32:12Z. pages 23

[67] OpenCV: Camera Calibration and 3D Reconstruction, . URL https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html. pages 25, 26

[68] PyTorch, . URL https://pytorch.org/. pages 26

[69] R.I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997. ISSN 01628828. doi: 10.1109/34.601246. URL http://ieeexplore.ieee.org/document/601246/. pages 26

[70] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004. ISSN 0162-8828. doi: 10.1109/TPAMI.2004.17. URL http://ieeexplore.ieee.org/document/1288525/. pages 26

[71] Essential Matrix, Fundamental Matrix, . URL https://velog.io/@sunbei00/Essential-Matrix-Fundamental-Matrix. pages 27

[72] ros-visualization/rviz, June 2024. URL https://github.com/ros-visualization/rviz. original-date: 2012-09-18T18:15:03Z. pages 28

[73] Sanja Fidler. CSC420 Lecture 12: Depth from Stereo. 2015. URL https://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf. pages 29, 30

[74] Henri P Gavin. The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems. 2024. URL https://people.duke.edu/~hpgavin/lm.pdf. pages 31

[75] Michael Grupp. evo: Python package for the evaluation of odometry and SLAM., 2017. URL https://github.com/MichaelGrupp/evo. original-date: 2017-09-13T19:40:00Z. pages 33

[76] Theresa Huber, Simon Schaefer, and Stefan Leutenegger. DynamicGlue: Epipolar and Time-Informed Data Association in Dynamic Environments using Graph Neural Networks, March 2024. URL https://arxiv.org/abs/2403.11370v2. pages 53