Visual-lidar Map Alignment for Change Detection in Infrastructure Applications

by

Jake McLaughlin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2023

Abstract

Current human-based infrastructure inspections face several problems. Firstly, human inspectors face safety risks working in hazardous environments and reaching challenging areas in order to inspect an infrastructure asset fully. Secondly, they are time-consuming and costly, requiring significant resources to perform. Lastly, human inspections are subjective and prone to human error, which causes inconsistent assessments. These challenges limit engineers abilities to make accurate assessments of infrastructure health and can make infrastructure management difficult and costly.

Current research in Simultaneous Localization and Mapping (SLAM) has shown promise when applied to automated infrastructure inspections due to its ability to provide accurate 3D maps of infrastructure assets; allowing objective and efficient inspection of assets using the generated 3D models. For the purpose of inspecting infrastructure assets, repeat visits and monitoring changes to the structure are paramount to determining the health of the asset. When generating a 3D map in GPS denied environments (as is typically the case for many infrastructure assets such as bridges), it is typically with respect to the first sensor measurement, meaning multiple scans of the same area will not be in the same reference frame. Many publicly available SLAM solutions do not provide the ability to automatically align 3D scans of the same area from mapping sessions gathered across time (multi-session mapping). When performing repeat inspections of the same asset, data needs to be merged and integrated seamlessly into a single reference frame for accurate association of regions of interest over time.

The central contributions of this thesis are: the implementation of a generic map alignment algorithm utilizing both visual and lidar data for enhanced robustness, and the collection of an infrastructure centric dataset for repeat inspections. These contributions greatly enhance the usefulness of existing automated infrastructure inspection pipelines by enabling the tracking of changes in infrastructure assets over time, allowing for accurate estimation of the degradation infrastructure assets. These contributions also have a significant impact on the general field of SLAM, by decoupling map alignment from SLAM, researchers have more freedom to explore new SLAM approaches without the necessitation of using approaches that incorporate multi-session capability.

To achieve accurate and robust 3D map alignment, this work assumes that an initial trajectory of each scan has been generated by SLAM and that camera and lidar data is available from the scanning session. The map alignment process utilizes techniques from Visual and Lidar relocalization to align maps in a way that is robust to environment change. The map alignment approach was successfully validated on the custom datasets gathered in the Structures Lab at the University of Waterloo and at the Conestogo Bridge in Kitchener, Ontario. The map alignment algorithm proposed here demonstrates an average increase in alignment accuracy of 15.6% in the Structures Lab dataset and 72% in the Conestogo Bridge dataset when compared to the traditional naive alignment approach. Additionally, the results show that utilizing both visual and lidar for place recognition improves accuracy by an average of 14% when compared to using just one sensor modality.

Acknowledgements

I would like to thank my supervisor, Dr. Sriram Narasimhan, for the support and guidance in my pursuit of meaningful research.

I would also like to thank my fellow lab members for supporting each other and fostering a community of continual improvement and a high standard of excellence. A special thanks goes out to Nicholas Charron for pushing me to fulfill my potential as a researcher.

Dedication

This is dedicated to my parents, Keith and Beth, without whom I would not be able to pursue my passion.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Infrastructure maintenance is an integral part of any advanced society, responsible for the upkeep and safety of physical structures ranging from roadways and bridges to pipelines and power grids. In recent years, the burgeoning field of robotics and automation has presented a compelling case for revolutionizing the traditional, labor-intensive manual inspection processes. This thesis proposes to extend the scope of these automated inspections by exploiting offline 3D map alignment techniques using visual and Light Detection and Ranging (lidar) sensor data.

1.1 Automated Infrastructure Inspection

The adoption of automation in infrastructure inspections has resulted in a profound shift towards precision and accuracy. A fundamental technology enabling this shift is 3D mapping, which involves the creation of a digital twin of the physical world. Analyzing the 3D maps generated from this process with rich visual information can identify structural anomalies or deterioration, empowering timely preventive measures and repairs. However, to accurately assess deterioration, accurate association across time is necessary. Most current 3D mapping packages do not provide the ability to align results into single reference frames (in the absence of GPS), making data association difficult. This thesis proposes a solution to this problem by augmenting existing automated inspection pipelines with offline map alignment. Most automated inspection pipelines follow 3 broad steps: 1) Data collection, 2) 3D Mapping and 3) Automated inspection.

1.1.1 Data Collection

3D scanning devices are equipped with various sensors, cameras, and instruments to gather data about the infrastructure being inspected. This may include high-resolution imagery, lidar scans, thermal imaging, and more. Either a human controlled robot equipped with these sensors or a human inspector is deployed to navigate and scan an infrastructure asset, capturing detailed information from different angles and perspectives. This approach to 3D scanning combines the accuracy of state-of-the-art SLAM algorithms, with the intuition of a human operator.

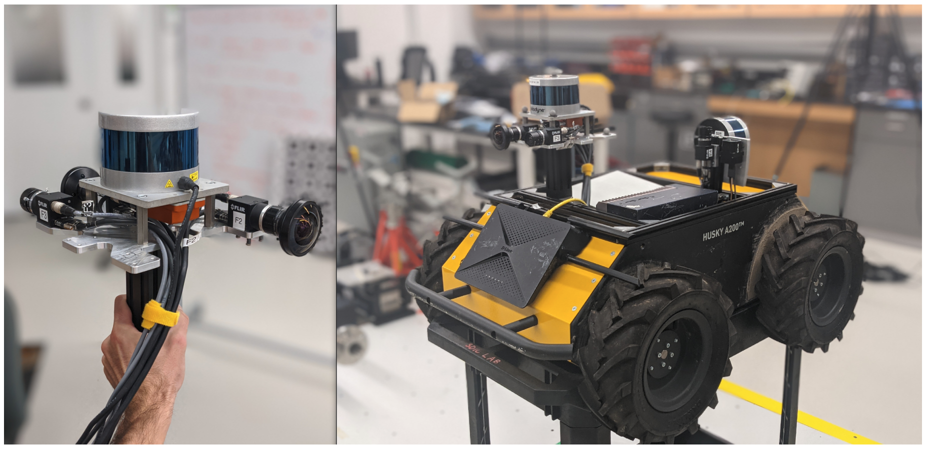


Figure 1.1: Infrastructure scanning data collection platforms. (Photos by: Nicholas Charron)

1.1.2 3D Mapping

Utilizing high-resolution cameras, lidar sensors, and advanced algorithms, an inspector can automatically scan complex environments, capturing precise and detailed 3D models of infrastructural assets. 3D Mapping algorithms generally fall into 3 categories: lidar based [53], visual based [10], or a combination of both [54], each with pros and cons for different environments and situations. Typically inspectors would have to be within arms reach of a defect to perform a proper inspection of an infrastructure asset, by utilizing precisely built 3D maps using long range sensors to perform inspection [**?**] an inspector can gather data from a distance (or remotely with a robot) for safe inspection afterwards.

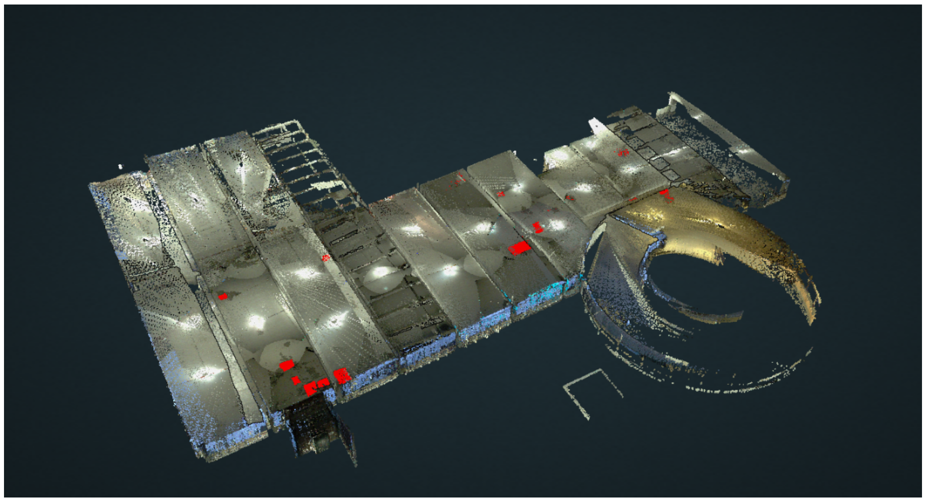


Figure 1.2: Concrete defect labelled point cloud map (defects in red).

1.1.3 Inspection

Upon acquisition of an accurate 3D model of the environment, various inspection operations can be automated. These operations can use either conventional knowledge-based algorithms or learning-based methodologies. Such inspection procedures encompass the automatic segmentation of structural primitives, such as girders, columns, and walls, to enabling measurements of their dimensions. More detailed inspection tasks involve the automatic detection and quantification of concrete defects (spalls, delaminations, cracks, etc.). This typically involves the identification of defects within image data, followed by their subsequent annotation within the 3D model, enabling their visualization and quantification. Other inspection tasks that require temporal alignment of scans encompass broad change detection, reporting of concrete defect growth, and monitoring of alterations in structural dimensions.

1.2 Motivation

Despite the advancements in 3D mapping and automated inspection, most current 3D mapping algorithms do not implement robust methods of aligning multiple sessions into a unified reference frame. This is particularly relevant for inspections, as their nature necessitates repeated data collection over the lifetime of the asset. Multi-session mapping, which involves aligning and comparing 3D maps from different inspection sessions, can provide quantitative insights into the gradual changes or defects in the infrastructure. It is, therefore important to investigate multi-session mapping for automated infrastructure inspection purposes.

Those 3D mapping solutions that do implement map alignment are fraught with another challenge: the dependence on the front-end Simultaneous Localization and Mapping (SLAM) method, meaning to use their map alignment, their SLAM system must also be used. A more flexible approach would be to decouple the offline map alignment process from the specific SLAM method used, which will increase the overall system's versatility. To address these challenges, this thesis will first identify existing literature on multi-session mapping, then identify the current limitations of their application to infrastructure inspections, and will propose an algorithm tailored for inspection tasks. The proposed algorithm takes advantage of visual and lidar relocalization methods for robust offline 3D map alignment. By exploiting both visual and lidar data, the proposed solution aims to automate the map alignment process that is agnostic to the SLAM technique employed. The primary end goal from an application standpoint is to facilitate accurate reporting of changes in the infrastructure health between subsequent scans.

Chapter 2

Background

2.1 3D Transformations

In metric map representations in robotics, the environment and the robot itself are represented mathematically using coordinates in 3D space. For instance, the position of the robot or an object could be represented as a 3D point (x, y, z), and the orientation of the robot or an object could be represented as a rotation about the three axes (roll, pitch, yaw), also known as Euler angles, a 3x3 rotation matrix, or as a quaternion (cite).

A crucial aspect of 3D geometry is the concept of coordinate transformations. A coordinate transformation changes the basis of the coordinate system, allowing us to represent points or vectors in one coordinate system (the "source" system) in terms of another coordinate system (the "target" system). Coordinate transformations are essential to transform data recorded from one sensor into the coordinate frame of another on the same robot. For instance, a lidar on a robot will report its data as a set of 3D points in the lidar's frame (the optical center); once the transformation between the lidar and a camera on the robot are known, this data can be transformed into the camera's reference frame.

To represent and compute coordinate transformations in 3D space, 4x4 transformation matrices are often used (cite). A 4x4 transformation matrix can represent both rotation and translation in a single matrix. To use this representation homogeneous coordinates are utilized to operate on 3D points and vectors. Homogeneous coordinates represent n dimensional points as a vector of size n+1, with a 1 in the last position.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.1}$$

Here, $\mathbf{R}$ is a 3×3 rotation matrix that represents the rotation part of the transformation, and $\mathbf{t}$ is a 3×1 vector that represents the translation. The 0 is a 1x3 matrix of zeros, and $\mathbf{I}$ is the identity. Given a 3-vector $\mathbf{P^a}$ representing a point $\mathbf{a}$, and a 4x4 matrix $\mathbf{T_M}$ representing the coordinate frame of $\mathbf{M}$, we define $\mathbf{P^a_M}$ as the vector starting from the origin of the frame $\mathbf{M}$ and stretching to point $\mathbf{P_a}$. The process of this transformation is given in Eq. 2.2.

$$\begin{aligned}\mathbf{P^a_M} &= \mathbf{T_M} \times \mathbf{\hat{P^a}} \\ &= \begin{bmatrix}\mathbf{R_M} & \mathbf{t_M} \\ \mathbf{0} & 1\end{bmatrix} \times \begin{bmatrix}\mathbf{P^a} \\ 1\end{bmatrix}\end{aligned} \tag{2.2}$$

Where $\mathbf{\hat{P^a}}$ are the homogeneous coordinates of $\mathbf{P^a}$. This transformation is equivalent to performing the rotation and translation as separate computation directly on the Cartesian coordinates of $\mathbf{P^a}$, shown in Eq. 2.3.

$$\mathbf{P^a_M} = \mathbf{R_M} \times \mathbf{P^a} + \mathbf{t_M} \tag{2.3}$$

It is important to note that transformations are not commutative, meaning the order in which they are applied matters, however, they are associative. The associative property allows for efficient "chaining" of transformations. Equation 2.4 shows the associative property of transformation matrices.

$$\begin{aligned}\mathbf{P^a_C} &= \mathbf{T^C_B} \times (\mathbf{T^B_A} \times \mathbf{P^a_A}) \\ &= (\mathbf{T^C_B} \times \mathbf{T^B_A}) \times \mathbf{P^a_A} \\ &= \mathbf{T^C_A} \times \mathbf{P^a_A}\end{aligned} \tag{2.4}$$

Figure 2.1 illustrates three coordinate frames with relative transforms between each other. It also shows a landmark (yellow star) as seen in each reference frames as $\mathbf{P^a_F}$, where $\mathbf{F}$ is the specific reference frame. It can also be seen that the landmark in one frame can be transformed into another using the relative transforms between frames. Additionally, it can be seen that to transform $\mathbf{P^a_A}$ into C, can be done with the single transform $\mathbf{T^C_A}$, as was demonstrated in Equation 2.4.

Note that a 3D transformation matrix is made up of a rotation matrix, $\mathbf{R_M}$, and a translation vector. $\mathbf{R_M}$ belongs to the Special Orthogonal group $\mathbf{SO(3)}$, while $\mathbf{T_M}$

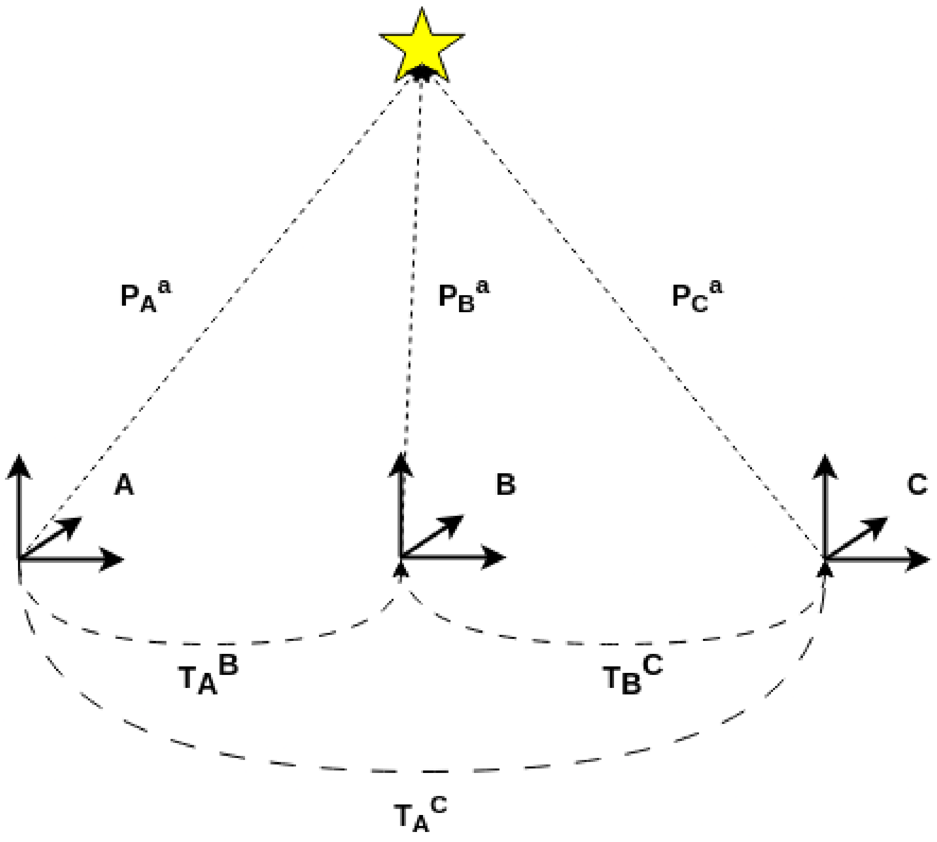
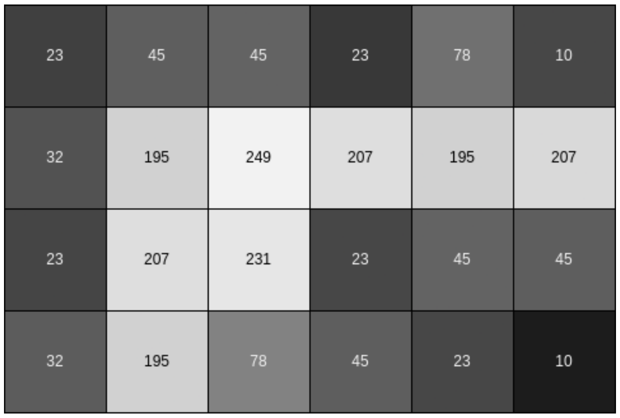


Figure 2.1: Diagram of two 3-dimensional coordinate frames.

belongs to the Special Euclidean group **SE(3)**. The collection of rotation matrices and transformation matrices doesn't form a vector space. Instead, they create matrix Lie groups requiring special algebra. While the mathematics underpinning matrix Lie groups isn't discussed here, more details can be found in texts such as [3].

2.2 Visual Sensors

Visual sensors (cameras) play a crucial role in perceiving the environment for autonomous robots. Understanding how the data they collect can be processed to extract useful information about the environment and how to relate the 2D measurements (pixels), as shown in Figure 2.2, they gather to the 3D world is important in understanding their usefulness in SLAM and for map alignment.



23	45	45	23	78	10
32	195	249	207	195	207
23	207	231	23	45	45
32	195	78	45	23	10

Figure 2.2: Gray-scale image visualization.

2.2.1 Image Processing

Image processing performs specific operations on an image, usually to enhance the image or extract useful information from it. These operations can include contrast enhancement, noise reduction, edge detection, and more. A key component of image processing in the context of computer vision and robotics is feature detection. Visual features are distinctive structures or patterns in an image, such as corners, edges, or blobs. These features can be used to match different views of an object or scene, track objects over time, or estimate the motion of a camera. Visual features are first detected by sliding a window over the entire image and based on the neighborhood around the current pixel, the algorithm will decide whether it is a suitable feature or not. The Harris Corner [22] detector determines this by computing the changes in image intensity in all directions from the center pixel and determining whether it is a corner. The FAST (Features from Accelerated Segment Test) feature detector determines if a pixel is a feature by selecting a ring of 16 pixels around the given pixel and using the intensity values of the ring to determine if the pixel is a corner. Figure 2.3 shows an example of a ”good” feature that is easy to distinguish, versus a ”bad” feature. Notice in figure 2.3, a bad feature is one that can match to many different places in the second image, but a good feature is one that only has one good match, this is referred to as ”saliency”.

Once features are identified in an image, we want to describe them in a way that allows us to recognize the same points in other images, regardless of scale, rotation, or illumination changes. Feature descriptors are usually represented as multi-dimensional vectors. The elements of these vectors quantify specific characteristics of the region around the detected feature. This could be the gradient (change in intensity) in different directions, color

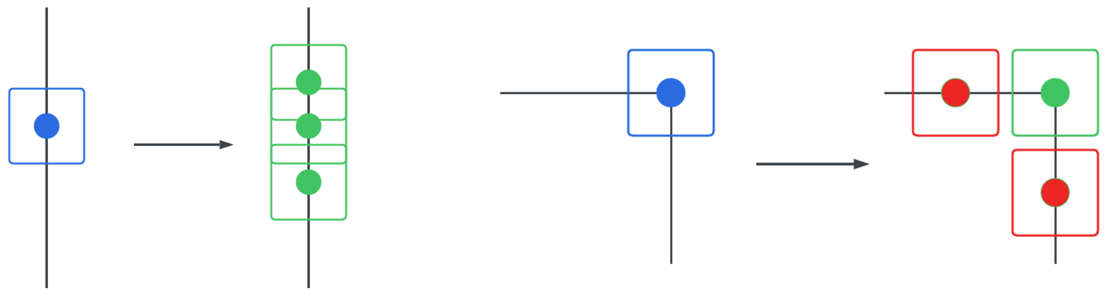

(a) Bad Visual Feature (b) Good Visual Feature

Figure 2.3: Comparison between a bad visual feature (a) and a good visual feature (b) A bad feature is one that has many potential matches, where as a good feature is one that has very few potential matches, this is known as saliency.

information, texture information, etc. SIFT (Scale-Invariant Feature Transform) [33], and ORB (Oriented FAST and Rotated BRIEF) [48] are two of the most well known descriptors used today, both relying on knowledge based algorithms, however there exists recent work in applying deep learning to local visual features, as shown in the SuperPoint descriptor [16].

The goal of feature description is to provide a unique, robust signature for each feature that can be easily compared and matched with features from other images. Feature matching is a critical part of many computer vision applications, including image stitching [36], structure from motion [71] and visual odometry [70]. Feature description is also used in visual place recognition (VPR) [51] solutions. VPR tries to determine if the current view from a camera matches a previously seen view stored in a database, even under drastically different viewing conditions (like changes in lighting, weather, or seasons) or from different viewpoints. A common approach to solving VPR with visual features is through the use of visual bag of words. The bag-of-words model is borrowed from the natural language processing (NLP) field which quantizes a document as a histogram of the frequency of words in it, it can then find similar documents by comparing the histograms of the documents. In a similar way, an image can be quantized as a histogram of visual words. The details on how feature descriptors are transformed into visual words for place recognition is described in [19].

2.2.2 Pinhole Camera Model

Cameras are extensively used in robotics for tasks such as object recognition, navigation, manipulation, and interaction with the environment. They provide a wealth of visual information that can be processed and used for decision-making. To use the information cameras provide in robotics, a mathematical model called the pinhole camera model is used to describe the geometric relationship between three-dimensional points in the world and two-dimensional points in an image. It assumes an ideal pinhole camera where light passes through a single point (the pinhole) and projects an inverted image onto the image plane. The pinhole camera model is often used to project 3D world points onto a 2D image, or to back-project 2D image points into 3D space. These transformations involve various camera parameters, such as the focal length, the camera center (or "principal point"), and potentially lens distortion parameters.

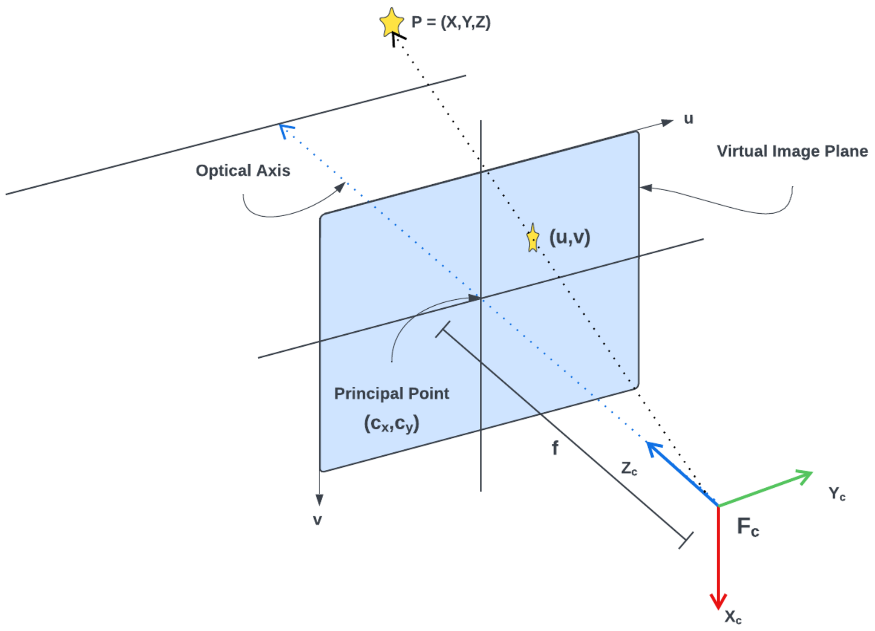


Figure 2.4: Pinhole camera model visualization.

To understand how the pinhole model works consider the 3D point in figure 2.4 $P = (X, Y, Z)$ in the world coordinate system. This point projects onto the virtual image plane through the optical center (principal point) of the camera. The virtual image plane is a conceptual plane where the 3D points in the world get projected onto 2D points. This plane is considered to be located at the focal length ,f, in front of the camera's optical center. The 2D coordinates (x, y) of the point on the image plane can be calculated from the 3D coordinates as $x = f(X/Z)$ and $y = f(Y/Z)$, where f is the focal length of the camera (the distance from the pinhole to the image plane). These image coordinates are in physical units. To convert these to pixel coordinates (u, v), we typically also need to consider two more parameters: the pixel scaling factors, denoted as (fx, fy) that describe how many pixels there are per unit distance in the x and y directions and the principal point (cx, cy) which is the location in the image where the optical axis intersects the image plane. These parameters give us the complete projection equation as shown in 2.5. There exists various other geometric models for representing cameras, with more ability to model the inherent distortion that lenses exhibit either by design or from manufacturing inaccuracies, [59] provides a detailed background on various geometric camera models in current literature.

$$\begin{aligned} u &= f_x \times X/Z + c_x \\ v &= f_y \times Y/Z + c_y \end{aligned} \tag{2.5}$$

This operation is often written and implemented in matrix form, as shown in 2.6

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times 1/Z \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.6}$$

The matrix in 2.6 is known as the camera intrinsic matrix, and is typically denoted by **K**. Note that while it is relatively simple to project 3D points into 2D pixels, the inverse from one observation alone is not possible. Consider the ray extending from the optical center through a pixel (u, v); all values along that ray correspond to the same X/Z and Y/Z. However, given a pixel value, the normalized direction vector in the camera frame can be calculated using Eq. 2.7.

$$\begin{aligned} \hat{x} &= (u - c_x)/f_x \\ \hat{y} &= (v - c_y)/f_y \end{aligned} \tag{2.7}$$

2.2.3 Multi-View Geometry

Multi-View Geometry (MVG) is a concept in computer vision which refers to the process of determining geometric information about a 3D scene from two or more images taken from different viewpoints. The process typically involves 3 steps: 1) Feature matching/tracking across images 2) Relative pose/egomotion estimation 3) Visual feature triangulation.

In the last section we described the concept of visual features. Tracking features between two images is typically done in one of two ways. The first is method is known as descriptor based tracking. The approach aims to match features across subsequent images by finding the best fitting feature in the previous image to each feature in the current image, doing this for every image results in a track across arbitrary images. The second method to track features across images is to compute the optical flow on the image and use it to estimate the motion of the feature from the previous image to the current one. Optical flow is the apparent motion of brightness patterns in the image. The Lucas-Kanade method [34] is a well-known method of optical flow based visual feature tracking.

Relative pose estimation is the process of estimating the motion of a sensor between two consecutive sensor measurements. In the case of visual sensors this is typically done by estimating the Essential or Fundamental matrix. The Essential matrix is a 3x3 that encapsulates the rotation and translation between two camera views, assuming calibrated cameras (i.e., the intrinsic camera parameters like focal length and principal point are known). The Fundamental matrix is also a 3x3 matrix, however it also encodes the intrinsic calibration information of the camera. Estimating the relative motion between two images only requires a set of 2D-2D (pixel-pixel) correspondences. For details on how the essential matrix is estimated from pixel correspondences see [23].

Feature triangulation is a process used in computer vision to determine a 3D point in space from several 2D observations of that point. It uses the principles of triangulation: if you know the position and orientation of two cameras when they capture the same point, you can cast rays along the viewing direction from each camera to that point. The intersection of these rays in 3D space gives the position of the point, this is illustrated in 2.5.

The initial estimates of the cameras motion, and visual feature locations are typically refined further through a minimization of reprojection error. Reprojection error is the distance between the original 2D feature point and the point where the 3D point reprojects. Lower reprojection error indicates that the 3D point and the camera poses are likely to be accurate. Minimizing this error provides the best estimate of the 3D structure and camera motion given the measurements available.

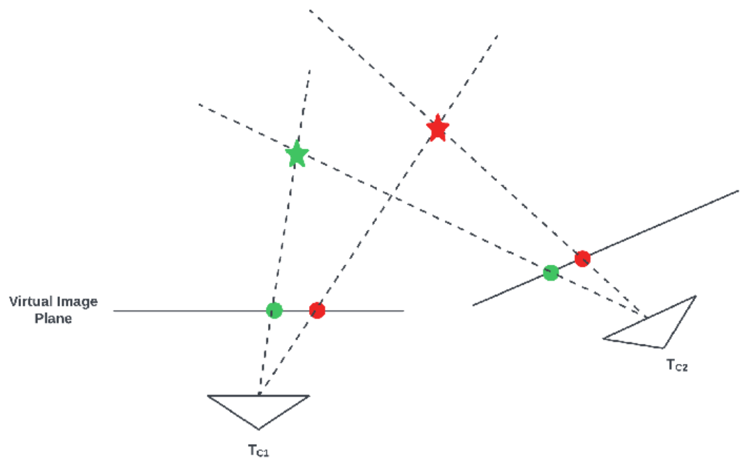


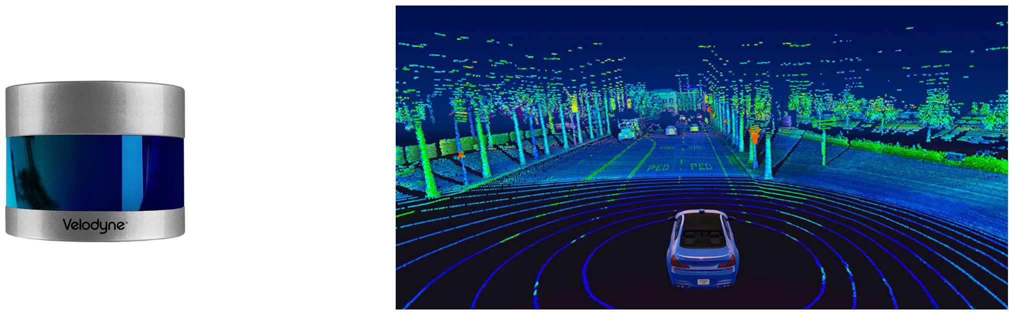
Figure 2.5: Visual feature triangulation.

In the case where feature locations have already been estimated, pose estimation can be done using 2D-3D correspondences, opposed to 2D-2D correspondence that essential matrix estimation uses. This problem is known as the Perspective-n-Point (PnP) problem. As the name suggests, the PnP problem refers to determining the pose of a camera given n correspondences between 3D points in the world and their 2D projections in an image. There are various algorithms to solve the PnP problem, depending on the specifics of the situation. For example, if we have exactly three point correspondences, we can solve the problem directly. This is known as the Perspective-3-Point [20, 42] (P3P) problem. If we have more than three correspondences, we can use methods like EPnP [31] (Efficient Perspective-n-Point), DLS (Direct Least Squares) [24], or others.

2.3 LiDAR Sensors

We now go on to our examination of LiDAR sensors after our consideration of vision sensors. These sensors, which differ from visual sensors in that they directly sense the depth of the immediate surroundings, are incredibly helpful for 3D mapping and infrastructure inspections. LiDAR technology creates precise three-dimensional reconstructions of the target environment by using the time-of-flight (ToF) of pulsed laser light to estimate distances.

LiDAR sensors are typically classified into two categories based on their scanning mechanism and the data they provide: 2D and 3D LiDAR sensors.



(a) Velodyne Lidar [68] (b) Lidar scan rendering [67]

Figure 2.6: Example lidar sensor and scan rendering.

1. **2D LiDAR Sensors:** Also known as single-plane LiDAR, these sensors capture a horizontal 'slice' of the environment, providing distance measurements in a single plane. These are often used in ground-based robotic applications, where they assist in tasks such as obstacle detection and avoidance, and navigation. An example of this type of sensor is the Hokuyo UST-10LX.

2. **3D LiDAR Sensors:** These sensors capture data in multiple planes, providing a three-dimensional point cloud of the surrounding environment. These sensors find extensive use in autonomous vehicles and aerial robots for tasks like 3D mapping and complex navigation. Velodyne's VLP-16 and Ouster's OS1 are examples of 3D LiDAR sensors.

To achieve a 360° scan of the environment, traditional (mechanical) LiDAR sensors are equipped with a set of vertically stacked lasers (ranging from 16 to 128 beams), which are then spun around rapidly. Traditional LiDAR sensors have several limitations, including higher costs, larger form factors, and lower durability due to the presence of moving parts. Solid-state LiDAR sensors, on the other hand, have no moving parts. Instead, they steer the laser beam electronically. Solid-state LiDAR sensors have a smaller form factor, improved durability, lower power consumption, and are more cost-effective compared to their mechanical counterparts, primarily because of the absence of moving parts. However, their range and resolution are often lower than traditional LiDAR sensors. An example of a solid state LiDAR is the Livox Mid-70.

While visual sensors primarily rely on ambient light and can provide rich texture information, they may struggle in low light or high contrast environments. On the other hand, LiDAR sensors are active sensing systems and are therefore less affected by lighting conditions, but do not capture color or texture information. Often, both visual and LiDAR sensors are used in conjunction, leveraging the strengths of each to provide a more complete understanding of the environment.

2.3.1 Scan Registration

Scan registration refers to the process of aligning multiple LiDAR scans (also known as point clouds) into a common coordinate system. This is crucial for creating a consistent and accurate representation of the environment. The process involves transforming individual scans so that the overlap between consecutive scans is maximized.

There are several algorithms and techniques that can be used for scan registration. A commonly used method is the Iterative Closest Point (ICP) algorithm [4]. ICP minimizes the difference between two clouds of points by iteratively adjusting one cloud (the "source") to best match another (the "reference"). It does this by minimizing the sum of the distances between corresponding points in the two clouds, readjusting the correspondence estimation, and minimizing again, until convergence. Consider a given set of point correspondences, the minimization process aims to solve the least squares problem expressed in Eq. 2.8

$$T^R_{St} = \underset{T^R_{St}}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \left[r_i - T^R_{St} s_i \right]^2 \tag{2.8}$$

In Eq. 2.8, N is the number of corresponding points, r_i and s_i are the i^{th} corresponding points from the reference and source point clouds. T^R_{St} is the transform from the source frame to the reference frame at time t. Now knowing the heart of the ICP algorithm we can outline the main steps of the ICP algorithm:

1. **Initialize:** Start with an initial estimate of the transform from the source cloud to the reference cloud. This could be identity if the two scans are close, or a transform estimated from other sensors.
2. **Correspondence estimation:** For each point in the source cloud, find the point in the reference cloud that is closest to it in space (nearest neighbour search)
3. **Minimize Point-to-Point Error:** Given the set of correspondences, solve Eq. 2.8 to estimate the relative transform

4. **Apply the transformation:** Apply the computed transform from Step 3. to the source cloud

5. **Iterate:** Repeat steps 2-4 until the change in error between iterations drops below a certain threshold

Once the scans have been registered, the relative transformations computed during this process can be used to estimate the sensor's motion. This is a crucial step in LiDAR-based SLAM systems. Additionally, the aligned point clouds can be merged to create a comprehensive 3D map of the environment.

ICP and its variations [4, 52], however, are susceptible to local minima, which means that if the initial alignment is not sufficiently close to the proper one, the algorithm may converge to an inaccurate solution. Additionally, ICP makes the assumption that areas in the two scans are almost identical to one another, which might not be true in dynamic situations or when there are occlusions. More durable methods have been developed in order to address these problems. An example of a registration method that is more resistant to changes in point density is the Normal Distributions Transform (NDT) [5], which matches distributions of points rather than specific ones. Using feature-based approaches, which extract features from the point clouds and match them instead of using raw points, is an alternative strategy utilized in [63].

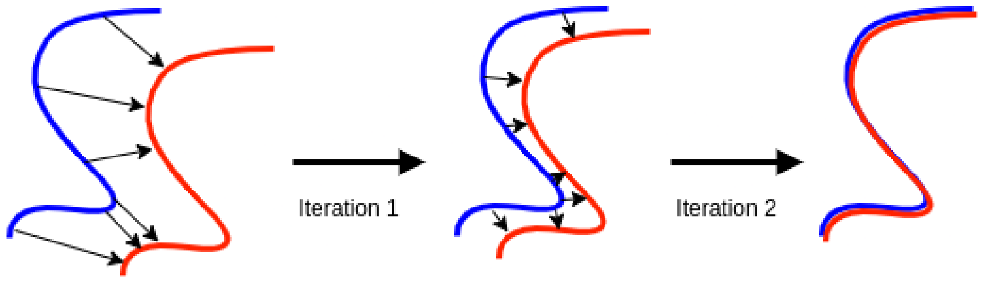


Figure 2.7: Visualization of the Iterative Closest Point (ICP) algorithm.

2.4 Simultaneous Localization and Mapping

SLAM is a fundamental problem in robotics and computer vision that involves the task of a robot or an agent simultaneously localizing itself in an unknown environment while constructing a map of that environment. SLAM is particularly important in scenarios where external localization systems, such as GPS, are not available or unreliable, such as indoor environments, underground areas, or areas with limited sensor range.

SLAM systems typically utilize various types of sensors, such as cameras, lidars, sonars, or even inertial measurement units (IMUs), to perceive the environment and estimate the robot's position. The process involves a combination of two main tasks: mapping and localization.

1. **Mapping**: This involves constructing a representation of the environment. The robot collects data from its sensors and uses it's current pose estimate to create a map that represents the layout, geometry, and features of the surroundings. The map can be represented in different forms, such as occupancy grids, feature-based maps, or point clouds.

2. **Localization:** Simultaneously, the robot estimates its own position within the constructed map. By comparing sensor measurements with the map, the robot determines where it is located in the environment. This is achieved through techniques such as multi-view geometry 2.2.3, scan registration 2.3.1, or probabilistic filtering methods like the Extended Kalman Filter (EKF)

Additional to the primary task of Localization and Mapping, a full SLAM system also incorporates loop closure. Loop closure is the process of detecting and correcting previous errors when the agent or robot revisits a previously explored location. It involves two steps: 1) place recognition and 2) correcting it's pose through establishing correspondence between its current measurements and the existing map. Additionally, map re-use is a task that is often overlooked but is essential to the creation of a comprehensive SLAM system. Utilizing an existing map or prior environmental knowledge to help the SLAM process is known as map re-use. This can involve adding well-known landmarks or structures into the mapping process or using pre-existing maps as a prior for localization. Reusing maps increases accuracy and efficiency by making use of already available data.

SLAM algorithms can be broadly classified into two categories: recursive and batch. Recursive SLAM [6, 61] performs real-time mapping and localization, continuously updating the map and position estimates based only on the current map and robot state and the newest sensor information. Batch SLAM [15, 45, 63], on the other hand, aims to estimate the map and robot state by minimizing the error between what the sensors perceived from what is currently estimated as the map and robot state, using sensor measurements from all time steps. For the mathematical underpinnings of these methods, see Appendix A.

Chapter 3

Literature Review

3.1 Remote Sensing and Automated Infrastructure Inspections

Remote sensing can be described as any method that uses sensors to capture data from a distance (without physical contact with the object(s) being studied). The application of various remote sensing techniques applied to infrastructure inspections has been very well studied. One of the most commonly used sensor in the civil and construction industry is the Terrestrial Laser Scanner (TLS). A TLS is a ground-based remote sensing technology that measures distances using laser light. A TLS device emits a beam of light toward a target object and measures the time it takes for the reflected light to return to the sensor. Using the speed of light, the system then calculates the distance to the target. By rotating the laser and detector assembly, the system can scan its surroundings and generate a 3D point cloud representation of the environment.

The work in [57] compares the accuracy of TLS based inspection to manual surveying of bridges in multiple bridge inspection goal metrics (structure length, sidewalk width, minimum under clearance etc). The method utilizes a pipeline that takes as input the bridge inspection goals and the laser scanner type to determine the best locations to collect scans to achieve the goals given. The next task is to filter the raw point cloud data with a Gaussian filter to reduce noise. Finally, in the data interpretation phase, the inspector manually segments the cloud to fit geometric primitives (ellipsoid, plane, cylinder etc). After this step, model fitting methods are used to automatically detect the different bridge components, resulting in a richly labelled semantic point cloud of the bridge. Based on this information inspectors can calculate values of the bridge inspection goals either manually or automatically. The primary advantage of this method is the data comprehensiveness of the TLS, whereas the previous methods would utilize very sparse laser scanners. The dense clouds the TLS provides can accurately detect surface details (such as potential deformations). Another advantage of this process is the reduction in inspection time, where the manual process would take one whole day of on-site time and weeks for off-site processing, their process reduces the process to roughly 2 hours and a few days, respectively. Most methods of using TLS for infrastructure inspection follow a similar pipeline of data collection, 3D model generation, semantic labeling, and measurement gathering. For a more in-depth review of recent TLS based infrastructure inspection methods see [46].

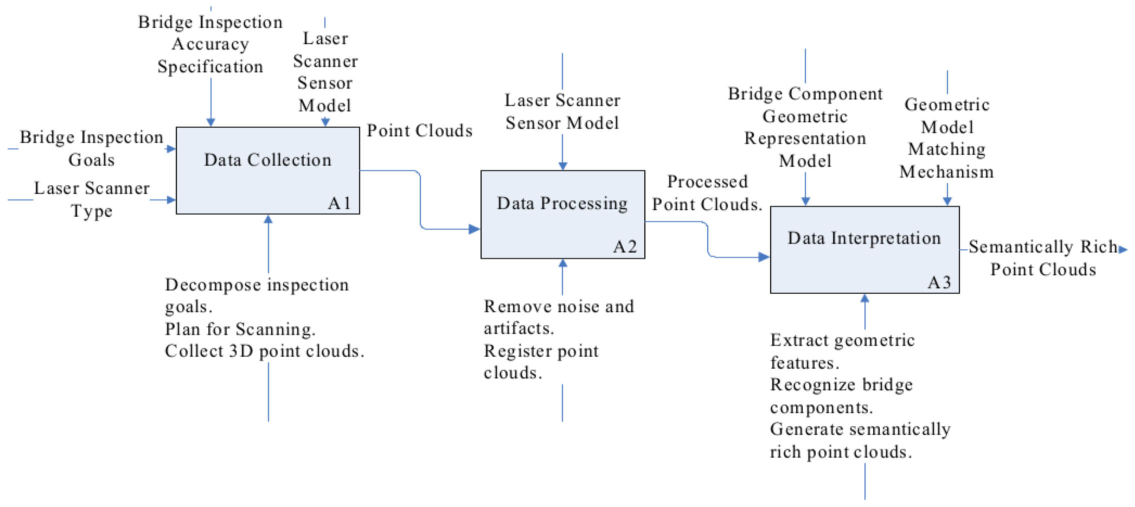

Bridge Inspection Accuracy Specification
Laser Scanner Sensor Model
Bridge Inspection Goals
Laser Scanner Type
Data Collection
A1
Point Clouds
Decompose inspection goals.
Plan for Scanning.
Collect 3D point clouds.
Laser Scanner Sensor Model
Data Processing
A2
Remove noise and artifacts.
Register point clouds.
Processed Point Clouds.
Bridge Component Geometric Representation Model
Geometric Model Matching Mechanism
Data Interpretation
A3
Semantically Rich Point Clouds
Extract geometric features.
Recognize bridge components.
Generate semantically rich point clouds.

Figure 3.1: TLS data collection process, taken from [57]

While TLS scanners have been shown to provide extremely accurate and dense 3D models that inspectors can use to automate the inspection process, they can also be extremely expensive and can still take a relatively long time for data collection and processing. Recent research in mobile 3D mapping has shown its efficacy in reducing data collection and processing time. Taking advantage of localization and mapping algorithms can significantly reduce the laborious and time-consuming task of 3D scanning with a TLS.

The method proposed in [29] uses a robotic system equipped with Ground Penetrating Radar (GPR), Global Positioning System (GPS), Acoustic Arrays, Electrical Resistivity and downward facing surface cameras to perform a comprehensive inspection of bridge decks. They utilize Extended Kalman Filtering for localization using GPS, Inertial Measurement Unit (IMU) and wheel encoders, while other sensors are used primarily for post processing and analysis. The robot is given preselected start and stop locations using GPS and then utilizes a path planning algorithm so that it will cover the entire inspection area. Using the resulting trajectory and the surface cameras which take images every 2 feet, the images are stitched together to create one uniform image of the entire bridge deck. Following this, concrete cracks on the surface are automatically detected using a gradient based method and linked together. Given known distance of the cameras to the planar surface of the bridge deck the scale of pixels can be accurately estimated, the results of the inspection is a single image of every concrete crack on the image surface in real world scale. They also propose to use a "start-stop" approach to scanning where the robot will stop at regular intervals to deploy sensors that require contact with the bridge surface, this data can similarly be stitched together to provide a to scale map of the delaminations (a delamination is concrete that has been internally separated from the main structure, but it still attached) and corrosions found within the bridge deck. The approach presented is limited by its ability to extract crack scale only on bridge decks, and its need to stop regularly to collect more detailed data.

Unmanned ground vehicles (UGV) have exhibited considerable efficacy in conducting broad-spectrum inspections of infrastructure, as illustrated in [11]. The proposed robotic platform is equipped with a GPS, an IMU, a thermal imaging camera, an RGB camera, and a pair of LIDAR systems, oriented both horizontally and vertically, to ensure comprehensive scanning capabilities. While a human operator controls the robot with a remote controller, a recursive state estimator, the Extended Kalman Filter (EKF), is employed for the execution of Simultaneous Localization and Mapping (SLAM). Details of SLAM algorithms will be explained in subsequent sections. This process utilizes the GPS, IMU, wheel encoders, and LIDAR data to effectively estimate the robot's position and orientation within its environment. In turn, the resulting trajectory obtained via the EKF-based SLAM provides a foundation for constructing a dense three-dimensional map through the assimilation of data sourced from the LIDAR systems. Upon the generation of the 3D map, colorization/labelling is accomplished utilizing the RGB or thermal camera. A ray-tracing algorithm is applied to preclude inaccurate colorizations arising from occlusions. The study additionally puts forward a semi-automated pipeline for the detection of concrete defects, thereby enabling the identification of issues such as cracks and exposed rebar, as evident in the RGB images, and delaminations discernable in infrared (IR) images. Delaminations are particularly discernible in IR images owing to the temperature differential existing between the created air gap and the intact concrete.

The study [35] extends upon the preceding research, incorporating advancements in Simultaneous Localization and Mapping (SLAM) methodologies and the introduction of neural network-based mechanisms for the detection of structural defects. The configuration of the robotic platform has been augmented with two additional RGB cameras featuring fisheye lenses, positioned upward and forward-facing, to significantly enhance its colorization capacity. In its localization strategy, this method implements a dual phase strategy. The first phase involves online LIDAR odometry, as proposed in [63], to establish an initial trajectory. Following this, a technique known as ”batch estimation” is employed. Batch estimation refers to a process where all data is processed as a single large optimization problem, or batch, rather than sequentially or incrementally. In the context of this research, the robot’s sensor data is collectively analyzed in an offline environment to refine the initial trajectory estimate and enhance accuracy. Further details of batch estimation will be elaborated on in subsequent sections. Lastly, this approach implements a neural network model for the fully automated detection of concrete spalls and delaminations within RGB and thermal images, respectively. As a final component, a mechanism for the quantification of these detected defect areas is introduced, enabling a thorough examination and quantification of infrastructural impairments.

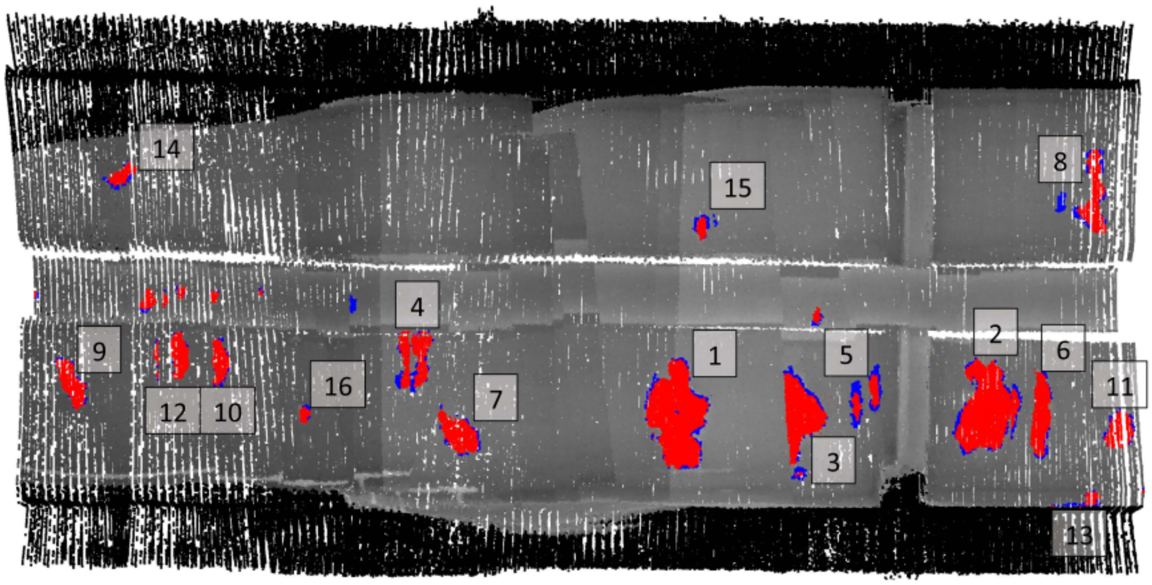


Figure 3.2: Point Cloud data labelled with IR images and automatically segmented delaminations (red), taken from [35]

Artificial neural networks have demonstrated significant efficacy in the identification of small defects in visual data. In addition, noteworthy research has been conducted to enable the recognition of more generalized semantic elements, such as girders, columns, and roads, directly within three-dimensional (3D) datasets, as demonstrated in the study [62]. This research introduces a methodology to autonomously segment a point cloud representing a bridge into specific semantic classes, namely background, slab, girder, pier cap, and pier. The classification process implemented in the study is bifurcated. The initial phase involves the extraction of local descriptors, utilizing the Signature of Histograms of Orientations (SHOT) [49] technique, from the point cloud. These descriptors are subsequently fed into a fully connected neural network, which has been trained to perform the segmentation into the aforementioned classes.

The amalgamation of this research with automated 3D mapping techniques of infrastructure assets, along with the identification of visual defects, renders a comprehensive and precise 3D model for engineering inspection. However, the current state of research in automated infrastructure inspections have been primarily focused on improving the accuracy of single session scans while disregarding the advancement of the robustness and generalizability of existing relocalization methods for accurate alignment of maps.

3.2 Multi-Session SLAM

SLAM is a very important tool in the context of automated infrastructure inspections as it can enable to ability to generate accurate 3D representations of the infrastructure asset to be inspected. Although many SLAM methods can be used to achieve this, this section will focus on the existing multi-session SLAM packages, due to their ability to enable associations across time which is very important for the lifetime maintenance of infrastructure assets.

Multi-session mapping is a technique used in the field of SLAM where the goal is to construct a consistent and unified map from multiple mapping sessions, collected at different times, under different conditions, or even by different devices. This approach is useful in several scenarios such as long-term mapping, updating maps over time, or in situations where multiple robots are employed to map large or complex environments. Multi-session mapping typically includes the following steps:

1. **Individual Mapping Sessions:** Each session generates a local map of the environment during that specific scan.

2. **Map Merging:** These local maps are combined into a global map by determining and applying the relative transformations between the individual maps.

3. **Optimization:** After merging, global optimization is performed to minimize inconsistencies and errors, resulting in a more accurate and unified map.

4. **Updating Over Time:** The global map can be updated as more mapping sessions are conducted, reflecting changes in the environment over time.

Due to the difficulty of incorporating robust multi-session mapping capability, many academic SLAM works choose to ignore it altogether, opening up a need for a map alignment algorithm that can be used irrespective of the SLAM method used. This literature review aims to unpack the current SLAM techniques that incorporate some form of multi-session mapping capability and highlight their limitations in their application to repeat infrastructure inspections.

The landscape of SLAM packages reveals a scarcity of robust multi-session mapping solutions, the majority of which are primarily vision-based. Notably, one of the pioneering visual multi-session mapping solutions, RTAB-Map, was unveiled in 2014 [30]. RTAB-Map implements its estimation as a batch optimization (graph SLAM), whereby edges between camera poses represent relative motion constraints supplied by a user-selected visual odometry method (options include VINS-Mono [45], OKVIS [32], MSCKF [56], among others), this is known as loosely coupling the odometry from the full pose graph. In addition to establishing odometry constraints between sequential camera poses, RTAB-Map actively searches for potential loop closures employing a visual bag-of-words approach. Upon identifying a previously visited location, a loop closure hypothesis is generated, subject to verification via a Bayesian filter. This process ultimately determines the acceptance or rejection of the hypothesis. Upon acceptance, a loop closure correction constraint is introduced into the graph for subsequent optimization. RTAB-Map's multi-session mapping functionality is incorporated using a mechanism similar to its loop closure process. By loading a previous map, the current session persistently seeks loop closures in both the past and current maps. A significant enhancement of RTAB-Map includes the ability to integrate LiDAR measurements, thereby enabling users to employ LOAM [63] as the odometry source, and to use it for refining the loop closure correction estimate. Notably, while LiDAR is used for odometry and loop closure refinement, in the absence of visual information, the multi-session functionality becomes a manual process. This condition necessitates the user to provide a coarse estimate of the device's location in the previous map. Due to the sole reliance on visual information for place recognition, RTAB-Map can struggle in dynamic environments and may fail to relocalize in a previous map.

ORB-SLAM3 [10] introduced in 2021, extends ORB-SLAM [39] and ORB-SLAM2 [40] with IMU integration and with multi-session mapping capability. Similarly to RTAB-Map, ORB-SLAM3 formulates its estimation as a graph based optimization, however it incorporates reprojection error constraints between camera poses and visual landmarks. Again, like RTAB-Map, ORB-SLAM3 uses a visual bag of words approach for loop closures which will search for loop closures in the current map, as well as in every other map that has been loaded as prior information to ORB-SLAM3. Upon successful loop closure, ORB-SLAM3 also introduces the concept of map-merging which will load both maps and intelligently merge them into a single current map. ORB-SLAM3's primary claimed advantage is their inclusion of 3 levels of correspondence: short term between subsequent images, medium term between the current image and the local map, and long term between the current image and all previous keyframes for loop closure. ORB-SLAM3, like all visual methods, struggle in detecting loop closures in similar regions or in highly dynamic environments.

Arguably the most comprehensive solution for multi-session mapping is in Maplab [50], an open-source framework that was initially launched in 2018 and subsequently extended with additional features in the 2022 update, Maplab 2.0 [12]. Like RTAB-Map [30], Maplab decouples the odometry and mapping processes, thereby empowering users with the liberty to opt for their preferred odometry. As the selected odometry processes the data, the Maplab server concurrently constructs the map for that particular session, facilitating either real-time multi-robot mapping or offline map alignment and merging. The integration of LiDAR in Maplab is analogous to that in RTAB-Map [30]; it is employed either as a source of odometry or to rectify loop closure using the ICP algorithm. Nevertheless, place recognition does not utilize LiDAR, thereby implicating a shared limitation with dynamic environments to RTAB-Map. However, the design of Maplab is inherently flexible, allowing for straightforward augmentation with any place recognition algorithm, including those that leverage fused visual-LiDAR based approaches.

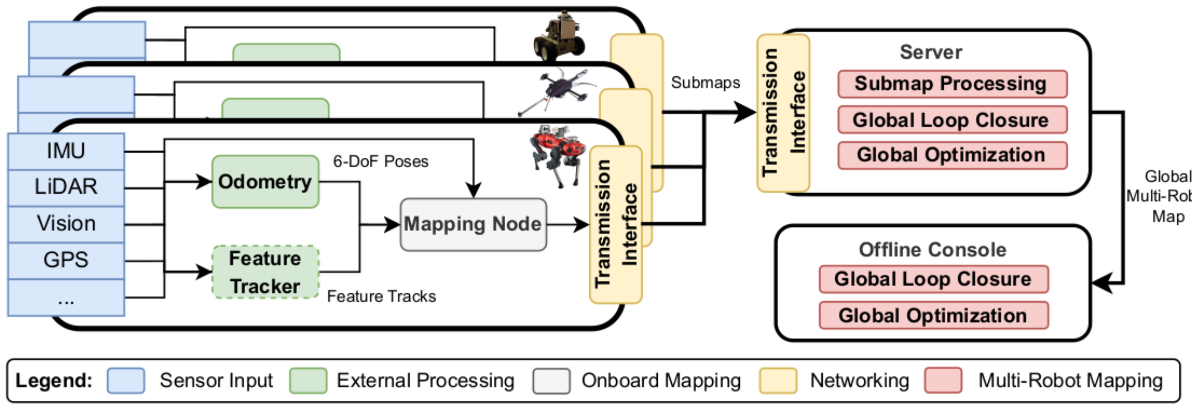


Figure 3.3: Maplab 2.0 framework, taken from [12]

Within the realm of open-source LiDAR SLAM, multi-session mapping capabilities, especially pertaining to 3D LiDAR, is less abundant compared to the methods prevalent in visual SLAM. The software package Cartographer [25], which offers both 2D and 3D LiDAR SLAM features, including loop closure and multi-session capability, unfortunately confines its multi-session capability to 2D LiDAR. This makes it unsuitable for infrastructure inspection tasks, which requires rich 3D data for precise evaluation. To the best of the authors' knowledge, the most robust long-term 3D LiDAR SLAM package available is LT-Mapper [28]. It offers a full 3D LiDAR SLAM package with multi-session capabilities and supports persistent mapping. LT-Mapper employs the Scan Context [27] global LiDAR descriptor for place recognition, although it does not integrate any visual information into this process. Persistent mapping refers to the generation of a map that captures elements considered to be enduring across multiple mapping sessions of the area. During the mapping against a persistent map, LT-Mapper concurrently generates an updated live map, storing the difference between the two in the form of a delta map. This results in a computationally efficient method for change detection and updating of the persistent map.

The multi-session SLAM packages discussed here which have primarily employed either visual or LiDAR-based place recognition techniques to achieve map alignment. However, it is noteworthy that, to the best of the authors knowledge, there exists a research gap regarding the exploration and investigation of a map alignment strategy that effectively amalgamates both visual and LiDAR-based place recognition algorithms. Such an approach would potentially enhance the robustness and accuracy of map alignment, thereby warranting further investigation and research attention in this domain.

3.3 Place Recognition

Place recognition is the ability of a robotic system to identify previously visited locations using its sensory data. This capability is important to SLAM processes, as it assists in loop closing when the robot revisits a previously explored environment thus correcting any accumulated drift in the mapped environment. Using any combination of visual, LiDAR, or other sensory information, place recognition enhances localization accuracy and in turn mapping accuracy.

Place recognition algorithms typically fall into two categories: knowledge based or deep learning based. A common knowledge based visual place recognition algorithm is the Dynamic Bag of Words algorithm (DBoW) [19]. The BoW model, in general, treats an image as a collection of visual words. A visual word corresponds to a specific feature descriptor extracted from the image. These descriptors are then clustered, and each cluster's centroid becomes a visual word. These visual words form a vocabulary. When a new image is encountered, its descriptors are assigned to the closest visual word in the vocabulary, and the image is represented as a histogram of these visual words. DBoW extends this model with a dynamic approach, accommodating the vocabulary to be updated as new images and features are encountered. This makes it more robust in real-time applications where new environments may constantly be encountered, such as autonomous driving or robotic navigation. Moreover, DBoW2 uses binary descriptors instead of traditional real-valued ones. This leads to faster processing and less memory usage, while still providing high accuracy, making it a popular choice for loop closure detection in SLAM applications. Learning based place recognition algorithms have gained much popularity, NetVLAD [1] is one such algorithm. NetVLAD was designed as a way to aggregate local descriptors into a compact fixed-size global descriptor, also known as a Vector of Locally Aggregated Descriptors (VLAD), it is a feature encoding method similar to the BoW encoding. In the context of place recognition, typically a VLAD is computed by aggregating many local descriptors from an image (for instance, SIFT [33] descriptors) into a single fixed size descriptor to represent the image as a whole. The advantage of this approach is that the VLAD descriptor has a fixed size, regardless of the number of local descriptors, and it retains a certain amount of spatial information that gets lost in other aggregation methods like BoW. NetVLAD takes this idea and integrates it into a trainable Convolutional Neural Network (CNN). The NetVLAD layer in this CNN is essentially a differentiable version of VLAD, which means that the entire network, including the parameters of the NetVLAD layer, can be trained end-to-end. This makes it possible to train the network to produce global descriptors that are directly optimized for the place recognition task, resulting in significantly improved performance compared to traditional methods.

In a similar vein to visual place recognition, LiDAR place recognition is typically solved using knowledge based algorithms or learning based. A very common knowledge based LiDAR place recognition algorithm is the Scan Context [27] algorithm. Scan Context creates a global LiDAR descriptor by converting the 3D point cloud data from the LiDAR scan into a 2D matrix, by creating "bins" on the point cloud. The Scan Context descriptor, illustrated in 3.4, is a matrix where the vertical axis represents the specific ring from the LiDAR scan and the horizontal axis is the discretized "sectors" from the 360° scan, the values in the matrix represent the maximum height of the points in that "bin". The resulting descriptor is essentially a panoramic view of the environment from the sensor's point of view. On top of implementing the descriptor itself, an efficient two-phase matching algorithm is introduced for efficient place recognition. Alternatively, Link3d [14] implements a local LiDAR keypoint detector and descriptor focused on finding edge features in a point cloud and efficiently describing them in a vector. BoW3D [13] extends the Link3D [14] descriptor with bag of words functionality, allowing for efficient place recognition in LiDAR data. PointNetVLAD [60] is a deep learning architecture for point cloud based place recognition. It was introduced as a modification of the original PointNet [44] architecture and is designed to generate global descriptors of 3D point clouds for the purpose of place recognition in large-scale environments. The first step in the algorithm is to pre-process each point cloud using PointNet [44] to generate a global descriptor of the scan. The second step is to feed the global descriptor into NetVLAD [1] to learn the optimal representation of the global descriptor for place recognition.

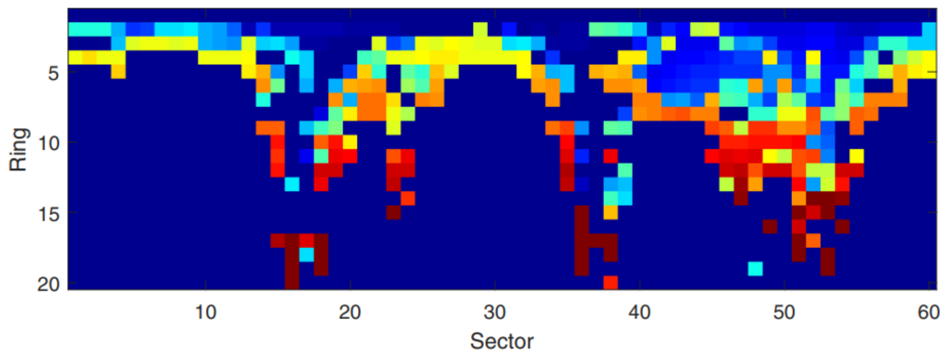


Figure 3.4: Scan Context descriptor, taken from [27]

Similarly to the discussed multi-session SLAM methods, the place recognition methods discussed here typically employ only a visual based approach or a LiDAR based approach. While it is fairly common to use visual data for place recognition and LiDAR data for pose estimation, there has been fairly little exploration on using both visual and LiDAR data for place recognition to reduce false positives.

3.4 Thesis Contributions

SLAM has undergone significant breakthroughs, and much study has been done on its use in infrastructure inspection. However, because they are often limited to single-session mapping, open-source and academically derived SLAM algorithms continue to have a considerable gap in their practical usefulness. There is a real need for a generic map alignment algorithm given that researchers introducing new SLAM approaches frequently do not focus on multi-session mapping. The usefulness of automated infrastructure inspections using SLAM would drastically increase with a method to align data across scanning sessions that is robust to environmental changes for accurate reporting on the change in infrastructure health over time. This thesis aims to implement a map alignment algorithm utilizing both visual and LiDAR information for place recognition and pose estimation. This thesis will also study the proposed algorithm compared to existing methods. The addition of this component has significant ramifications for both the general field of SLAM and for its particular application to infrastructure inspection.

1. A review and discussion of existing multi-session SLAM packages and place recognition techniques, proposing an alternative multi-session mapping approach using offline map alignment that is independent of the SLAM method of choice, allowing for greater exploitation of state-of-the-art SLAM methodologies (Chapter 3).
2. Collection of an infrastructure centric dataset for validation and for further use in automated infrastructure inspection research (Chapter 4, and Chapter 5)
3. Implementation of a map alignment approach utilizing a combination of approaches in existing work to enhance robustness (Chapter 4).
4. Validation on two infrastructure centric datasets (Chapter 5).

Chapter 4

Methodology

The methodology proposed here for the automated inspection of infrastructure over multiple sessions is outlined broadly in Fig. 4.1, green segments are the segments focused on in this thesis. The pipeline starts with data collection via a handheld scanning device, in which visual, inertial and LiDAR data are collected. Subsequently, this data is utilized to estimate the trajectory of the device and construct a 3D point cloud of the infrastructure asset. In the optional third stage, the focus of this thesis, the immediate trajectory and associated map can be aligned with a prior generated map of the same area. This provides a chronological perspective of the asset's condition. Provided the collected data, the computed 3D model and trajectory are available, various inspection tasks can be computed for each scan (dimension measurements, defect detection and quantification, etc). And lastly, given that each scan is aligned in the same reference frame, reporting on the change between inspections can be performed automatically. This thesis will display the efficacy of the proposed method by automatically computing and visualizing the structural change between the scans.

4.1 Data Collection

4.1.1 IG Handle

The data required to implement the proposed map alignment strategy is time-synchronized visual and LiDAR data, as well as an initial estimate of the trajectory of any of the sensors. To enable this data collection the Sensing and Robotics Lab (SRI Lab) at UCLA (formerly

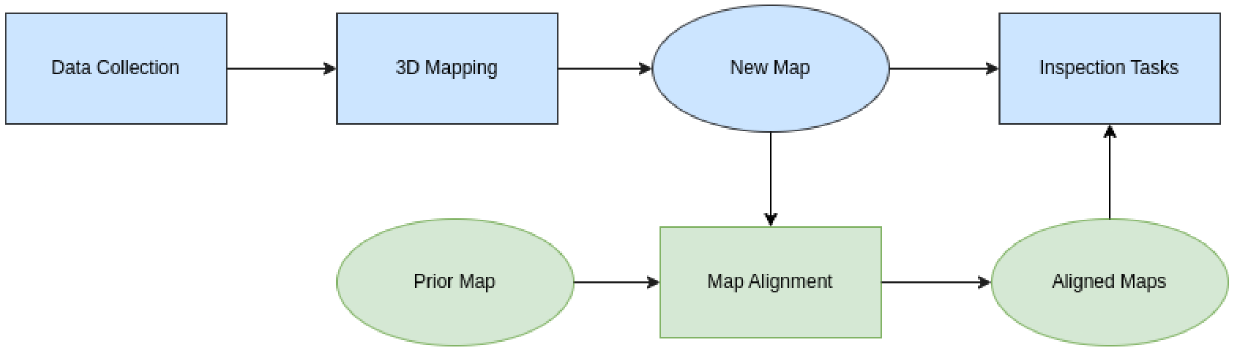


Figure 4.1: Full Inspection Pipeline.

the Structural Dynamics Identification and Control (SDIC) at the University of Waterloo) has developed a modular platform 4.2, designated as IG Handle, capable of being operated handheld by a single inspector or being mounted on various types of robotic platforms. The specific sensors the platform is equipped with are:

- one Flir Blackfly USB3 RGB camera with small FOV lens
- two Flir Blackfly USB3 RGB camera with wide fisheye lenses
- one Xsens MTi-30 AHRS IMU
- one Velodyne VLP16 LiDAR

The data collected from these different sensors must be integrated and synchronized spatially and temporally. To achieve accurate and consistent timestamps across every sensor the platform uses a micro-controller and a central computer that have synchronized clocks (through a handshake process). The micro-controller is used to trigger the sensors when to collect data; using this information, the main computer can determine the precise time that the data was collected (not when the data was received).

4.1.2 Intrinsic Calibration

Sensor calibration involves both intrinsic calibration and extrinsic calibration. Intrinsic calibration is the process of determining the characteristics of each sensor (intrinsic to the sensor). LiDAR intrinsic information is typically provided by the manufacturer and accounts for various noise that is introduced from manufacturing inaccuracies. IMU intrinsic

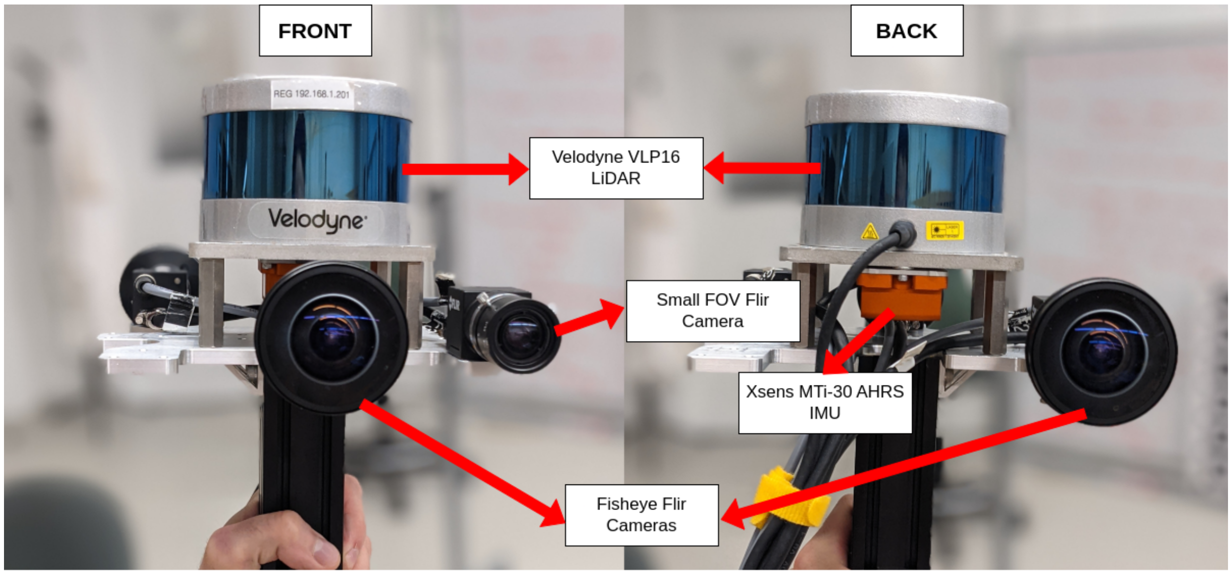


Figure 4.2: IG Handle. (Photos by Nicholas Charron)

calibration involves characterizing the noise profiles that the IMU is susceptible to; for an in-depth review of how IMU noise is characterized see [43]. However for most applications the noise that can be accurately accounted for are divided into two aspects. The first noise value is the noise density which essentially provides the standard deviation of the sensors noise over a frequency of 1Hz. The second noise is random walk noise which emerges due to the integration of white noise (an IMU reports data as linear acceleration and angular velocity, both requiring integration to retrieve position and orientation). Random walk noise indicates how integration error will accumulate over time. The process of calibrating and determining these noise values involve recording data from the IMU when it is completely stationary for an extended period of time (at least 3 hours) so that the calibration process can assume that any variance in the data being recorded is due to noise and not from actual motion. The IMU noise intrinsics are calibrated using the Allan Variance ROS calibration tool [8].

The Flir Blackfly cameras must be calibrated to find their focal length, principal point, and distortion parameters. These parameters are calibrated using the Kalibr [47, 18]. The checkerboard target method for calibrating for a single cameras intrinsic parameters was used. The process involves using a checkerboard target 4.3 with known dimensions, along with automated checkerboard detection in the image to retrieve known 2D to 3D correspondences. The calibration process is an optimization problem that aims to find the

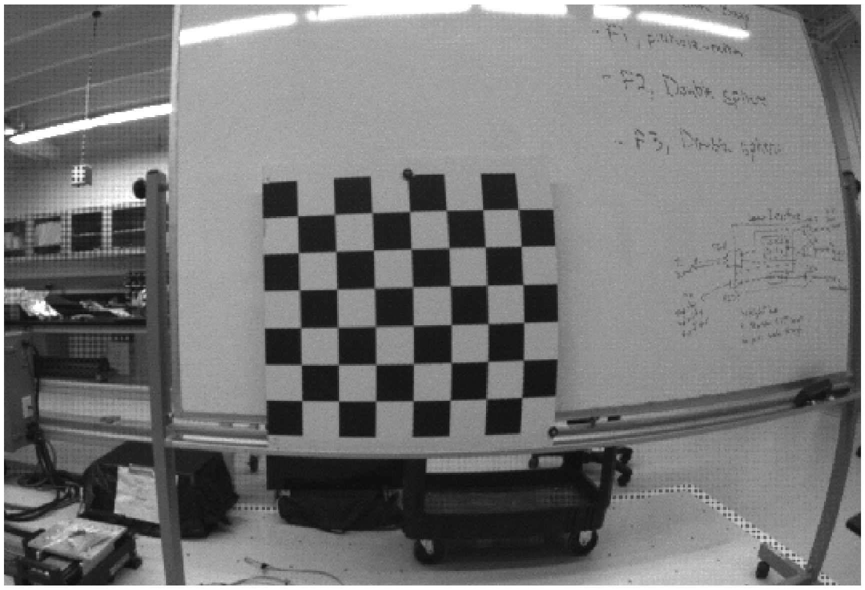


Figure 4.3: Checkerboard target for calibration

intrinsic parameters that minimize the reprojection error between the 3D points and their corresponding 2D measurements.

4.1.3 Extrinsic Calibration

Extrinsic calibration involves finding the transformation from one sensor's coordinate frame to another. Given that there are 5 sensors on IG handle, using the combination formula we get 10 possible extrinsic transforms we would have to calibrate for. However, taking advantage of the associative property of 3D transformation matrices we can reduce this number drastically. Using this property we perform 4 calibration processes: 1) calibration between the two front facing cameras (F1 and F3), 2) calibration between one front-facing camera (F1) and the IMU, 3) calibration between the rear-facing camera (F2) and the IMU and 4) calibration between the LiDAR and a front facing camera. This calibration process produces the tree of transforms displayed in 4.4, which can be used to find the transform between any sensor in the tree by chaining multiple transforms.

The Kalibr calibration tool [18] is used for the camera-to-camera calibration (T_{F1}^{F3}) as well as the camera-IMU calibrations (T_{F1}^{IMU} and T_{F2}^{IMU}). The camera-LiDAR calibration was performed using the Matlab Lidar Camera Calibrator [69], which uses methods proposed in [2, 66]. The process involves collecting data from the camera and from the LiDAR

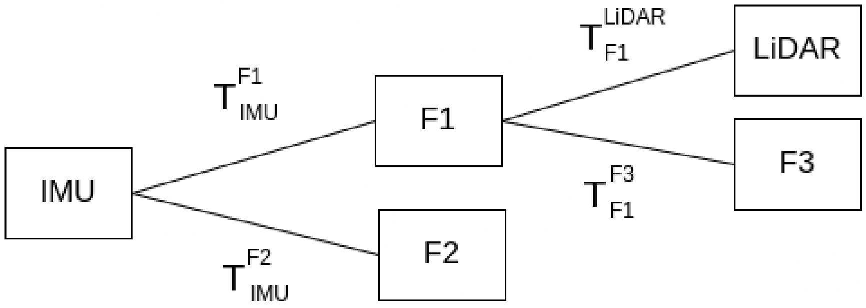


Figure 4.4: Tree of extrinsic calibrations.

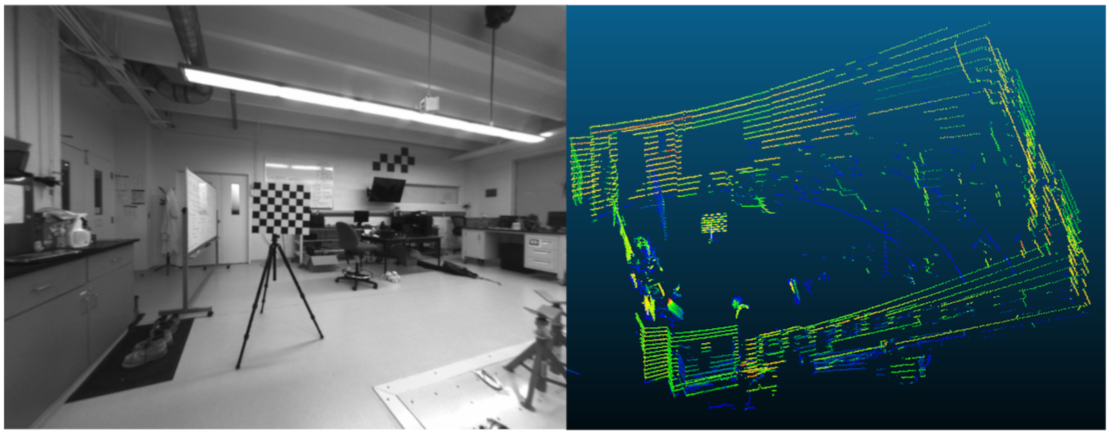


Figure 4.5: Associated image and LiDAR scan of a checkerboard target.

where both sensors have a checkerboard of known size within their field of view, as shown in figure 4.5. The checkerboard target is first detected in both the LiDAR scan and the camera image. The LiDAR will detect all the 3D points that are part of the checkerboard target with respect to the LiDAR's frame. Utilzing the known camera intrinsics, and dimensions of the checkerboard, the 3D locations of the points and lines of the checkerboard with respect to the camera origin can be estimated. Then using a scan registration approach 2.3.1, the relative pose between the camera and LiDAR is estimated.

4.2 3D Mapping

The 3D mapping task involves using the LiDAR and inertial data collected by IG handle to estimate the trajectory of the device using SLAM. SLAM typically involves estimating the motion between subsequent sensor measurements (odometry), for example given two subsequent measurements scans at timestamps t_1 and t_2 we want to estimate the transformation matrix $T_{t_2}^{t_1}$. Given the motion between subsequent measurements SLAM also keeps track of the pose of the measurement with respect to a world frame. Suppose the first measurement at t_0 is initialized to be the world frame ($T_{t_0}^{W} = I$), each time a new measurement is received, the motion between it and the last measurement ($T_{t_1}^{t_0}$) is computed, the transform between the newest measurement and the world frame is also kept track of using equation 4.1. Once the initial estimate of the pose of the device is retrieved with respect to the world frame, it is typically refined by minimizing the error between it's measurements and the current state of the map. In the case of LiDAR this process would begin by computing the motion between subsequent scans using scan registration 2.3.1 to find the pose with respect to the world, then the pose would be further refined by registering the same scan against the current map. The final result of SLAM is the entire trajectory of the device, given as a list of transforms at every time instant as in equation 4.2.

$$T_{t_1}^{W} = T_{t_0}^{W} T_{t_1}^{t_0} \tag{4.1}$$

$$T = \left[T_{t_1}^{W}, T_{t_2}^{W}, \ldots, T_{t_N}^{W}\right] \tag{4.2}$$

The SLAM algorithm used in this work is SC-LIO-SAM [26] which extends the existing LIO-SAM [53] with the Scan Context [27] descriptor for loop closure detection. LIO-SAM [53] formulates the SLAM problem as a factor graph optimization problem (see Fig. 4.6). For details on factor graphs see A.2.

The factor graph proposed in the aforementioned paper contains 3 types of constraints; 1) A Lidar odometry factor, adapted from [63], 2) an IMU preintegration constraint, as introduced in [17] and 3) a loop closure constraint. The vanilla version of LIO-SAM utilizes a euclidean distance based approach for identifying loop closures. Essentially, when the current position of the device is within a certain radius of a previous pose as loop closure operation is attempted with the previous pose. However SC-LIO-SAM replaces this euclidean based approach with the ScanContext [27] descriptor, by continuously searching for the most similar ScanContext descriptor from the map, and performing loop closure with any scans that are within a threshold of similarity. To allow for real-time operation, SC-LIO-SAM only performs a pose graph optimization on LiDAR keyframes (KF), which are a subset of the LiDAR scans taken at a lower rate, however, the poses of every LiDAR scan are still stored as a LiDAR sub-keyframe (SKF) and with respect to its nearest keyframe ($T^{KF}_{SKF_i}$).

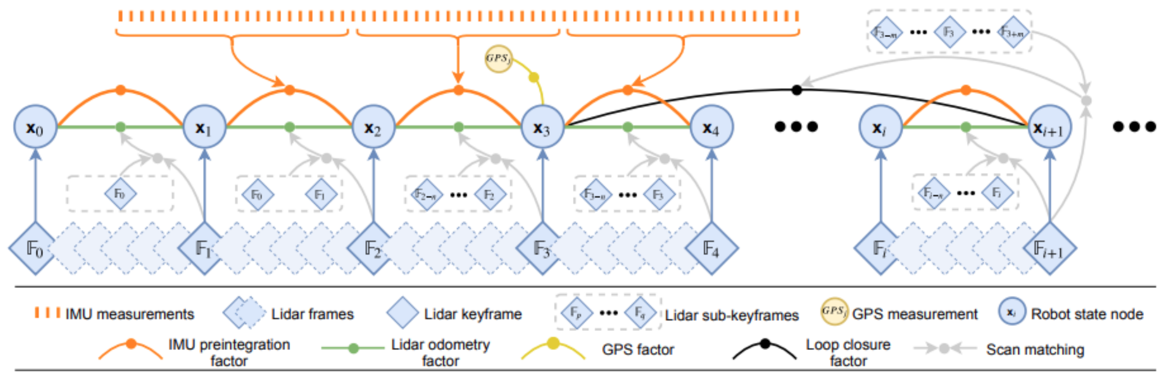


Figure 4.6: Structure of LIO-SAM, taken from [53]

Building the 3D map provided the trajectory from SLAM is as simple as transforming every LiDAR scan from its own reference to the world frame using its pose in the trajectory. However, most SLAM systems will output two trajectories, the first is the result of just odometry which exhibits significant drift, but provides a pose estimate for every sensor measurement; the second is the optimized trajectory which includes loop closures and therefore has less drift; however only estimates for a subset of the sensor measurements are provided. Algorithm 2 in [58] proposes a method to unify both of these trajectories to generate a drift-free trajectory with estimates for every sensor measurement. This work utilizes this algorithm to unify the two trajectories provided by SLAM before generating 3D maps.

4.3 Map Alignment

The proposed map alignment algorithm can be split into 4 primary steps; 1) Visual place recognition (VPR) candidate search, 2) LiDAR place recognition (LPR) outlier rejection, 3) Map to Map scan registration, 4) Non-rigid trajectory alignment. This section will outline each step independently, with defined inputs/outputs between steps, the full pipeline is

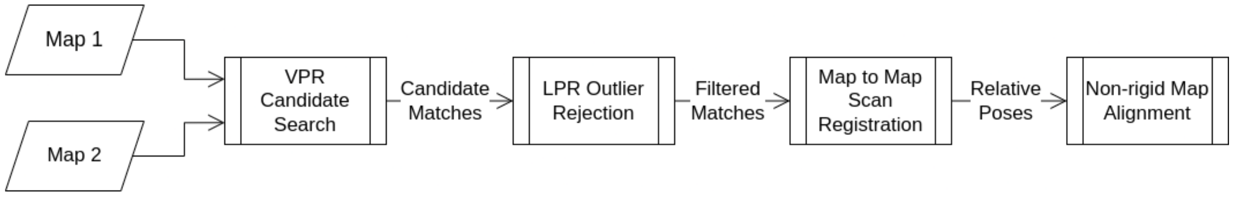


Figure 4.7: Full Map Alignment Pipeline

illustrated in figure 4.7. The pipeline has been designed with modularity in mind, allowing each stage to be easily replaced with a different method (i.e. replacing VPR, LPR and scan registration methods). Each map input in this system contains 3 sets of data: images, point clouds and trajectory, each queryable by timestamp. The following section will make use of the following conventions to refer to the data in each map:

- M_n : Map n
- $I^i_{M_n}$: Image from M_n at timestamp i
- $S^i_{M_n}$: Point cloud from M_n at timestamp i
- $T^{W_n}_{L_i}$: Transform from LiDAR frame at timestamp i to M_n's world frame
- $[t^0_n, t^1_n, \ldots, t^m_n]$: Set of timestamps from M_n

4.3.1 Visual Place Recognition

The goal of the VPR candidate search is to perform an initial search between the images in each map and propose a list of candidate matches. The candidate matches take the form of a list of timestamp pairs, from M_1 to M_2. The overall pipeline of this step is shown in figure 4.8.

Instead of adding $I^i_{M_1}\ \forall i \in [0, 1, \ldots, m]$ to the image database, only images that are spatially separated are used to reduce the number of redundant images in the database. To achieve this we utilize the input trajectory of the first map and only add images to the database if they have moved over 2 metres, or moved over 45° since the last image that has been added. This ensures that there has been a significant visual change since the last image was added to the database, to reduce redundancy.

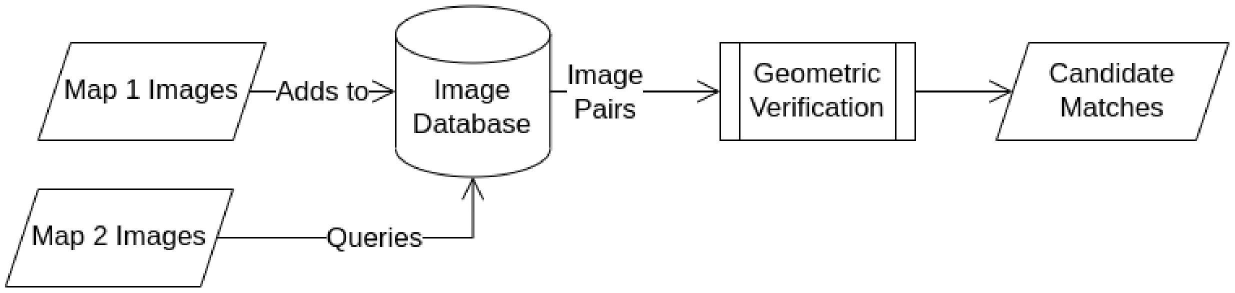


Figure 4.8: Visual Place Recognition Module.

To store and query images the method proposed in [19] is used. This method involves transforming each image into a histogram of visual words. This involves computing ORB [48] descriptors for each point that is tracked using the aforementioned feature tracker, and assigning each descriptor to a word in the visual vocabulary, which will then be accumulated in its histogram of words and added to the database. To make the system more robust, rather than using a pre-trained vocabulary such as the one used in [41], which is trained on the RAWSEEDS [7] dataset, we directly train a custom vocabulary on the features extracted from the images in M_1.

Once the image database is created for the M_1, lets call it D_{M_1}, we query the database with $I^i_{M_2}$ $\forall j \in [0, 1, \ldots, m]$. The query will return the image from the database that has the most similar histogram of words, as well as the computed similarity score, $s(v^i_{M_1}, v^j_{M_2})$, where $v^i_{M_n}$ is the histogram of words computed for $I^i_{M_n}$. If this score is greater than a preset threshold, α, it is considered for geometric verification. The geometric verification process first involves directly matching visual features between the image pair and computing the number of inlier feature matches. The ORB descriptors in each image are directly matched using the brute force feature matching method from OpenCV. Matching descriptors across images involves finding the descriptor from the second image that is most similar to the query descriptor using a distance function specialized to the descriptor used, in the ORB [48] descriptor case the distance metric is the Hamming distance [21]. Once feature matches have been computed, the essential matrix [23], between the images is estimated using a combination of Random Sample and Consensus (RANSAC) and the 7-point algorithm [65]. The 7-point algorithm uses 7 pixel correspondences to estimate the essential matrix between two images, which can be decomposed into the rotation and relative translation (not to scale) between images using singular value decomposition. Using this relative transformation, all feature matches can be triangulated 2.2.3, and reprojected to determine whether the match is an inlier or not (an inlier match would reproject to within a distance threshold, β, of its pixel measurement). To find the best possible relative transformation

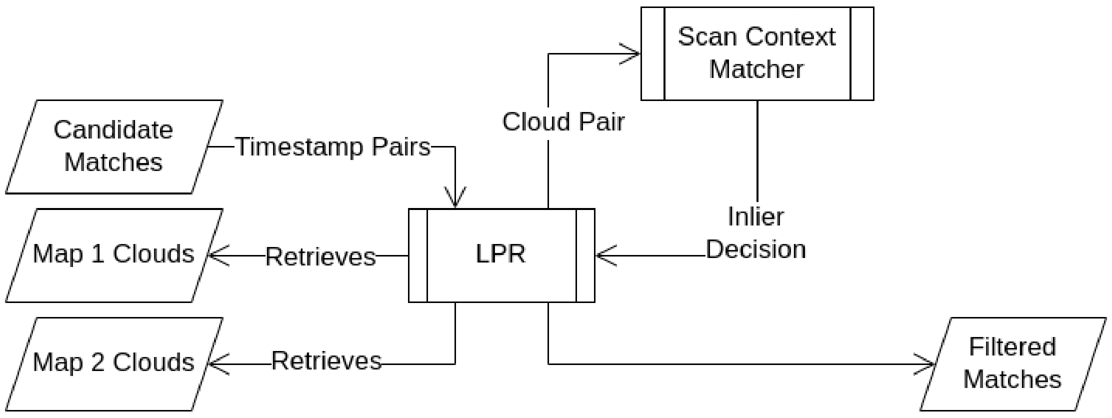


Figure 4.9: LiDAR Place Recognition Module.

between images the 7-point algorithm is used within a RANSAC framework, where for N iterations, 7 pixel correspondences will be randomly sampled to estimate a relative transform between images, which will be used to compute the number of inliers on the entire set of pixel correspondences (consensus). The maximum number of inliers throughout the RANSAC process is used to determine whether the image pair has passed the geometric verification, for a candidate match to be accepted, the total number of inliers must be more than or equal to ϕ of the total number of matches between the image pair.

If the image pair $(I^i_{M_1}, I^j_{M_2})$ passes both the BoW similarity check and the geometric verification, their associated timestamps, (t^i_1, t^j_2) are added to the list of candidate matches, the output of this step is shown in equation 4.3.

$$C^{M_2}_{M_1} = \Big[(t^{i1}_1, t^{j1}_2), (t^{i2}_1, t^{j2}_2), \ldots, (t^{in}_1, t^{jn}_2)\Big] \tag{4.3}$$

4.3.2 LiDAR Place Recognition

The LPR module is independent of the VPR module and takes only as input the list of candidate matches that was computed by VPR. The LPR module aims to further validate the candidate match set by computing the similarity of the surrounding point clouds to the candidate matches using a LiDAR descriptor, this work uses the ScanContext [27] descriptor, however due to the modularity of the proposed system, any LiDAR place recognition method can be used instead [13, 60]. The flow diagram of this module is illustrated in figure 4.9.

The first step of the LPR module is to retrieve the associated point clouds for each candidate match. Although the point cloud data and image data share the same reference time frame, the sensors collect data at different rates and not at the same time, therefore to retrieve the associated point cloud to a timestamp from a candidate match, we find the point cloud with the closest timestamp instead. The assumption here is that both the LiDAR and camera sensors collect data at a high enough rate that the difference between the timestamps will be negligible. Due to the VLP16's relative sparsity in single scans (only 16 vertical beams), the proposed algorithm retrieves the closest point cloud to the candidate timestamp, as well as the R clouds preceding and proceeding the timestamp. These $2R + 1$ clouds then get transformed into the "middle" clouds frame using the maps trajectory. Assume we have a timestamp from a candidate match pair t_1^i, we then find the closest point cloud from M_1, $S_{M_1}^{i'}$, where i' is the closest timestamp to i, as well as the R surrounding clouds $S_{M_1}^{i' \pm r}$. To transform each cloud into the middle clouds frame we use equation 4.4.

$$(S_{M_1}^{i' \pm r})\prime = (T_{L_{i'}}^{W_n})^{-1} \, T_{L_{i' \pm r}}^{W_n} \, S_{M_1}^{i' \pm r} \tag{4.4}$$

After each cloud is transformed into the middle cloud frame, they are all aggregated into a single point cloud scan. This process is repeated for both timestamps in a candidate match. The next step is to compute the ScanContext [27] descriptor for each aggregated scan. There are two distances that can be computed from ScanContext descriptors, the first distance is computed by summing the distances between the corresponding columns. However since the ScanContext descriptor is not viewpoint invariant this distance is only useful when there is little viewpoint change between the point clouds being compared. However, in this case there can be a drastic viewpoint change between point clouds, so we use the slower, but more robust distance metric, which calculates the minimal distance from every possible column-shift. Figure 4.10 shows an example of two scan context descriptors that are of the same area, but are from different viewpoints. In this case the traditional distance metric would result in many false negatives. If a candidate matches ScanContext distance below a set threshold, ψ, then the candidate match is accepted. This module then outputs a filtered list of matches in the same format as equation 4.3.

4.3.3 Map-to-Map Scan Registration

After the candidate matches have been filtered with LPR, the next step is to estimate the relative poses between the maps at the candidate match timestamps. We first get a coarse

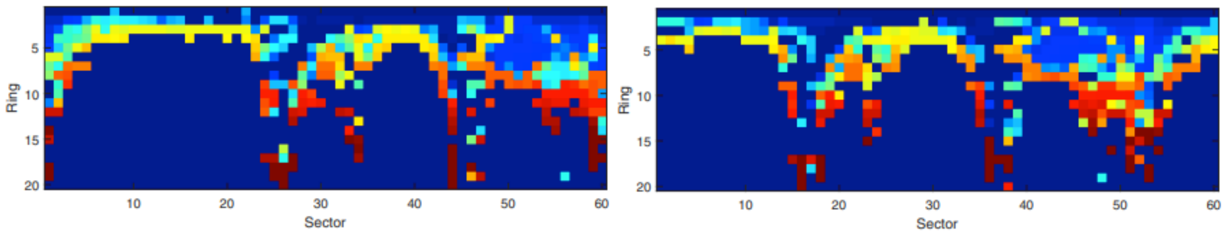


Figure 4.10: Column shifted Scan Context match, taken from [27].

estimate of the transform $T_{W_2}^{W_1}$ by using the transforms for each lidar scan in their world frames, $T_{L_i}^{W_1}$ and $T_{L_j}^{W_2}$ using equation 4.5. This initial estimate is a coarse estimate since the matches provided by VPR and LPR are not necessarily of the exact same location.

$$\widehat{T_{W_2}^{W_1}} = T_{L_i}^{W_1} * (T_{L_j}^{W_2})^{-1} \tag{4.5}$$

Then the same process outlined in 4.3.2 is used to retrieve an aggregate cloud around each timestamp, however each cloud is transformed into their respective world frames W^1 and W^2. Scan registration is then performed to refine the initial coarse estimate given in 4.5, resulting in an "update" transform T_R, using these aggregate scans. The final estimate of $T_{W_2}^{W_1}$ is computed using equation 4.6.

$$T_{W_2}^{W_1} = T_R * \widehat{T_{W_2}^{W_1}} \tag{4.6}$$

The scan registration algorithm used is known as Generalized-ICP (GICP) [52], which is an extension of the original ICP algorithm 2.3.1, however instead of minimizing just point to point error, GICP minimizes the error between the point in the source cloud to the closest computed plane in the target cloud, this is known as point-to-plane error. GICP is known to be more robust to outliers than regular ICP, making it very suitable for this application. If the resulting fitness score from the scan registration is larger than a threshold ξ, the match is discarded. For each match the best registration result is kept. The goal of this step is to store an estimate of the relative transform between maps for every candidate match, thus computing a "trajectory" of relative transforms where each timestamp in M_2 has an associated transform $T_{W_2}^{W_1}$. This process is outlined in algorithm 1.

```
Data: $M_1$, $M_2$, $(t_1^i, t_2^j)$, relativeTrajectory, relativeScores
$S_{M_1}^{i'\pm 3} = T_{L_{i'}}^{W_1} \times$ retrieveAggregateScan($M_1$, $t_1^i$, 3);
$S_{M_2}^{j'\pm 3} = T_{L_{j'}}^{W_2} \times$ retrieveAggregateScan($M_2$, $t_2^j$, 3);
$(T_{W_2}^{W_1}, fitness) =$ GICP($S_{M_1}^{i'\pm 3}$, $S_{M_2}^{j'\pm 3}$);
if $fitness \leqslant \rho \wedge relativeScores\left[t_2^j\right] \geqslant fitness$ then
    relativeTrajectory[$t_2^j$] $= T_{W_2}^{W_1}$;
    relativeScores[$t_2^j$] $= fitness$;
end
```

Algorithm 1: Map-to-Map Scan Registration.

4.3.4 Non-Rigid Map Alignment

The final step of the map alignment process is to apply the relative trajectory estimate computed in 4.3.3 to the trajectory in M_2 to get it in the world frame of M_1. However SLAM, even with loop closures, can still display drift over time, therefore the estimated relative trajectory will not be constant over time. Knowing this we aim to apply the relative trajectory adaptively over M_2's trajectory resulting in a "Non-Rigid Alignment". This is one of the reasons that scan registration isn't run on the full point cloud from each map as this would produce a rigid alignment between maps which can result in some areas of the map being more misaligned than others. This difference is illustrated in figure 4.11.

To perform the non-rigid alignment, we determine the associated map to map transform, $T_{W_2}^{W_1}$, for every pose in M_2's trajectory by interpolating from the relative trajectory. To interpolate over the **SE3**, a B-Spline is estimated for the estimated relative trajectory. Details of the derivations for the B-Spline used are outlined in [38]. For each pose in the original trajectory, the interpolated relative pose is applied to the original pose in M_2's trajectory to transform it into the W_1 frame. The full algorithm is laid out in 2. The B-Spline also has the added benefit of smoothing the resulting trajectory, and reducing the effect of outliers.

4.4 Change Detection

After accurate map alignment is accomplished various inspection tasks taking advantage of a unified reference frame can be performed. To validate and illustrate the method proposed

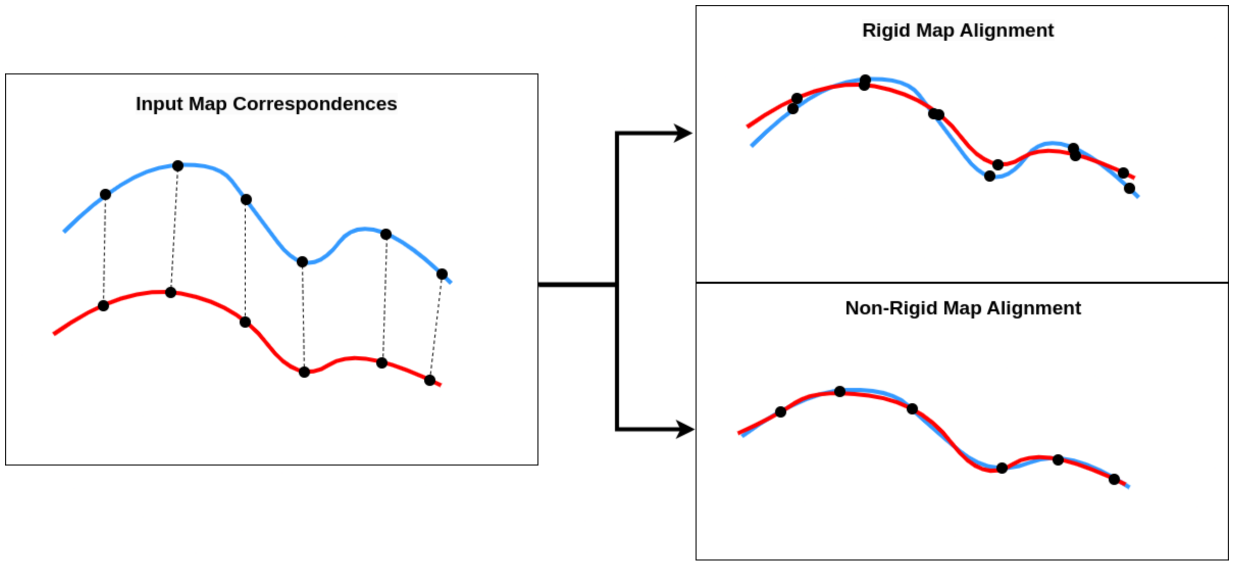


Figure 4.11: Non-Rigid Map Alignment.

```
**Data:** $M_2$, relativeTrajectory
**Result:** alignedTrajectory
$spline = \textbf{estimateSpline}(relativeTrajectory)$;
**for** $T^{W_2}_{L_i} \in M_2$ **do**
    $T^{W_1}_{W_2} = \textbf{getInterpolatedPose}(spline, t^i_2)$;
    alignedTrajectory$[t^i_2] = T^{W_1}_{W_2} T^{W_2}_{L_i}$;
**end**
**return** alignedTrajectory;
```

Algorithm 2: Non-Rigid Map Alignment.

here we will utilize a simple change detection scheme to highlight the parts of the map that have changed between two or more scans. Change detection can be performed in a similar manner to the second step of the ICP 2.3.1 algorithm, which is to estimate correspondences between the two clouds for each point, however instead of minimizing that difference in an optimization process, a threshold is used to identify if a point should be labelled as a changed point. For example if a point p_1 in M_1 has no point in M_2 that is less than a distance τ away then it is labelled as a changed point.

However simple change detection is not the only operation that can be performed using well aligned point cloud maps. The automated concrete defect inspection methods proposed in [35] can be utilized in conjunction with this method to track the growth of defects over time. In fact with aligned maps, association across time becomes a trivial task, requiring only a nearest neighbour search for the same region of interest in the euclidean space, this is illustrated in figure 4.12. For example, attempting to associate the green objects between maps in figure 4.12 before map alignment is non-trivial due to the need to search through each object in the other map to find the most similar one, however after map alignment the association becomes trivial.

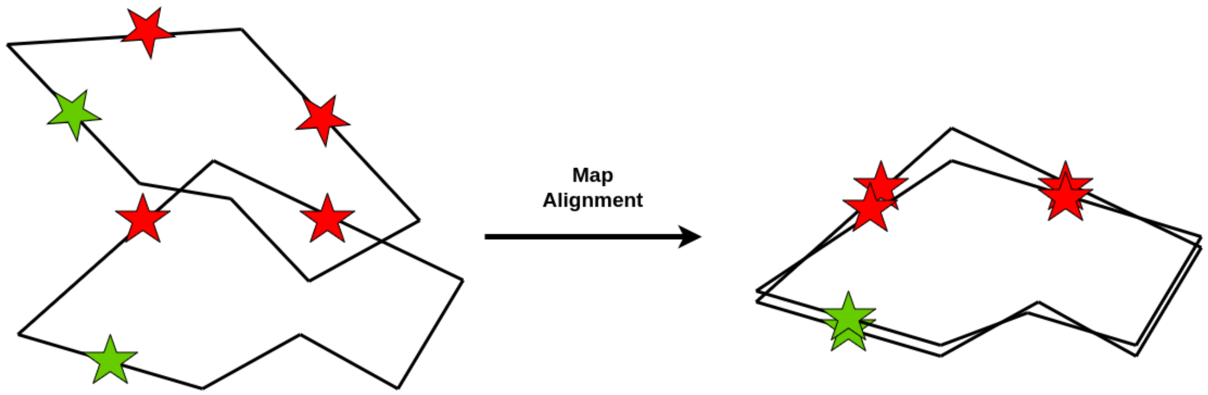


Figure 4.12: Map alignment for association over time.

Chapter 5

Results

5.1 Data Sets

In the experimental evaluation, two distinct datasets were employed, which were gathered using the device outlined in 4.1.1. The first dataset was gathered from the Structures Laboratory at the University of Waterloo, spanning four unique acquisition instances. Each acquisition was spaced with varying temporal intervals to test the robustness of our approach across different time spans. This Structures Laboratory dataset is an example of an indoor environment, rich in visual and lidar data. The second dataset was gathered from the Conestogo Bridge in Kitchener, Ontario. Comprising of three scans, each at distinct times, this dataset serves as an outdoor example, markedly sparse in visual and lidar data, making it a more challenging test case for validation.

5.1.1 Structures Lab

The Structures Lab dataset contains 4 different scanning sessions at 4 different times, the details of the data are as follows:

1. September 30, 2021, 1:11 PM, 183 Images, 929 Point Cloud Scans
2. September 30, 2021, 1:14 PM, 152 Images, 775 Point Cloud Scans
3. October 21, 2021, 12:32 PM, 1479 Images, 753 Point Cloud Scans

4. October 27, 2021, 2:15 PM, 1909 Images, 967 Point Cloud Scans

Each scanning session resulted in a map using the method explained in 4.2; Figure 5.2 shows an example map of the Structures Lab. Figure 5.1 shows some images collected from the Structures Lab dataset.

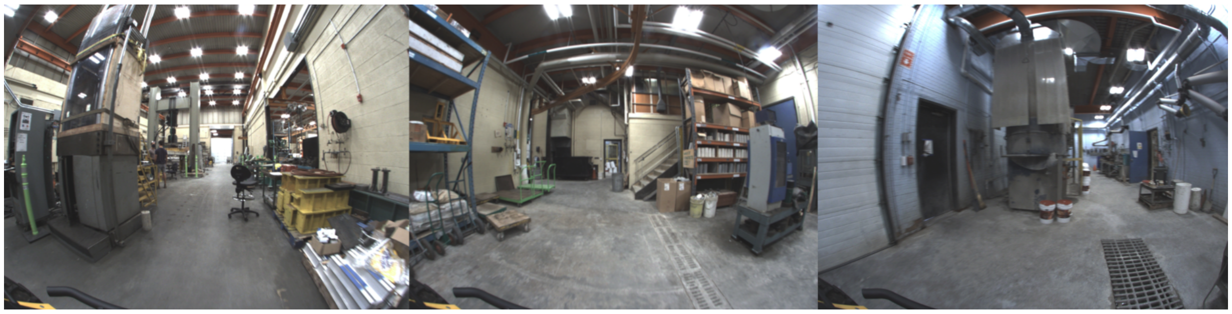


Figure 5.1: Images from SL_1.

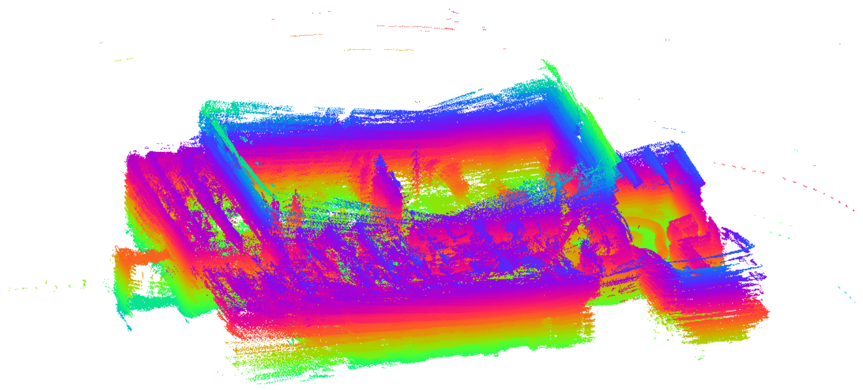


Figure 5.2: Point Cloud Map of SL_1.

5.1.2 Conestogo Bridge

The Conestogo Bridge dataset similarly contains 3 different scanning sessions collected at 3 different times. The details of the data collected are outlined:

1. October 7, 2021, 1:51 PM, 1680 Images, 850 Point Cloud Scans

2. October 27, 2021, 4:21 PM, 4564 Images, 2285 Point Cloud Scans
3. October 27, 2021, 4:33 PM, 3615 Images, 1813 Point Cloud Scans

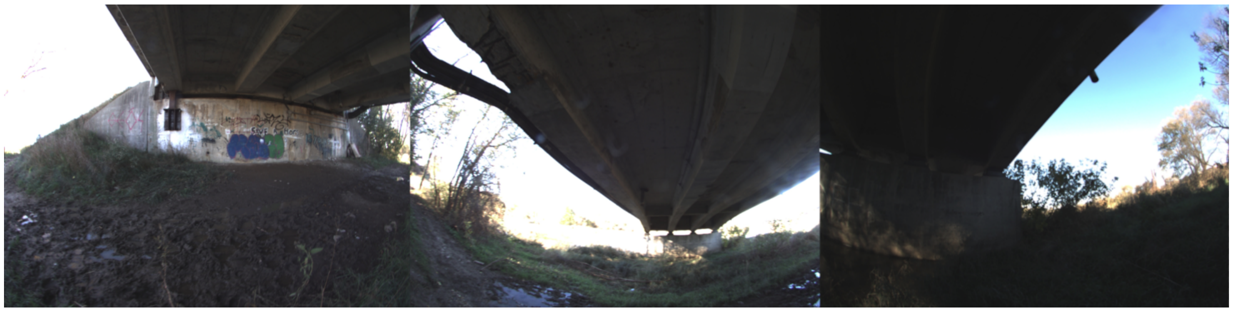


Figure 5.3: Images from CB_2.

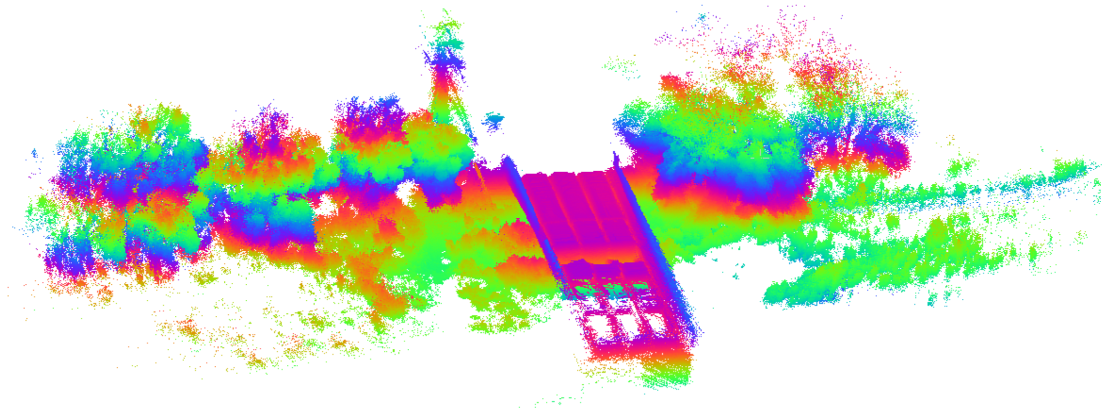


Figure 5.4: Point Cloud Map of CB_2.

Figure 5.3 shows sample images from the Conestogo Bridge dataset, demonstrating the relative sparsity of the visual information available when performing outdoor inspections, making it challenging for mapping and inspection.

5.2 Map Alignment

For the following tests, we will refer to the i'th scan from the Structures Lab dataset as SL_i, and the Conestogo Bridge dataset as CB_i. And we will refer to the result of an alignment between the i'th and j'th scan as DS_i-DS_j, where DS is the specific dataset.

5.2.1 Optimal Parameter Determination

The first tests performed to validate and determine the best parameter values were between the datasets SL_4-SL_1. To evaluate the accuracy of each parameter combination, we compute the point cloud error on the resulting map alignment result. The error computed is typically called the point cloud "fitness" and is the root mean-squared error (RMSE) of the distances between point correspondences. We aim to determine the optimal parameters for minimum visual similarity score, α, minimum geometric inlier percentage, ϕ, maximum lidar scan context distance, ψ, maximum lidar fitness score, ξ, and aggregate cloud radius, R.

Test 1 determines the optimal R and ξ values by holding all other parameters constant ($\alpha = 0.5$, $\phi = 0.5$, $\psi = 0.15$). The results of this test are in Table 5.1. Values with N/A, are tests where not enough matches were found to be able to estimate a valid relative trajectory.

Table 5.1: Aggregation Radius and Scan Registration Fitness Ablation

	R=6	R=9	R=18
$\xi = 0.5$	N/A	0.205	0.169
$\xi = 0.75$	0.169	0.151	0.148
$\xi = 1.0$	0.154	0.162	0.143
$\xi = 2.0$	0.172	0.174	0.153

We see in Table 5.1, the optimal values are $R = 18$ and $\xi = 1.0$. For our second test we determine the optimal values for α and ϕ and hold all other parameters constant ($R = 18$, $\xi = 1.0$, $\psi = 0.15$).

Test 2 shows that if α is too high, we remove too many potential matches and can't estimate a relative trajectory. It also shows that the optimal value of ϕ is in the middle of the low and high end. We choose $\phi = 0.65$ and $\alpha = 0.5$, even though they don't

Table 5.2: Geometric Inlier Ratio and Visual Similarity Score Ablation

	$\phi = 0.5$	$\phi = 0.65$	$\phi = 0.8$
$\alpha = 0.2$	0.142	0.137	0.181
$\alpha = 0.5$	0.160	0.144	0.162
$\alpha = 0.8$	N/A	N/A	N/A

provide the lowest error, choosing too small of either can result in too many outliers and choosing too large can result in too few estimates. For example, if α is too low, then there is a higher chance of accepting dissimilar images, if it is too high, then there may not be enough candidate matches to continue the algorithm. However if ϕ is too low then there is a higher chance that a dissimilar image that passed the visual similarity check will be accepted, and if it is too high, then again there may not be enough candidate matches to continue. The last test aims to find the optimal value for the maximum Scan Context distance, ψ.

Table 5.3: Maximum Scan Context distance

$\psi = 0.15$	0.152
$\psi = 0.25$	0.170
$\psi = 0.4$	0.181

The last test shows that a stricter threshold on the maximum scan context distance provides better results; hence, we choose $\psi = 0.15$ for the subsequent tests. Although we have chosen these specific values for the datasets presented in this work which generalize well to each dataset, the results can be further optimized by reproducing the method above for each dataset.

5.2.2 Outlier Detection

Lastly, to validate that using a combination of both visual and lidar based place recognition techniques enhances the map alignment, we compare the accuracy when using just Scan Context (SC) or the geometric visual inlier detection (Geo), or to both. Additionally, we turn off the maximum point cloud fitness threshold (ξ) to isolate the effect of the outlier

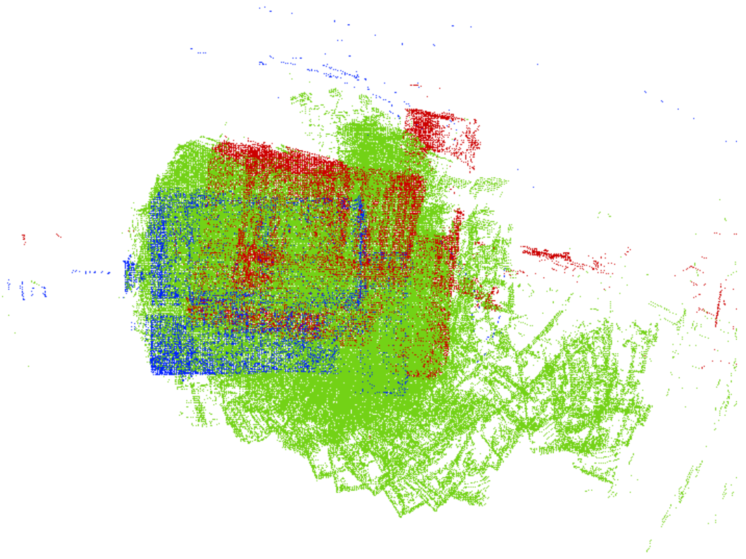


Figure 5.5: Map Alignment without visual or SC outlier rejection (blue: SL_1, red: SL_2, green: Aligned SL_2).

detection methods. Without either SC or Geo, the result is unusable, as visualized in Figure 5.5. The results in Table 5.4 show that while both outlier rejection methods provide good results on their own, there is an improvement when using them together as proposed in this thesis.

Table 5.4: Outlier Detection Methods

Geo	SC	SC + Geo
0.171	0.164	0.143

5.2.3 Dataset Results

Once the optimal parameters are determined, we perform map alignment on each dataset (SL and CB), by aligning each scan to the first scan in the dataset. We evaluate the results by comparing the point cloud fitness score that results from using GICP on the full map.

Structures Lab

Table 5.5 shows the results of map alignment on the SL dataset compared to GICP. The results are visualized in Fig. 5.6 and 5.7, where the maps of SL_1, SL_2, SL_3, SL_4 are represented as blue, green, yellow and red point clouds respectively. We see from the results that our approach outperforms GICP by a small margin, this is due to the relatively small initial offset of the input maps and the fact that each map, globally, is relatively similar. However, we still see an improvement on map alignment quality using our approach, which is due to the non-rigid map alignment method used.

Table 5.5: Structures Lab Map Alignment Fitness

Dataset	GICP (m)	Our Approach (m)
SL_2-SL_1	0.176	0.128
SL_3-SL_1	0.182	0.179
SL_4-SL_1	0.177	0.143

Conestogo Bridge

Table 5.6 shows the results of map alignment on the CB dataset compared to GICP. The results are visualized in Fig. 5.8 and 5.9, where the maps of CB_1, CB_2, CB_3 are represented as green, blue, and red point clouds respectively. This dataset tells a different story from the SL dataset. We see a drastic improvement on map alignment quality over GICP due to the initial poor alignments of the maps, making GICP harder to converge to an accurate estimate. Moreover, the different maps are very different in size and exhibit much more challenging features to use for scan matching (i.e. trees and bushes). Figure 5.10 shows the results of GICP vs our approach where we observe that GICP is unable to converge and cannot be used to perform robust map alignment.

5.3 Change Detection

The change detection approach used to validate our method was a simple radius search for each point in the cloud. That is, for every point in the first cloud, we check to see if any points in the second cloud are within a radius of this point, if there is not, then it

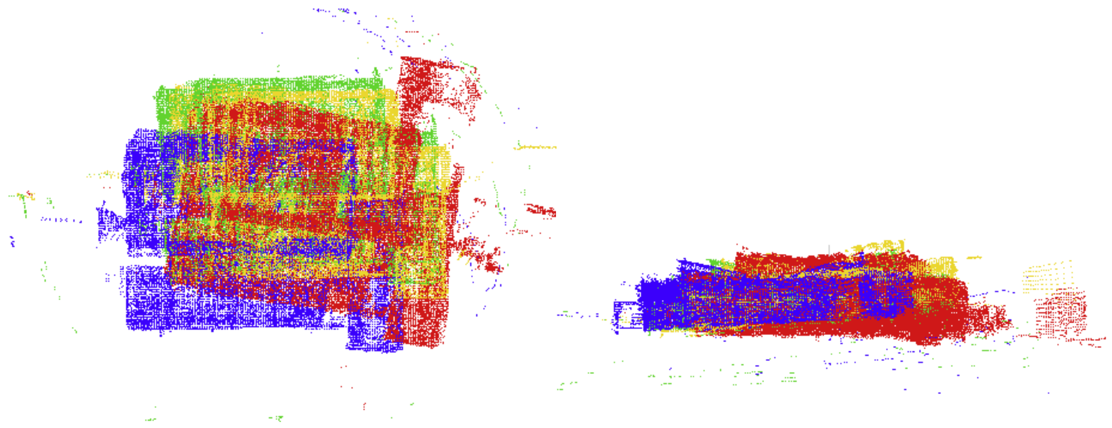


Figure 5.6: Structures Lab scans before map alignment (left: top view, right: side view).

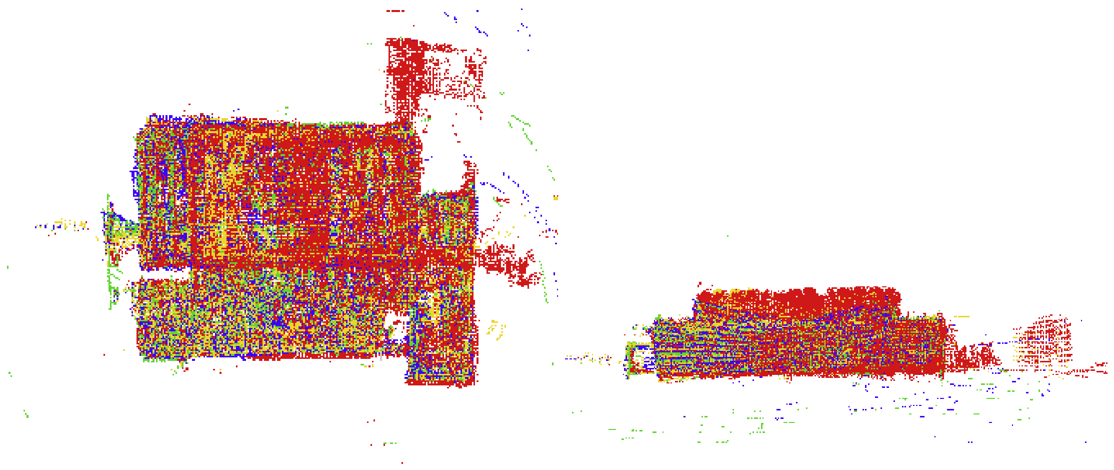


Figure 5.7: Structures Lab scans after map alignment (left: top view, right: side view).

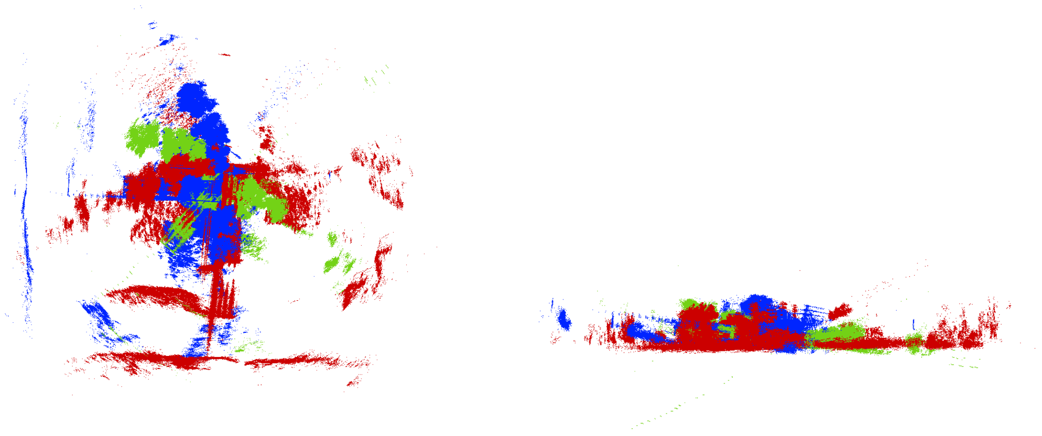


Figure 5.8: Conestogo Bridge scans before map alignment (left: top view, right: side view).

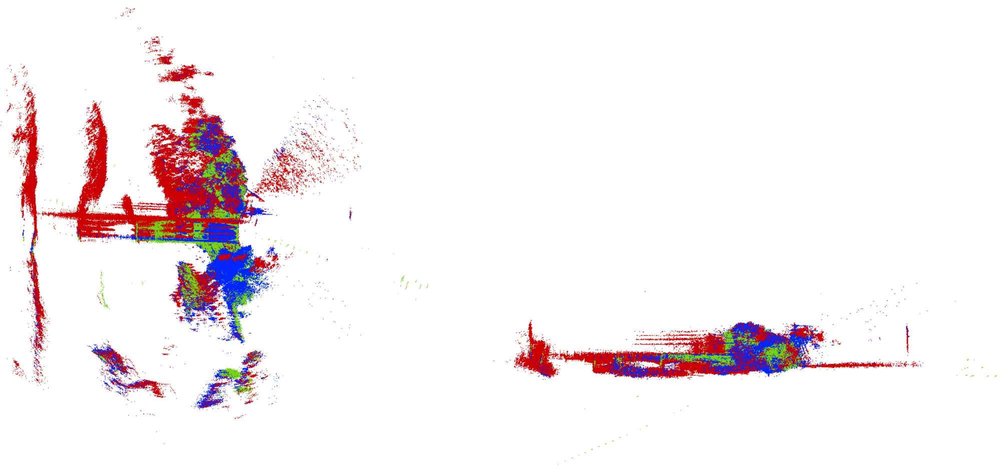


Figure 5.9: Conestogo Bridge scans after map alignment (left: top view, right: side view).

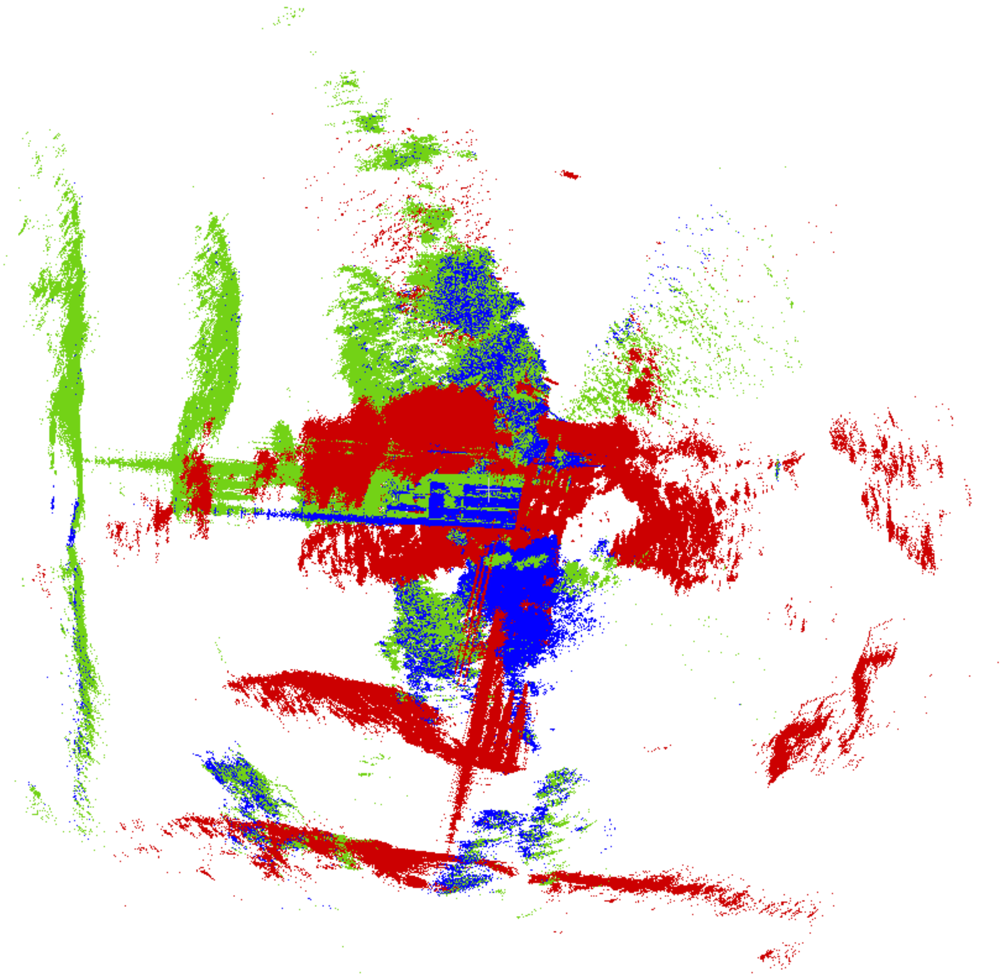


Figure 5.10: Our approach (green), vs GICP (red) aligned to CB_2 (blue).

Table 5.6: Conestogo Bridge Map Alignment Fitness

Dataset	GICP (m)	Our Approach (m)
CB_1-CB_2	2.00	0.705
CB_2-CB_3	1.30	0.249

is determined to be a change. While simple, this approach demonstrates the efficacy of using this map alignment approach for detecting changes in other manners, such as object detection, classification, and subsequent change reporting between objects over time (such as concrete defects). Figure 5.11 shows an example in the SL dataset of a given visual map pair, where can see a large change between with the introduction of a forklift in map 2. This change is shown in the point cloud map representing points determined to be changes from the first map in figure 5.12, circled in red.

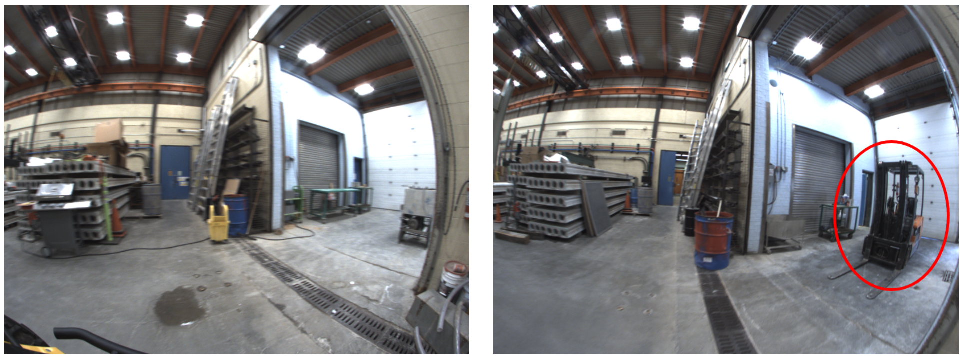


Figure 5.11: Visual match between map 1 (left) and map 2 (right).

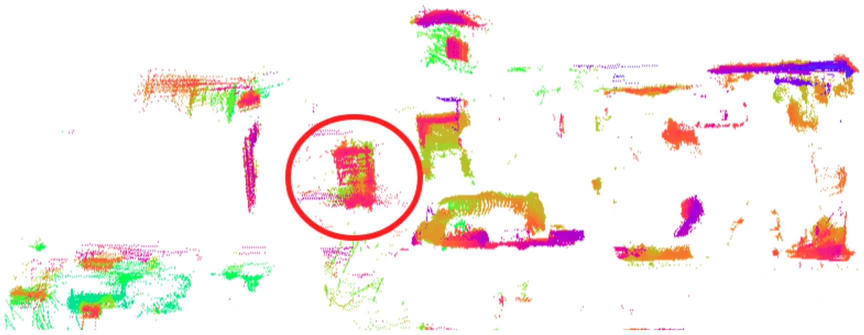


Figure 5.12: Corresponding point cloud map changes between map 1 and map 2.

Chapter 6

Conclusions

The work presented in this thesis aims to address the existing gap in multi-session mapping when applied to infrastructure inspection automation. The presented work accomplishes this by presenting an offline, SLAM agnostic, approach to map alignment for change detection between scans. This allows the use of new state-of-the-art SLAM methods, even if they do not have robust multi-session capability built in.

In this thesis, we found that our approach outperforms the GICP method in terms of the resulting point cloud error for point cloud map alignment (even when given good initialization). Interestingly, when solely relying on visual place recognition without incorporating geometric outlier rejection from visual matches or outlier rejection from lidar-based descriptor matching, our method's performance deteriorated, rendering it ineffective and unusable. However, when we incorporated either visual geometric outlier detection or lidar outlier detection using lidar descriptors, there was a noticeable enhancement in performance. Nonetheless, the best results were achieved when both detection methods were utilized in conjunction.

By demonstrating that map alignment can be executed offline, the concern of choosing a specific SLAM system due to its incorporation of multi-session capabilities is alleviated. Furthermore, this work is an innovative extension to the current multi-session mapping capabilities. While existing literature has primarily used visual or lidar-based place recognition for relocalization, this research combines the approaches, demonstrating that their integration can lead to enhanced accuracy in map alignment. This combined approach not only propels the domain forward but also fills a noticeable void in the existing body of knowledge, emphasizing the potential of leveraging multiple sensory modalities for advanced mapping solutions.

While the findings presented in this research carry significant implications for map alignment, it's important to acknowledge certain limitations that could impact broader applications. Firstly, the necessity to manually tune parameters for each dataset diminishes the generalizability of the method. Additionally, the offline nature of the process, while offering certain advantages, is inherently cumbersome and can be time-consuming. Another potential bottleneck arises from the prerequisite of having initially accurate maps of each area built from the chosen SLAM system. The system does not inherently rectify or account for poor trajectories. Thus, the quality of output is tied to the quality of input. Lastly, the robustness of these methods remains somewhat uncertain, having been tested on only two datasets, which limits the breadth of its validation in diverse scenarios.

Several directions for further research in the area of map alignment are revealed in light of the findings and the inherent limitations. First and foremost, incorporating the stated methodology into an online lidar-visual SLAM system may facilitate real-time, dynamic mapping solutions that could help to address some of the issues with the offline paradigm. Furthermore, it would useful to investigate more advanced approaches to visual and lidar place recognition, particularly emphasizing learning-based strategies. These techniques, which make use of deep learning or machine learning, might offer more flexible and reliable recognition capabilities. The impact of using this technology for static object association across several scans is another area that merits further exploration. Such a study would provide insights into the method's effectiveness in assuring accuracy and consistency when mapping static features in an environment throughout a range of sessions or periods. To overcome the constraints associated with manual parameter tuning, future work should explore the potential for automatic parameter optimization. Furthermore, delving into offline trajectory refinement may mitigate the need for precise prior trajectories to obtain reliable results. Lastly, evaluating the proposed algorithm across diverse open-source datasets and environments can provide a more comprehensive assessment of its effectiveness across a broader spectrum of scenarios.

In summary, this thesis has broken new ground in the domain of map alignment and discovered areas where there's room for refinement. Our approach, as shown in our results, out performs the GICP method in producing a more accurate point cloud for map alignment, underlining its potential. Yet, the study also demonstrated the relationship between various recognition and outlier detection mechanisms. While solely relying on visual place recognition was shown to be insufficient, the introduction of visual geometric outlier detection and lidar outlier detection using lidar descriptors increased robustness of the approach. However, with existing challenges like the need for manual parameter tuning, the offline nature of the process, and the requirement for accurate initial maps, there's an opportunity for further innovation and exploration. Future prospects of integrating this into online SLAM systems, exploring advanced place recognition techniques, and studying static object associations hint at this research's potential.

References

[1] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5297–5307, 2016.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

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

[4] Paul Besl and Neil McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.

[5] Peter Biber and Wolfgang Straber. The normal distributions transform: A new approach to laser scan matching. volume 3, pages 2743–2748. International Conference on Intelligent Robots and Systems (IROS), IEEE, 2003.

[6] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. Zürich, 2015. ETH-Zürich. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); Conference Location: Hamburg, Germany; Conference Date: September 28 - October 2, 2015.

[7] Andrea Bonarini, Wolfram Burgard, G Fontana, Matteo Matteucci, Domenico Sorrenti, and Juan Tardos. Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets. 01 2009.

[8] Russel Buchanan. Allan variance ros. https://github.com/ori-drs/allan_variance_ros, 2020.

[9] Michael Calonder, Vincent Lepetit, Christoph Strech, and Pascal Fua. Brief: Binary robust independent elementary features. European Conference on Computer Vision (ECCV), IEEE, 2010.

[10] Carlos Campos, Richard Elvira, Juan J. Gomez Rodgrigues, Jose M. M. Montiel, and Juan D. Tardos. Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam. volume 37, page 1874–1890. IEEE Transactions on Robotics, IEEE, 2021.

[11] Nicholas Charron, Evan McLaughlin, Stephen Phillips, Kevin Goorts, Sriram Narasimhan, and Steven L. Waslander. Automated bridge inspection using mobile ground robotics. *Journal of Structural Engineering*, 145:04019137, 2019.

[12] Andrei Cramariuc, Lukas Bernreiter, Florian Tschopp, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, and Cesar Cadena. maplab 2.0 – a modular and multi-modal mapping framework. In *IEEE Robotics and Automation Letters*, pages 520–527, 2023.

[13] Yunge Cui, Xieyuanli Chen, Yinlong Zhang, Jiahua Dong, Qingxiao Wu, and Feng Zhu. Bow3d: Bag of words for real-time loop closing in 3d lidar slam. In *IEEE Robotics and Automation Letters*, pages 2828–2835, 2023.

[14] Yunge Cui, Yinlong Zhang, Jiahua Dong, Haibo Sun, and Feng Zhu. Link3d: Linear keypoints representation for 3d lidar point cloud. *arXiv preprint arXiv:2206.05927*, 2022.

[15] Frank Dallaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. pages 337–33712. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2018.

[17] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[18] Paul Furgale, Hannes Sommer, Jerome Maye, Joern Rehder, Thomas Schneider, and Luc Oth. Kalibr. https://github.com/ethz-asl/kalibr, 2014.

[19] Dorian Galvez-López and Juan D. Tardos. Bags of binary words for fast place recognition in image sequences. volume 28, pages 1188–1197. IEEE Transactions on Robotics, IEEE, 2012.

[20] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:930–943, 2003.

[21] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[22] Chris Harris and Mike Stephens. A combined corner and edge detector. pages 147–151. Alvey Vision Conference, 1988.

[23] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[24] Joel Hesch and Stergios I. Roumeliotis. A direct least-squares (dls) method for pnp. International Conference on Computer Vision (ICCV), 2011.

[25] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[26] Giseop Kim. Sc-lio-sam. https://github.com/gisbi-kim/SC-LIO-SAM, 2021.

[27] Giseop Kim, Sunwook Choi, and Ayoung Kim. Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. In *IEEE Transactions on Robotics*, page 1856–1874, 2022.

[28] Giseop Kim and Ayoung Kim. Lt-mapper: A modular framework for lidar-based lifelong mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7995–8002, 2022.

[29] Hung M. La, Nenad Gucunski, Kristin Dana, and Seong-Hoon Kee. Development of an autonomous bridge deck inspection robotic system. *Journal of Field Robotics*, 34:1489–1504, 2017.

[30] Mathieu Labbe and Francois Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666, 2014.

[31] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81:155–166, 2009.

[32] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul T. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, pages 314–334, 2015.

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

[34] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. pages 121–130. Imaging Understanding Workshop, 1981.

[35] Evan McLaughlin, Nicholas Charron, and Sriram Narasimhan. Automated defect quantification in concrete bridges using robotics and deep learning. *Journal of Computing in Civil Engineering*, 34:04020029, 2020.

[36] Jalpa D. Mehta and S. G. Bhirud. Image stitching techniques. In S. J. Pise, editor, *Thinkquest˜2010*, pages 74–80, New Delhi, 2011. Springer India.

[37] Dominik Merkle, Annette Schmitt, and Alexander Reiterer. Concept of an autonomous mobile robotic system for bridge inspection. page 11535. SPIE Remote Sensing, SPIE, 2020.

[38] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440, 2018.

[39] Raul Mur-Artal, José M. M. Montiel, and Juan D. Tardos. Orb-slam: a versatile and accurate monocular slam system. In *IEEE Transactions on Robotics*, pages 1147–1163, 2015.

[40] Raul Mur-Artal and Juan D. Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. In *IEEE Transactions on Robotics*, pages 1255–1262, 2017.

[41] Raul Mur-Artal and Juan D. Tardós. Fast relocalisation and loop closing in keyframe-based slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853, 2014.

[42] Mikael Persson and Klas Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. European Conference on Computer Vision (ECCV), Springer, 2018.

[43] L.B. Pupo, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, and Universitat Politècnica de Catalunya. Departament de Teoria del Senyal i Comunicacions. *Characterization of Errors and Noises in MEMS Inertial Sensors Using Allan Variance Method*. Universitat Politècnica de Catalunya. Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, 2016.

[44] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[45] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. In *IEEE Transactions on Robotics*, pages 1004–1020, 2018.

[46] Maria Rashidi, Masou Mohammadi, Saba Sadeghlou Kivi, Mohammad Mehid Abdolvand, Truong-Hong Linh, and Bijan Samali. A decade of modern bridge monitoring using terrestrial laser scanning: Review and future directions. *Remote Sensing*, 12:3796, 2020.

[47] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, 2016.

[48] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. International Conference on Computer Vision (ICCV), IEEE, 2011.

[49] Samuela Salti, Frederico Tombardi, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description'. pages 356–369. European Conference on Computer Vision (ECCV), Springer, 2010.

[50] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart. maplab: An open framework for research in visual-inertial mapping and localization. In *IEEE Robotics and Automation Letters*, pages 1418–1425, 2018.

[51] Stefan Schubert, Peer Neubert, Sourav Garg, Michael Milford, and Tobias Fischer. Visual place recognition: A tutorial, 2023.

[52] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. Generalized-icp. 06 2009.

[53] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. IEEE International Conference on Intelligent Robots and Systems (IROS), page 5135–5142. IEEE, 2020.

[54] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. page 5692–5698. IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021.

[55] Ken Shoemake. Animating rotation with quaternion curves. SIGGRAPH '85, page 245–254, New York, NY, USA, 1985. Association for Computing Machinery.

[56] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. In *IEEE Robotics and Automation Letters*, pages 965–972, 2018.

[57] Pingbo Tang, Burcu Akinci, and James H. Garrett. Laser scanning for bridge inspection and management. pages 206–207. IABSE, 2007.

[58] Alexander Thoms, Gabriel Earle, Nicholas Charron, and Sriram Narasimhan. Tightly coupled, graph-based dvl/imu fusion and decoupled mapping for slam-centric maritime infrastructure inspection. *IEEE Journal of Oceanic Engineering*, 48(3):663–676, 2023.

[59] Valyslav Usenko, Nikolaus Demmel, and Daniel Cremmers. The double sphere camera model. pages 552–560. International Conference on 3D Vision (3DV), IEEE, 2018.

[60] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4470–4479, 2018.

[61] Daobin Wang, Huawei Liang, Tao Mei, Hui Zhu, Jing Fu, and Xiang Tao. Lidar scan matching ekf-slam using the differential model of vehicle motion. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 908–912, 2013.

[62] Tian Xia, Jian Yang, and Long Chen. Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning. *Automation in Construction*, 133:103992, 2022.

[63] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. Robotics: Science and Systems X, Robotics: Science and Systems Foundation, 2014.

[64] Shishun Zhang, Longyu Zheng, and Tao Wenbing. Survey and evaluation of rgb-d slam. *IEEE Access*, 9:21367–21387, 2021.

[65] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27:161–195, 03 1998.

[66] Lipu Zhou, Zimo Li, and Michael Kaess. Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5562–5569, 2018.

[67] How to choose the right lidar sensor for your project. `https://store.clearpathrobotics.com/blogs/blog/how-to-choose-a-lidar`. Accessed: 2023-10-05.

[68] Ultra puck. `https://velodynelidar.com/products/ultra-puck/`. Accessed: 2023-10-05.

[69] Matlab lidar camera calibrator. `https://www.mathworks.com/help/lidar/ref/lidarcameracalibrator-app.html`, 2021.

[70] Matlab monocular visual odometry. `https://www.mathworks.com/help/vision/ug/monocular-visual-odometry.html`, 2023.

[71] Matlab structure from motion. `https://www.mathworks.com/help/vision/ug/structure-from-motion.html`, 2023.

APPENDICES

Appendix A

SLAM Mathematical Foundations

A.1 Recursive State Estimation

Recursive state estimation, also known as recursive Bayesian filtering, is a key concept in SLAM that involves estimating the state of a robot or an agent over time based on incoming sensor measurements. At the core of recursive state estimation is the concept of filtering, which refers to the process of iteratively updating the estimated state (orientation and position) of the robot as new sensor measurements become available. The estimation is performed in a recursive manner, meaning that each new measurement is used to update the current estimate, leading to an improved estimate for the next time step.

One widely used filtering technique in SLAM is the EKF, which is a variant of the Kalman filter suitable for nonlinear systems. The EKF approximates the posterior probability distribution of the robot's state using a Gaussian distribution. The algorithm consists of two main steps: 1) prediction and 2) update. During the prediction step the state is estimated based on its current estimate and the motion model of the robot. The motion model describes how the robot is expected to move from one state to another, taking into account factors such as velocity, acceleration, and rotational motion. By incorporating the motion model, the filter generates a prediction of the robot's state and estimates the uncertainty associated with the prediction. During the update step new sensor measurements are compared with the predicted state to refine the estimate. This update step combines the predicted state with the estimate from the sensor measurements, taking into account their respective uncertainties. The update process involves computing the likelihood of the measurements given the predicted state (measurement model) and adjusting the predicted state based on the difference between the predicted and measured values.

Eq. A.1 defines a generic motion model that takes 3 inputs x_{t-1} robot state at the previous timestamp, u_t control inputs at the current timestamp, and w_t the current noise parameters, to predict the state at the current time x_t. w_t is assumed to be Gaussian with zero noise and the covariances Q_t are set based on the noise of the raw inputs. x_t and w_t are both N dimensional vectors, with N being the number of parameters used to define the pose of the robot. $\hat{P}_0$ is the prior covariance on the initial pose, which can be set in various ways, typically the initial pose is set to be at the identity, with a very low covariance (high confidence).

$$\begin{aligned} &x_t = f\left(x_{t-1}, u_t, w_t\right); \quad t = (1, 2, \ldots, T) \\ &x_t \in \mathbb{R}^N; \quad x_0 \sim \mathcal{N}\left(\hat{x}_0, \hat{P}_0\right); \quad w_t \in \mathbb{R}^N \sim \mathcal{N}\left(0, Q_t\right) \end{aligned} \tag{A.1}$$

Eq. A.2 defines a generic measurement model, which relates the expected measurements given the state x_t and the measurement noise n_t. n_t is assumed to be Gaussian with zero noise, with its covariance R_t determined based on the noise of the raw measurements. y_t and n_t are both M dimensional vectors, where M is the number of measurements received at each time step. A detailed explanation of how the motion model and measurement model are used in the EKF as well as the derivation of the EKF solution can be found in [3].

$$\begin{aligned} &y_t = g\left(x_t, n_k\right); \quad t = (0, 1, \ldots, T) \\ &y_t \in \mathbb{R}^M; \quad n_t \in \mathbb{R}^M \sim \mathcal{N}\left(0, R_t\right) \end{aligned} \tag{A.2}$$

It is important to note that recursive state estimation holds the Markov assumption to be true. The Markov assumption claims that, given the current state of the system, the system's future state is independent of its previous states. In other words, all the data required to forecast the future state is present in the current condition.

A.2 Batch State Estimation

Batch estimation, is an alternate approach to state estimation in SLAM that involves processing all available sensor measurements and data in a single optimization to estimate the robot's trajectory and the map of the environment. It contrasts with recursive state estimation, which incrementally updates the estimate as new measurements arrive.

In batch estimation, all the available sensor measurements are collected and combined with the motion model and constraints to formulate an optimization problem. The goal is to find the trajectory and map variables that minimize the error between predicted and measured values, typically quantified as a squared error minimization problem. This optimization process is often solved using techniques such as least squares optimization or nonlinear optimization methods like Gauss-Newton or Levenberg-Marquardt. Whereas in recursive estimation we were estimating only the current robot pose x_t, in batch estimation we estimate the pose at all timesteps shown as a stacked vector of states in Eq. A.3. This state vector can also be extended with map variables (such as landmark positions) to simultaneously estimate the trajectory and the map.

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_T \end{bmatrix} \tag{A.3}$$

To estimate this state, we aim to minimize a set of defined error functions, which can be expressed again as a stacked vector of different error functions in Eq. A.4.

$$e(x) = \begin{bmatrix} e_a(x) \\ e_b(x) \\ \vdots \\ e_z(x) \end{bmatrix} = \begin{bmatrix} e_{a,0}(x) & \dots & e_{a,T}(x) & e_{b,0}(x) & \dots & e_{b,T}(x) & \dots & e_{z,0}(x) & \dots & e_{z,T}(x) \end{bmatrix}^T \tag{A.4}$$

Measurement error functions can be defined as the difference between the measurement, y_t and what is currently estimated, $g(x_t, 0)$ as in Eq. A.5. However, depending on how the problem is formulated, the definition of the motion error may differ. Since transformation matrices do not form a vector space (are not closed under addition), an error term cannot be established by subtracting the anticipated pose from the actual pose when the motion model is formulated using $\mathbf{SE(3)}$. As illustrated in Eq. A.6, a motion error can instead be specified as a function of the skew-symmetric transform ($\hat{}$), inverse skew-symmetric transform ($\check{}$), exponential map ($exp(\,)$), and logarithm map ($ln(\,)$). The exponential map maps a value from the Lie algebra ($\mathbf{se(3)}$) to the Lie group ($\mathbf{SE(3)}$), while the logarithm map performs the inverse. Details on Lie algebras and Lie groups can be found in [3].

$$e_{a,t}(x) = y_t - g(x_t, 0)\,; \quad t = (0, 1, \dots, T) \tag{A.5}$$

$$e_{b,t}(x) = ln(exp(\Delta t_t \hat{\omega}_t) T_{t-1} T_t^{-1})^{\vee}; \quad t = (1, \ldots, T) \tag{A.6}$$

$$e_{b,t}(x) = ln(\hat{T}_t T_t^{-1})^{\vee}; \quad t = 0 \tag{A.7}$$

Finally an objective function $J(x)$ can be set up as the sum of all the squared error terms weighted by the inverse of the covariances. The full objective function is expressed in Eq. A.8, which is commonly solved using the methods previously mentioned, Gauss-Newton or Levenberg-Marquadt, details of which can be found in [3].

$$J(x) = \frac{1}{2} e(x)^T W^{-1} e(x); \quad t = (1, \ldots, T) \tag{A.8}$$

Factor Graphs (FG)s are frequently used to set up the batch state estimation issue for simple visualisation and execution. A FG is a graphical model that is used to build optimization problems [15]. A simple state estimation problem is provided as an example for a FG in Fig. A.1. There are nodes and edges in a FG. While each edge represents a factor, each node represents a state. Fig. A.1 has 4 types of factors: 1) prior factors (f_0), 2) relative pose factors (f_1 and f_2) which could come from a sensor like an IMU or wheel odometry, 3) absolute measurement factors (f_3, f_4 and f_5) which could come from a sensor like a GPS and 4) pose to map variable (landmark) factor (f_6, f_7, f_8 and f_9) which could be produced from exteroceptive sensors like cameras or LiDARs.

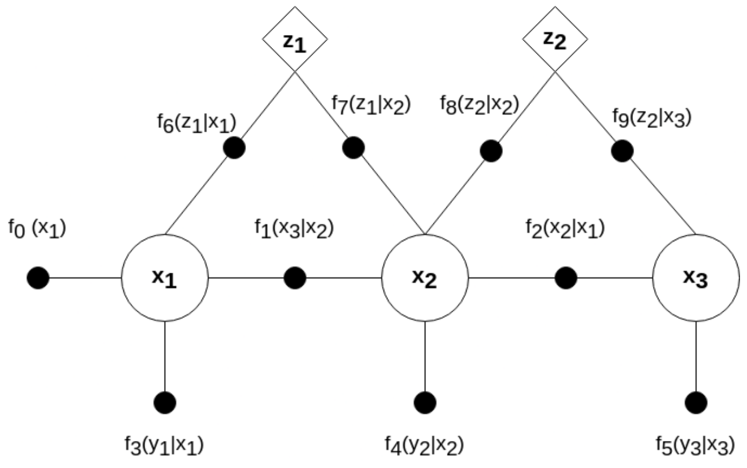


Figure A.1: Illustration of a factor graph figure with 4 types of factors.

Batch estimation offers one main advantage over recursive state estimation: global consistency, by considering all available data simultaneously, batch estimation can provide a globally consistent solution resulting in a more accurate trajectory and map. However, batch estimation also has some drawbacks, including its computational complexity (an ever-growing optimization problem as new measurements arrive) and delayed estimation, since batch estimation requires all data to be available before processing, it may introduce a delay in obtaining the final estimate. Both of these limitations severely restrict batch estimations usage in real-time applications. However an approach known as sliding window batch estimation can be used as a compromise between recursive and batch estimation.

In sliding window batch estimation, instead of considering all available data, a sliding window is used to select a subset of the most recent measurements and states for optimization. This window moves over time, incorporating new measurements and discarding older ones as the robot progresses through a process known as marginalization. The size of the window determines the number of states and measurements considered in the optimization problem. By restricting the optimization to a smaller window of data, sliding window batch estimation reduces the computational complexity compared to full batch estimation. It allows for efficient and incremental updates to the estimate, making it more suitable for real-time applications.