Visual Localization in Underground Mines and Indoor Environments using PnP

by

Edith Deretey

A thesis submitted to the
Department of Electrical & Computer Engineering
in conformity with the requirements for
the degree of Master of Applied Science

Queen's University
Kingston, Ontario, Canada
January 2016

Copyright © Edith Deretey, 2016

Abstract

This thesis presents a visual technique for localization in underground mines and indoor environments that exploits the use of a calibrated monocular camera. The objective of this work is to provide 6 degrees-of-freedom (6 DoF) localization in the presence of a 3D map made up of 3D point clouds co-registered with intensity information. An efficient data structure, referred to as the feature database, is used to store the image features appearing in the 3D map, and to efficiently retrieve a match for a query image during localization. The Perspective-n-Point (PnP) problem is then used to compute the 6 DoF position of the calibrated camera at the time of image capture given the location of the 2D query image features and their 3D coordinates.

Two experiments were performed using this technique: ground truth experiments and localization experiments. The ground truth experiments, performed on two datasets, compare the localization output against a reference position. The results of the ground truth experiments indicate that the calculated camera position using the proposed technique approximates the reference position, with small errors on the order of millimetres. It was also found that the accuracy of localization increases as more inliers become available in the query image. The localization experiments were performed on datasets collected in underground mines and indoor environments

to test the performance of the technique in different environments. The query images used in the localization experiments were captured with two different cameras, demonstrating that any type of monocular camera may be used during localization, as long as a sufficient number of environmental features can be extracted from the query images.

Co-Authorship

The work presented in this thesis was completed under the supervision of Dr. Michael Greenspan and Dr. Joshua Marshall. The contents were accepted for publication in a conference paper, which was co-authored by Ph.D. student Mirza Tahir Ahmed and my supervisors.

- Edith Deretey, Mirza Tahir Ahmed, Joshua A. Marshall, and Michael Greenspan. Visual indoor positioning with a single camera using PnP. Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN 2015), (Banff, Canada), Oct. 2015.

Acknowledgements

First and foremost, I would like to thank my supervisors Dr. Michael Greenspan and Dr. Joshua Marshall for their advice and guidance throughout my thesis research, and for the opportunity to work on this project.

Secondly, I am thankful for the members of the Robotics and Computer Vision Lab and the members of the Mining Systems Laboratory. In particular, I would like to acknowledge Mirza Tahir Ahmed, Mustafa Mohamed, Cody Stewart, and Andrew Yaworski for their advice and assistance during my research, and Heshan Fernando, Curtis Watson, Marc Gallant and Jordan Marr for their help with the field testing.

I am grateful to both the Barrick Hemlo Mine and to NORCAT for allowing me to collect data at their facilities. Thank you for being very accommodating during the site visit and for helping out with the experiments and the data collection process.

I am thankful to my family and friends for their encouragement and support. Finally, to my parents, thank you for your love, support, advice, and for the countless hours invested in my academic career.

Contents

List of Tables

List of Figures

Nomenclature

back name for the top part of the tunnel in underground mines.

DoF degrees of freedom.

DSA Double Solution Algorithm.

EMM environment measurement model.

FPS frames per second.

g^{2}o general graph optimization method [1].

GPS Global Positioning System.

IMU Inertial Measurement Unit.

IR infrared.

MAV micro aerial vehicle.

OI Orthogonal Iterative Algorithm.

OpenCV open source computer vision library [2].

PCL point cloud library [3].

PnP Perspective-n-Point problem.

PTAM Parallel Tracking and Mapping.

RANSAC Random Sample Consensus [4].

RFID Radio-frequency identification.

RGBDSLAM open source SLAM system for RGB-D cameras available in the robot operating system [5].

ROS robot operating system [6].

RTV rugged terrain vehicle.

SLAM simultaneous localization and mapping.

SVD Singular Value Decomposition.

VO visual odometry.

YAML human friendly data serialization file.

Chapter 1

Introduction

1.1 Motivation

In the field of navigation, localization is the problem of determining the position of a user at a particular point in time. Localization can be performed using a combination of absolute and relative measurements and is a key component of navigation systems [7]. Applications of localization include tagging location information for captured photographs, location-based advertising, and displaying the user's location in web-based map services such as Google Maps.

The introduction of the commercially available Global Positioning System (GPS) has significantly advanced the use of localization technology and the way we live our lives. GPS is an absolute positioning system which uses a network of satellites to find the position of the receiver [8]. Direct line of sight to a minimum of four satellites is required to determine the receiver's position, making it possible to find the receiver's location in most outdoor environments with coverage. The availability of GPS has opened the door to numerous technologies such as commercial GPS navigation systems and self-driving vehicles.

On the other hand, indoor localization is challenging due to the lack of an absolute positioning method similar to GPS [9]. Indoor localization has many uses, and the anticipated systems would be valuable in a number of different scenarios such as hospitals and malls, as well as industrial settings such as factories and underground mines. Research into localization in indoor environments is important because people spend the majority of their days indoors. In addition, there are many large, complex indoor environments, for example factories, malls and hospitals. Being able to localize inside these environments would increase safety and efficiency for the user. To provide an example, let us look at the scenario of a new hospital patient looking for the location of their appointment [10]. Since hospitals are complex structures made up of many wards and floors, finding the room is a complex task. The patient could get lost and be late for their appointment. They may also decide to ask for directions from one of the hospital workers, for example a nurse. This interrupts the nurse from their current work and makes them less efficient in completing their tasks. Such a scenario could also apply to other complex indoor environments. Therefore, having a widely available and accessible method for indoor localization would improve safety, productivity and the user's experience in finding their way around an unfamiliar environment.

Another challenging environment for localization is in underground mines [11]. This environment can be thought of as a more extreme case of indoor environments, as GPS is not available in underground mines either. Underground mines contain large networks of tunnels spanning multiple kilometres over many levels. Mines are expanding further below the surface of the earth in order to be able to extract the ore, and the dangers of working underground increase further down. Underground mines

are hazardous environments and dangers include cave-ins, vehicle collisions, and fires. There are systems in place to keep track of who is working underground at any point in time, with a tag-in/tag-out board, where a miner tags-in to indicate their location before heading underground. However, having a system available where each miner's precise location is known at all times would increase the safety of underground mines. In case of an emergency, knowing the precise location of all miners could speed up their rescue. In addition, many large vehicles operate at high speeds underground. Being able to track where all the vehicles and miners are could help prevent collisions. Keeping track of the location of the vehicles could also play a part in optimizing the mining process by preventing bottlenecks in high-traffic areas of the mine.

1.2 Problem Overview

Indoor localization techniques presented in the literature use a variety of sensors such as WiFi [12], radio-frequency identification (RFID) [9], odometers, inertial measurement units (IMUs), ultra-wideband radios [13], and optical systems. Odometers and IMUs are relative measurement sensors, measuring the position of the sensor with respect to its previous position. These two sensors are accurate over short distances, but accumulate error over time [7], and therefore they are not reliable on their own for localization. WiFi can be used as a method of determining absolute position by measuring the received signal strength. A database of the environment is first created, which stores the radio signal map of the signal strengths at different locations of the environment. Then at the time of localization, the collected signal strength is compared against the database to determine the receiver's position [14]. While this technique can determine absolute position, the disadvantage is that a network

of wireless transmitters must be installed throughout the environment. Therefore localization in indoor environments is currently an ongoing area of research, with most techniques using a combination of sensors.

The sensors that can be used for localization in underground mines are even more limited. While using WiFi for localization is common in indoor environments, this technique is not ideal for underground mines due to the large task of installing the wireless transmitters throughout the entire mine. As mines are constantly expanding, wireless transmitters would continuously need to be installed to account for these changes. RFID tags, which also require installation throughout the mine, may also be used for localization [11]. As the vehicle drives near one of the RFID tags, the ID of the tag is read, providing the vehicle with information about its approximate position within the mine. Other techniques include using laser scanners or LiDARs to create a 3D map of the mine and perform localization within the map. However these sensors are expensive [3], and therefore they are less attractive for wide-scale use.

This thesis focuses on developing an inexpensive localization system within a previously mapped environment that can be used both indoors and in underground mines. The proposed system should be affordable, without requiring the use of expensive sensors. The system should require minimal set-up with only minor changes to the environment, as opposed to the WiFi networks or RFID tags which need to be installed throughout the environment. The goal of this work is to determine whether a calibrated camera can be used to perform localization in indoor and underground mine environments without requiring the use of other sensors at the time of localization.

1.3 Research Objectives

The goal of this research is to create a simple, inexpensive system that can be used for localization in underground mines, using the smallest number of sensors possible.

Specific objectives of this thesis include:

1. Determine the feasibility of using monocular cameras in underground mines. Cameras are inexpensive compared to commercially available 3D sensors, widely available on the market with a large selection of models, and would allow the use of computer vision techniques to solve the localization problem. However, the main challenge of using cameras in underground mines is the lack of lighting available in the environment. With the help of external lights mounted onto the sensor platform, and with adjustment of the different camera parameters, it would possible to use cameras in this environment. Therefore the goal of this objective is to explore what the requirements are to be able to use cameras as the main sensor in underground mines.

2. Create a prototype of a system capable of performing localization using only a calibrated monocular camera.

3. Test the performance of the proposed system in different environments. This involved collecting the required data in each environment. The technique was mostly tested on indoor environments, as they were more accessible than underground mines. However, datasets collected from two underground mines were also used for testing the system.

4. Obtain a target localization accuracy of less than 0.5 m. The accuracy of existing indoor localization systems range from as little as 0.5 mm to as much

as 3 m, depending on the type of sensors and technique used [14]. Since the purpose of the proposed technique is to keep track of the position of people and vehicles in the environment, the target accuracy of the proposed technique should be less than 0.5 m.

1.4 Thesis Contributions

The main contributions of this thesis are as follows:

- Creating a localization system prototype using a calibrated monocular camera, incorporating several existing algorithms such as the Perspective-n-Point (PnP) problem [15], Landmark Indexing [16], and open source RGBDSLAM [5].
- Modification to Landmark Indexing, incorporating a moving window average when searching for the best matched database image in order to reduce the probability of returning an outlier.
- Collection of datasets in two underground mines, used for testing the performance of the proposed system. This is unique because vision sensors are not commonly used in underground mines due to the lack of lighting. In order to collect these datasets, experiments were first performed to determine the configuration of lights and camera parameters that resulted in the highest quality images.
- Proof of concept that monocular cameras can be used in underground mines to perform localization.
- Results showing that a different vision sensor can be used for localization than the sensor that was used for map building. This is advantageous because it

allows for more flexibility in choosing what camera to use for localization. Additionally, if a more expensive sensor is required to build a map of the environment, inexpensive cameras can still be used for localization, maintaining the affordability of the system. This is different from similar works, where the same sensors are used for both map building and localization.

1.5 Organization of Thesis

The remainder of the thesis is organized as follows:

Chapter 2, Background: The first part of this chapter provides a detailed description of the PnP problem and summarizes the key PnP algorithms available in the literature. The second part of this chapter presents different localization algorithms that use vision sensors. These algorithms are divided into those using RGB-D sensors, and those using only cameras.

Chapter 3, Methods: This chapter describes the proposed localization algorithm in detail. The chapter begins by summarizing two existing algorithms: RGBDSLAM and Landmark Indexing. These two works are used to aid localization. This is followed by a description of the data collection process used to collect the query images, and ends with a detailed description of the implemented localization process.

Chapter 4, Experimental Results: This chapter contains a description of all the experiments that were conducted on the proposed system. The chapter first describes the different datasets used, and proceeds to present the results of the ground truth experiments and the localization experiments. The chapter concludes with a presentation of the runtime of the proposed algorithm.

Chapter 5, Conclusions and Future Work: This chapter concludes the thesis

and outlines future work that could be performed on the proposed technique.

Chapter 2

Background

2.1 Perspective-n-Point Problem

The Perspective-n-Point (PnP) problem is the problem of determining the location of a calibrated camera using a number, n, of 3D object points and their corresponding 2D image projections. PnP is typically used within a Random Sample Consensus (RANSAC) [4] framework. Applications of PnP include determining the position of the camera within an environment [17], localization [18], or for finding the position of an object within the camera's field of view [19].

PnP has a multiple-solution phenomena that is widely studied in the literature. A minimum of $n = 3$ points are required in order to fully constrain the system and obtain a finite number of solutions for the camera position [20]. However, the case of $n = 3$ can result in up to four solutions, depending on the configuration of the 3D object points selected. Similarly, in the case of $n = 4$, the problem can have up to five solutions, whereas the case of $n = 5$ yields up to two solutions [21]. The case when $n \geq 6$ can be solved linearly and always produces a single unique solution [22]. Therefore, the study of PnP is focused on the P3P, P4P and P5P cases.

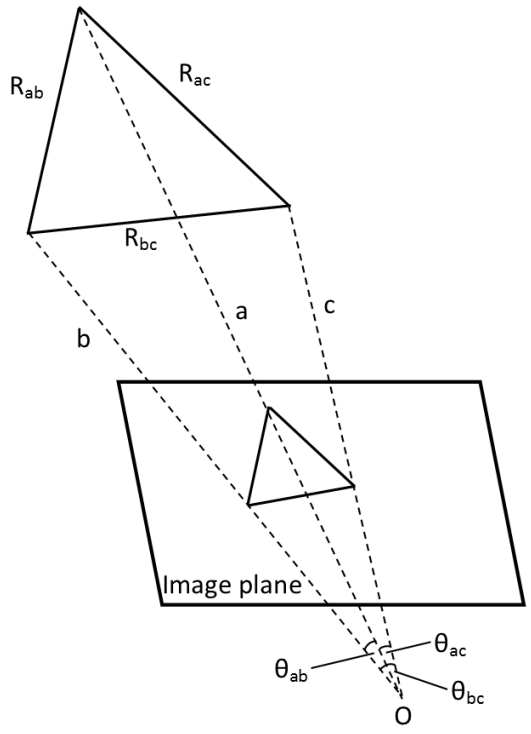


Figure 2.1: Illustration of the P3P geometry. The 3D location of the object points (the vertices of the triangle) are known, as well as their 2D projections on the image plane, and the camera's intrinsic parameters. a, b, and c are the unknown distances between the object points and the centre of perspective of the camera, O.

The research on PnP can be categorized into two main areas: the first focuses on creating robust algorithms for solving the camera position, while the second area focuses on solution classification [23]. The solution classification algorithms study the different cases of object point configurations and determine how many solutions are expected for the specific scenario. In most cases when multiple solutions exist, only one of the solutions is correct. Therefore knowing the expected number of solutions of the PnP problem is crucial in ensuring that all solutions have been explored and that the correct solution has been found.

Existing PnP algorithms can be divided into two categories: direct solutions and iterative solutions. Direct solutions are faster and less computationally expensive. However, they are sensitive to noise and outliers because they can only handle a limited number of reference points. Iterative solutions are accurate when initialized with good parameters, but require more computation to solve [24]. Additionally, iterative solutions can only find one solution at a time [25].

2.1.1 Direct Solutions

One of the earliest solutions of PnP can be found in [4], where Fischler and Bolles presented a solution to the P3P problem, along with a discussion on the number of solutions for the P4P and P5P problems. Their approach to the P3P problem begins by finding the length of the distance between the camera's centre of perspective and each of the object points. They begin by using the cosine law to obtain the following set of three equations:

$$\begin{aligned} R_{ab}^2 &= a^2 + b^2 - 2ab\cos(\theta_{ab}), \\ R_{ac}^2 &= a^2 + c^2 - 2ac\cos(\theta_{ac}), \\ R_{bc}^2 &= b^2 + c^2 - 2bc\cos(\theta_{bc}), \end{aligned} \tag{2.1}$$

where R_{ab}, R_{ac}, and R_{bc} are the distance between the object points, a, b, and c are the unknown distances between each object point and the camera's centre of perspective, and θ_{ab}, θ_{ac}, and θ_{bc} are the angular distances from the centre of perspective to the object points. In what follows, (2.1) is referred to as the *P3P equation system*, since it is used as a starting point for a number of PnP algorithms.

The P3P equation system is modified to form a fourth order polynomial

$$\mathrm{G4} * x^4 + \mathrm{G3} * x^3 + \mathrm{G2} * x^2 + \mathrm{G1} * x + \mathrm{G0} = 0, \tag{2.2}$$

where the coefficients G4, G3, G2, G1, and G0 represent a combination of the known values. The equations for these coefficients are shown in (A.1)-(A.5) in Appendix A. They are nonlinear equations, derived from the geometric parameters of the P3P equation system (2.1). The roots of (2.2) are found and used to compute a, b, and c, the lengths between the centre of perspective and each object point. Next, the 3D location of the centre of perspective is found, followed by the orientation of the image plane with respect to the world coordinate system.

In order to take advantage of data redundancy in the case when more than three points are available, Quan and Lan [26] proposed a linear method for obtaining a unique solution to the P4P and P5P problems. They start with the P3P equation system and obtain $\frac{(n-2)(n-1)}{2}$ fourth order polynomials, different from [4], for varying subsets of three points. These polynomials are arranged to form a measurement matrix. The square of the depth of the reference points is obtained by applying singular value decomposition (SVD) to the measurement matrix. The absolute orientation technique described in [27] is then used to find the camera's rotation and translation. Quan and Lan tested their technique on both synthetic and real data and both P4P and P5P algorithms were found to be stable in the presence of noise.

Horaud et al. [20] employed a geometric approach to solving the P4P problem. They began with four object points and drew three lines connecting all four points together. These line segments in 3D space were projected onto the image plane, and the geometric properties between the 3D lines and their projections were studied.

The key concept behind their technique is that the transformation matrix used to convert from the object-centred frame to the camera-centred frame can be split into two intermediate transformation matrices, $\mathbf{A}_1$ and $\mathbf{A}_2$. They begin by expressing the three lines in both the image frame and in the object frame. The matrix $\mathbf{A}_1$ is found by expressing the coordinate system that defines the image frame in terms of the camera frame. Meanwhile, $\mathbf{A}_2$ is found by expressing the lines from the object frame to be in terms of the image frame. This leaves two unknown angles, θ, ϕ, and the distance between the image frame and the object frame, represented by d_x. The following fourth order equation is obtained by applying geometric constraints

$$I_1 \cos^4 \phi + I_2 \cos^3 \phi + I_3 \cos^2 \phi + I_4 \cos \phi + I_5 = 0. \tag{2.3}$$

After finding ϕ, θ can be found by substituting the value of ϕ into one of the intermediate equations. The last unknown, d_x can then be solved geometrically. In the case of multiple solutions for (2.3), geometric constraints are applied to eliminate any solution that is outside the range of admissible values. There are some special configurations which result in a simpler equation, for example the case of coplanar points, three collinear image points, or when the object points form a right vertex.

Lepetit et al. [15] proposed a technique where the PnP problem could be solved in O(n) time, and have named it EPnP. The main idea behind EPnP is to express the 3D object points in terms of the weighted sum of four virtual control points. Doing this allows the solution to be found in O(n) time regardless of the number of object points present, since they are solving for the location of the four virtual control points. Using the projection of the 3D points on the image plane, they then obtain

$$w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}, \tag{2.4}$$

where (u_i, v_i) are the projection of the 3D points on the image plane, w_i is a scalar projective parameter, α_{ij} is the weight, the 3×3 matrix is the intrinsic matrix of the camera and (x_j^c, y_j^c, z_j^c) are the unknown locations of the virtual control points in the camera coordinate system. From (2.4), the authors substituted the third row into the first two rows to obtain the following two equations:

$$\sum_{j=1}^{4} \alpha_{ij} f_u x_j^c + \alpha_{ij}(u_c - u_i) z_j^c = 0, \tag{2.5}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_v y_j^c + \alpha_{ij}(v_c - v_i) z_j^c = 0. \tag{2.6}$$

These equations are rearranged to be in the form of $\mathbf{Mx} = \mathbf{0}$, where $\mathbf{x}$ contains all the unknowns and matrix $\mathbf{M}$ contains the rearranged coefficients of (2.5) and (2.6). The solution to the problem can be obtained by finding the null space of $\mathbf{M}$. In practice, this is done by using SVD to find the null eigenvectors of the matrix $\mathbf{M}^\top\mathbf{M}$. Multiple solutions may exist depending on the configuration of the object points. In the case of multiple solutions, all solutions for EPnP are computed and the solution that provides the smallest reprojection error is chosen as the correct solution.

From their experimental results on both real and synthetic data, EPnP was found to be more accurate than any of the state-of-the-art non-iterative PnP solutions. When compared to iterative solutions, EPnP performed faster than the best known iterative solution of Lu et al. [28]. However, EPnP was found to be slightly less

accurate than Lu's iterative solution when Lu's technique was provided with good initial values.

Li and Xu [29] solved the P3P problem by converting it into the Perspective Similar Triangle problem. By reducing the number of unknowns to two and applying the properties of similar triangles, they obtained a fourth order equation, which provided a solution to the P3P problem. They then extended this technique to the case of $n \leq 5$ in [30], by dividing the object points into subsets of three points each, and using the P3P algorithm mentioned above to obtain a fourth order polynomial for each subset. A cost function was used to combine these polynomials into a seventh order polynomial, whose roots are solved to obtain the final position.

Other direct solutions include Leng and Sun [25] who proposed a technique to find all the solutions to the PnP problem when one of the solutions is already available. Any iterative technique can be used to find the initial pose required for this technique. Kneip et al. [31] presented a PnP solution that computes the camera's position directly without having to convert the location of the control points into the camera's frame as is typical for most PnP techniques. Other direct solutions to the PnP problem include [32], [33], [34], and [35].

2.1.2 Iterative Solutions

In [36], Haralick et al. presented three iterative solutions to PnP. The first was an iterative least-squares solution, which starts by estimating the depth values for each object point. A rotation and translation are obtained and used to solve an error function. This process is iterated until a rotation and translation that minimizes the error function is found. The second solution is a least-squares adjustment by

linearization. It begins by describing the rotation matrix as a function of three angles. With each iteration, a translation value and a correction is found for these angles. The assumption is that these corrections are small enough to maintain the linearity of the system. The correction is added to the estimate and the process is repeated until the correction becomes negligibly small. The third solution uses the robust M-estimation technique to solve the problem.

Lu et al. [28] presented the Orthogonal Iterative Algorithm (OI). They use two collinearity equations to form their algorithm. The two equations are called the *image space collinearity equation* and the *object space collinearity equation*. The OI algorithm starts with the object-space collinearity equation and attempts to minimize the sum of squared error using

$$E(R, \mathbf{t}) = \sum_{i=1}^{n} ||e_i||^2 = \sum_{i=1}^{n} ||(I - \hat{V}_i)(R\mathbf{p}_i + \mathbf{t})||^2, \tag{2.7}$$

where R and t are the rotation and translation respectively, $\hat{V}_i$ is the observed line-of-sight projection matrix, and $\mathbf{p}_i$ are the coordinates of the input reference points. This equation is then split so that the translation can be expressed in terms of the rotation, and the main focus becomes finding the optimal rotation. Since the final equation used to calculate the rotation is dependent on itself, the rotation must be computed iteratively. Each rotation value is a function of the rotation obtained in the previous iteration. The translation for each iteration is calculated using the rotation obtained in the respective iteration. The initial rotation value can either be guessed or calculated using a weak perspective approximation of the the absolute orientation problem. The OI algorithm is globally convergent, such that a solution is always obtained, regardless of how close the initial values are. However, less iterations are

required when the initial values are closer to the solution.

Inspired by [36] and [28], Zhang et al. [24] presented a globally convergent iterative solution to the PnP problem based on an inverse projection ray approach. The inverse projection ray is defined as a ray which starts at the camera centre and points to the 3D object point. In the depth recovery stage the translation vector is calculated as a function of the rotation matrix and then the depth of the reference points are estimated using the inverse projection ray. This is followed by estimating the rotation matrix in the absolute orientation stage.

In [37], PnP is applied to specific configurations of reference points in order to obtain a unique solution. In the above case, four reference points must either form a triangle with the fourth point being in the centre, or they must form a quadrilateral. The remaining reference points are used to improve the accuracy of the solution.

2.1.3 Solution Classification and The Danger Cylinder

Solution classification algorithms classify the PnP problem into different categories based on the configuration of the 3D object points. These categories reveal how many solutions exist for the specific problem. For example, as mentioned before, the P3P problem can have up to four solutions. However, certain configurations of the control points would only yield two solutions. In practice, it is helpful to know the number of solutions that exist for certain configurations of control points to ensure that all solutions have been found and that the best solution is chosen as the correct one.

The first solution classification problem for the P3P problem was presented by Gao et al. [23], classifying the P3P problem into ten components. Their solution begins with the P3P equation system, combining the equations and eliminating variables

to obtain two quadratic equations. Finding the positive solutions of these quadratic equations solves the P3P problem. Wu-Ritt's zero decomposition method is used to decompose the problem into ten components, with each component containing a polynomial equation in triangular form, and polynomials. The entire set of equations for each component can be found in [23]. The first component is called the main component, and the solution falls into the main component most of the time. All other components are referred to as degenerate cases. The equations for the main component are

$$\begin{aligned} &a_0x^4 + a_1x^3 + a_2x^2 + a_3x - a_4 = 0, \\ &b_0y - b_1 = 0. \end{aligned} \tag{2.8}$$

Gao et al. also classified the maximal number of solutions for each of the ten components. Depending on the component, the polynomial that needs to be solved to find the solutions of P3P may be linear, quadratic, cubic or quartic. The rest of their paper shows how to solve each case to obtain all the solutions to the problem.

A more specific solution classification is presented in [38], which only applies to the P3P case when the three control points form an isosceles triangle. Sun and Wang [39] present another solution classification for the P3P problem, focusing on the case when the optical centre of the camera approaches the danger cylinder. They found that the number of solutions varies between three and four as the centre of perspective nears the danger cylinder. The solutions of the P4P problem for non-coplanar control points are classified in [21]. They found that even though the P4P problem can result in up to five solutions, most P4P configurations have only one solution.

Some P3P algorithms fail when the control points lie on or near the danger cylinder. The danger cylinder occurs when two conditions are met: First, the camera's centre of perspective is on or near the cylinder that contains all three object points. Second, the axis of the centre of perspective is perpendicular to the plane containing the object points [40]. The danger cylinder causes the P3P solutions to behave erratically. While having more object points results in a more accurate solution for the PnP problem, there are times when only three object points are available [40], which is why the study of the danger cylinder is important.

A solution to the P3P problem, called the Double Solution Algorithm (DSA) is presented in [40]. This solution is used when the centre of perspective is on or near the danger cylinder. Another solution to the P3P problem for non-collinear points located near the danger cylinder is presented in [41]. This is done by focusing on obtaining the camera's location, as its orientation can be obtained afterwards. Each time the camera moves, the change in coordinate system is obtained by finding the proper affine transformation to apply. The solution is iterative but when multiple solutions exist, only two of them can be found.

While the danger cylinder is an important case in P3P, not many papers are available on this topic. The papers discussed above are fairly recent, but it is doubtful whether much future research will be done in this area because it is such a specific area of study, only affecting a small portion of the PnP problems. In practical cases, the situation of having exactly three control points in an image and having those points, as well as the centre of perspective of the camera within the danger cylinder is small. Additionally, some of the current P3P algorithms, such as [22], still work even for points on or near the danger cylinder.

2.2 Localization Using Visual Sensors

Localization using visual sensors has been widely studied and multiple systems have been presented. Some systems use a series of other sensors such as GPS, IMUs and laser scanners in addition to cameras. However, these sensors are costly and the price can easily add up. With the increase in the amount of affordable cameras and RGB-D sensors in the market, such as the Microsoft Kinect, the interest in developing real-time navigation systems using these sensors has increased.

This section provides an overview of the key visual localization techniques available in literature. It is split into three sections, discussing techniques using RGB-D sensors, cameras without previous knowledge of the environment, and teach-and-repeat techniques, where information about the environment is available to the system.

2.2.1 RGB-D Sensors

Wang et al. [42] presented a technique for real-time 3D mapping in indoor environments using RGB-D sensors. Their technique is split into two steps: visual odometry, and 3D mapping. The visual odometry stage is used to determine the movement of the camera from its previous position. SURF [43] is used to detect and extract interest points in the current image (target frame) and the previous image (reference frame). The interest points are then matched to determine which points appear in both images. Then the image coordinates of these points taken from the current frame, and their depth information from the reference frame are input into P3P [31] to determine the motion of the camera.

Scherer and Zell [44] presented an RGBD-SLAM technique for autonomous micro aerial vehicles (MAVs). SLAM, which is short for simultaneous localization and

mapping, is the problem of creating a map of an unknown environment while determining the sensor's location within the map at the same time [45]. SLAM algorithms generally consist of a prediction and a correction step, which are repeated recursively. The localization in Sherer and Zell's technique is done by calculating a model for the vehicle's motion. For each new frame, the position of the current camera is estimated by this motion model. All the map points are then projected onto this frame using the estimated camera pose, and the keyframe with the most points located in the image is selected. Once the keyframe has been found, FAST [46] corners are detected in the current input image, and optical flow is used to determine which of the interest points in the keyframes are close to the FAST corner locations. These points are input into Gao's P3P algorithm [23] to find the current pose of the camera.

RGB-D sensors may also be combined with IMUs to determine the position of a sensor suite, such as in the technique presented by Huang et al. [47]. Their algorithm combines visual odometry, which is done on-board, and SLAM, which is performed on an external computer. For the most part, visual odometry is used for localization, with occasional use of SLAM to correct the drift in position estimates over time. FAST features are extracted from the image and inliers are detected. A rough estimate of the motion is then found using Horn's absolute orientation algorithm. This estimate is refined using a least-square solver and the feature matches that exceed a reprojection error threshold are discarded. The process is repeated and the final motion estimate is calculated.

Many SLAM algorithms are available for the Microsoft Kinect sensor. An example of one is presented in [48], which uses the images to extract keypoints, match them to previously found features, and determine the position of the camera using SVD. The

output of the algorithm is a 3D map of the environment, represented as a volumetric voxel map, along with the rotation and translation indicating the Kinect's current position with respect to the first frame.

While RGB-D sensors generally provide dense depth information, certain environments are difficult to sense with these sensors. For example, in some environments the distance of the objects from the camera may be out of range of the sensor, or if the environment contains excess sunlight, then some of the depth information is not available. Since depth data may not always be present, Scherer, Dube and Zell [44] presented a SLAM technique which uses monocular SLAM and integrates depth information into the system when it is available. This technique is a modification of an algorithm called Parallel Tracking and Mapping (PTAM). The modifications include adding depth points to the map without the need for triangulation and including depth in the bundle adjustment [49] process. Bundle adjustment is the iterative process of refining a visual reconstruction of a scene by minimizing a cost function to get an optimal structure and camera parameters. They found that including depth information increased the accuracy of visual SLAM.

2.2.2 Cameras in Unknown Environments

A visual SLAM system using stereo cameras was presented by Clipp et al. [50]. Their technique uses optical flow to determine the local motion of the camera by tracking features through consecutive images. The features in the two cameras are matched and triangulated to obtain 3D coordinates of each feature. Optical flow is used to track these 3D features through consecutive images. If the minimum average optical flow of features exceeds a threshold, then the image is selected to be a new keyframe.

Once a keyframe has been found, feature detection is performed and the 3D and 2D points are passed into P3P within a RANSAC framework to determine the position of the keyframe. Windowed bundle adjustment is used to refine the pose estimation, as errors exist throughout the pose estimation process. The adjusted keyframe is then added to the map. Finally, loop closure is performed on the map. This technique can also be used as visual odometry, in which case the loop closure is omitted. However, without loop closure, visual odometry accumulates large drift over time.

Visual localization techniques with monocular cameras rely on geometric constraints of the features. Sun et al. [51] presented a technique for monocular self-localization in an indoor environment using circular landmarks. At least two landmarks, in their case ceiling lights, must be in the camera's field of view in order to localize the camera. Specific points, such as the centre and tangent points, are selected on the circle and are used with PnP to solve for the camera position. The technique is accurate enough for indoor localization, resulting in only a small position and orientation error. The two main sources of error for this technique are blurry images and incorrect feature correspondences. Since this technique is limited to circular objects, which are not found in nature, the technique is constrained to indoor, man-made environments.

Localizing planar objects using PnP and planar geometry has been presented by Xu and Liu [52]. Constraining the control points to be planar allows the technique to check that the solution is correct, and to eliminate incorrect points. Each 3D point is compared to its neighbouring points to decide if it is planar with them, and if it is, it gets a vote. This is called the *normal vector voting strategy*. To check if the estimated pose is correct, triangulation is applied knowing the corresponding points

between the query and training image, as well as the estimated position, and the 3D points are reconstructed. The 3D points are checked to ensure they are planar, in which case the estimated pose is correct.

While these localization techniques do not require prior knowledge of the environment, they are limited to environments where the geometric constraints are met. Other techniques used in visual SLAM for obtaining a 6 degrees of freedom (DoF) position for the camera include using a branch and bound algorithm [53], or using stereo vision systems such as in [54] and [55].

2.2.3 Teach-and-Repeat Algorithms

Some visual localization techniques fall into a category called *teach-and-repeat*. Unlike the techniques described in section 2.2.2 where no prior knowledge of the environment is available, teach-and-repeat techniques begin by building a map of the environment before localization. In the teach phase the camera is moved through the environment and collects images. These images are saved and used to create a 3D map, typically using structure from motion. Then during localization the images obtained by the camera are compared against the 3D map to determine the position of the camera. This section covers some of the key teach-and-repeat algorithms in more detail.

Royer et al. [56] presented a teach-and-repeat technique using a single monocular camera. Structure from motion is used to build the 3D map of the environment with the data obtained during the teach phase. The information from the map is stored in a database containing landmarks with their 3D locations, in addition to position information for all the keyframes in the map. The localization process begins by capturing the current image and comparing it against each keyframe to find the

best match. The location of the best matched frame is used as the starting point for localization. This technique assumes that the movement of the camera between consecutive frames is small, therefore the approximate pose of the new frame is the same as the calculated pose of the previous frame. They then project the interest points present in the matched keyframe and find the 2D projection of those points on the current image. These projections are matched to the image interest points, which should be located in a small region of interest around the projected interest point. This matching step provides them with a list of 3D object points and their corresponding 2D image projections which is passed into PnP. The output of PnP gives them the pose of the camera.

Another technique for outdoor urban navigation using a monocular camera was presented by Segvic et al. [57]. Their 3D map is also created using structure from motion on the images collected during the teach phase. During localization, the best matched keyframe is found, and features are also tracked as new images are obtained. This allows them to smoothly transition between keyframes and to predict features. For feature detection and matching, they use three different algorithms and fuse them by allowing only one keypoint in a certain radius of pixels. By using three algorithms different types of features are found, which often complement each other and therefore more information can be obtained. Good matches must pass a certain threshold to be accepted, meaning the distance to the second-best match must be less than 60% of the distance to the best match.

Furgale and Barfoot [58] present a teach-and-repeat algorithm for robot navigation using a stereo camera. Their technique is similar to [56] and [57]. Instead of having one large global map, their database is made up of many submaps, which makes

the localization faster. The localization component is split into two parts, using visual odometry (VO) and localization. They found that having two techniques for localization improved the accuracy of their system and was very good in situations when one of the localization techniques failed, as the other technique could then take over. In the VO component, interest points from the stereo camera are matched to points in the database and the inliers are used to calculate the position of the camera. Localization is then used to improve the estimated position by combining the prior location obtained by VO and the error term to get the position of the camera. Clement et al. [59] modified this technique to work with a single monocular camera, estimating the depth using knowledge of the camera's position and orientation relative to the vehicle and approximating the ground to be planar.

2.3 Discussion

Many visual localization algorithms exist in literature, some of which have been discussed above. These algorithms either use SLAM to build a map and localize at the same time, or use teach-and-repeat techniques to first build a map and then follow by localization. These techniques, including those presented above, use the same sensor for both the map building and localization stages.

The key idea behind the localization technique presented in this thesis is that a different sensor can be used at the time of map building and localization. The proposed technique uses an RGB-D sensor (Microsoft Kinect version 1) to build a map of the environment, and then uses a calibrated monocular camera for localization. The idea behind this is that map building would be done offline and only once, while localization would be performed online multiple times. Therefore reducing the cost

of the camera used for localization is important, while a more expensive sensor may be used to build a map. Localization can be performed using the same map as long as significant changes do not occur in the environment. When the environment has changed sufficiently, a new map would need to be built to account for the changes.

The proposed technique falls into the category of teach-and-repeat algorithms, and is similar to that of Royer et al. [56]. Having a previously obtained 3D map of the environment allows the localization algorithm to not be bound by geometric constraints or specific configurations of the 3D object points. In addition, the proposed technique uses inexpensive, commodity cameras, which are widely available on the market, for both map building and localization. This provides the user with flexibility when choosing which camera to use.

Unlike Royer et al.'s algorithm, the proposed localization technique relies entirely on PnP to find the position of the camera, without needing to know the location of the previous image. Therefore the motion of the camera between consecutive images does not need to be small, as each image is only compared against the feature database. The efficiency of the feature database when using small datasets allows the system to maintain quick localization.

Chapter 3

Methods

The proposed localization algorithm relies on having a previously generated 3D map of the environment readily available. The map is used to create a feature database, which is then used to drive the localization process. While creating a 3D map requires the entire environment to be mapped before localization can be performed, it is still a better option than using RFID tags. First of all, RFID tags do not pinpoint the exact location of the user. Instead they tell the user that they are within range of a particular RFID tag, therefore providing low positioning accuracy [60]. Secondly, RFID tags must be installed throughout the entire environment and their exact location must be known. Since underground mines contain many levels spanning multiple kilometres, the installation of RFID tags is a time consuming process. As the mines continue to expand, additional tags need to be installed. The process of creating a 3D map is easier and quicker than installing RFID tags throughout the mine, as the mapping sensor can be attached to the vehicle, which is then driven through the mine to create the map.

The feature database is made using all the individual 2D images that were used to generate the 3D map. The feature database contains: a list of all the 2D image

features and their 3D point correspondences; the list of images each feature appears in; and a transformation matrix describing the position of each image. The images in the database are referred to as *database images*. *Query images*, which are the images obtained by the monocular camera at runtime, are compared with the database images to obtain the closest database image using *landmark indexing* [16]. The features between the query and the returned database image are matched, resulting in a set of correspondences between query image and database image features. The 3D coordinates of the matched database features are then read, and PnP is used to find the exact position of the query camera at the time of image capture. A block diagram of the entire process is shown in Figure 3.1. A more detailed block diagram of just the localization is presented later in the chapter. This chapter describes these processes in more detail.

3.1 Map Building

The open source RGBDSLAM [5] system available in the Robot Operating System (ROS) [6] is used to build the map of the environment. RGBDSLAM works with a number of RGB-D sensors, such as the Microsoft Kinect (version 1) or stereo cameras. This section briefly summarizes RGBDSLAM to provide an overview of how the 3D map of the environment is created, and discusses modifications done to the map to prepare it for use in the proposed localization system.

RGBDSLAM is a graph-based SLAM system. The front end of the system estimates the motion of the camera between the current input and the previous step. This is done by observing visual features using either SIFT [61], SURF [43], or ORB [62], and matching similar features between the current image and the previous image.

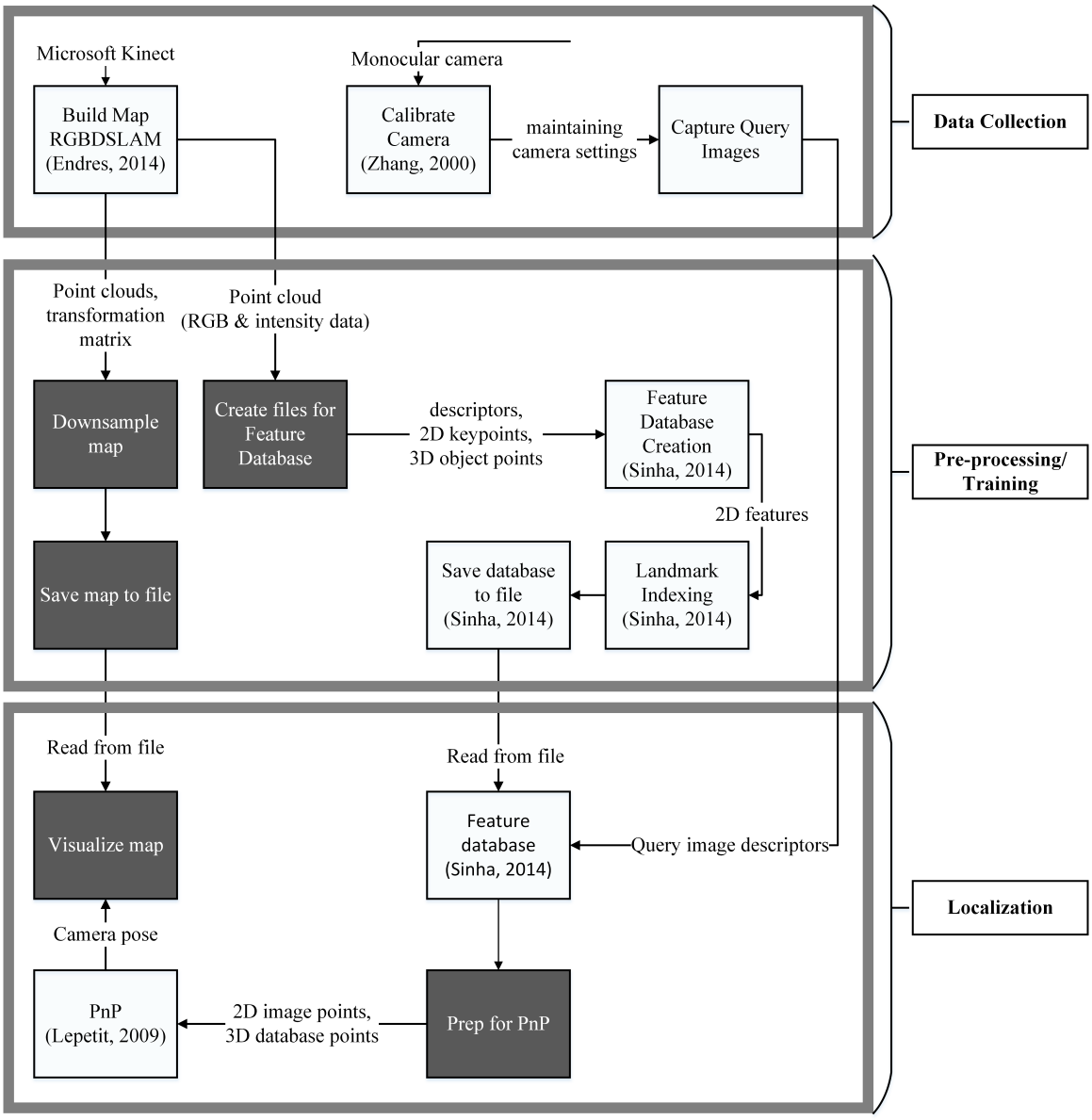


Figure 3.1: Block diagram showing the whole process involved in localizing a camera using the proposed technique. The arrows show the flow of data through the process. The lighter boxes represent the existing systems used, and the darker boxes represent the contributions of this work. The main contribution of this work is in the localization component, which involves combining available techniques to localize in various environments.

Transformation estimates are obtained using groups of three feature correspondences within a RANSAC framework. The transformation estimates are verified by computing the number of inliers that pass a threshold, and by using an Environment Measurement Model (EMM). The EMM uses information about the sensor's physical properties to discard transformation estimates that are not physically possible. An image is added to the list of *keyframes*, which are the images used to create the map, when significant movement has occurred. The *general graph optimization* method, g^{2}o [1], is used to obtain a globally consistent trajectory. g^{2}o minimizes a nonlinear error function which measures how well the poses match the constraints between them. Any edges containing an error higher than a threshold are removed from the graph. The output of the system is a 3D map of the environment as shown in Figure 3.2.

The open-source RGBDSLAM code was modified[1] to save the map in a format required for the proposed system. Each point cloud in the map, the intensity images associated with the point clouds, and a YAML file containing the transformation matrix describing the position of each point cloud is saved. Each point cloud file makes up a small portion of the map, with some overlap existing between two consecutive point clouds. A 3D map of the environment is reconstructed at a later time by combining all the points clouds together using their transformation matrices.

The idea behind the proposed system is that once a 3D map of the environment is available, localization can then be done using inexpensive, commodity cameras. The system has been tested with maps created using RGBDSLAM, but in the future it could be modified to work with 3D maps created using other techniques. As long

[1]The modification was done previously by Ph.D. student Mirza Tahir Ahmed, prior to the work presented in this thesis.

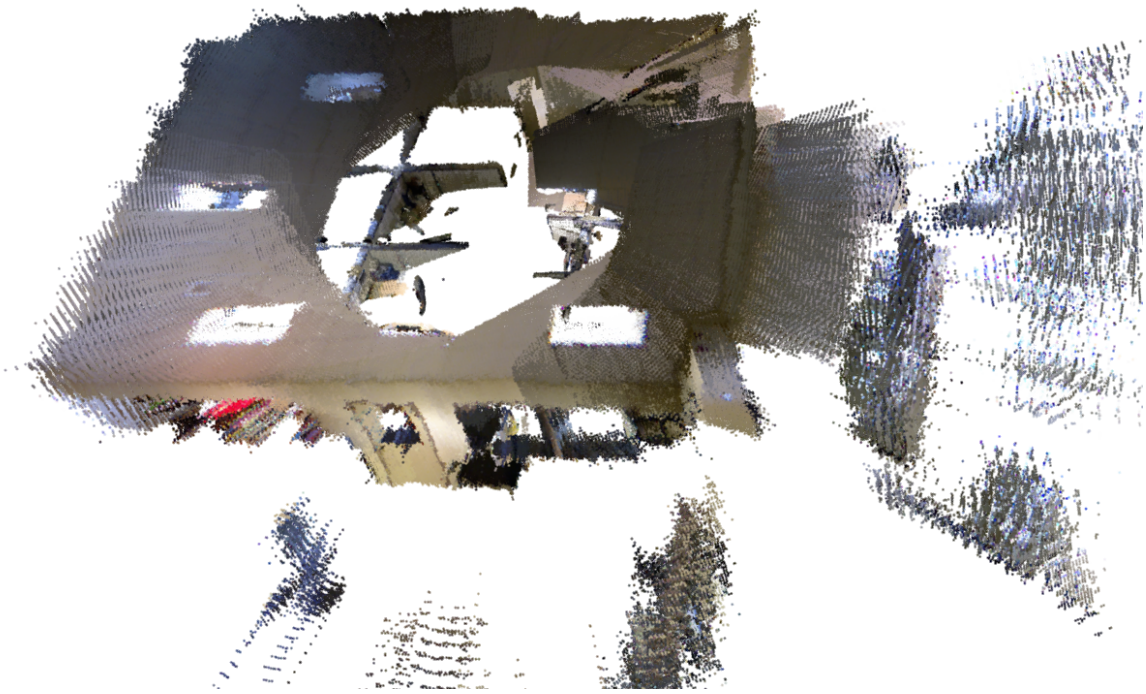


Figure 3.2: Downsampled 3D map created using RGBDSLAM and viewed in the PCL visualizer. The 3D map shows a part of a graduate student office at Queen's University.

as the required inputs (i.e., a set of point clouds, their corresponding images, and a file containing the transformation of each point cloud) are met, the system should be able to reconstruct the 3D map suitable for localization.

3.2 Feature Database

Once the 3D map is available, the feature database is created. The feature database stores descriptors of the features present in the 3D map. At runtime, the query image's descriptors are taken as input and the closest matching database image is efficiently identified using landmark indexing [16]. The closest matching database image is the image that contains the most features that also appear in the query

image. This is done by summing the number of query features that appear in each database image and picking the one with the highest number of votes. The feature database is used in the proposed system to aid with localization. This section briefly describes the feature database presented in [16], as well as the modifications that have been made to it to allow its integration into the proposed system.

The feature database structure is as follows

$$\langle f_i, \langle c_{i1}, c_{i2}, ..., c_{iN} \rangle \rangle, i \in [1, M], \tag{3.1}$$

where M is the total number of features, N is the total number of images, f_i is the i-th feature descriptor, $c_{ij}, j \in [1, N]$ is the boolean value indicating if the feature is present in a particular image. The boolean value c_{ij} is assigned as

$$c_{ij} = \begin{cases} 0 & \text{if } f_i \text{ does not appear in image } j, \\ 1 & \text{if } f_i \text{ appears in image } j. \end{cases}$$

Therefore the feature database provides a table listing each feature once and indicating in which image each feature appears.

Before the feature database can be created, a pre-processing step is needed to prepare the data. Each of the images corresponding to point clouds within the map are processed, and their feature descriptors, along with 2D and 3D features are found. This information is stored and loaded into the database during initialization. The database currently works with SIFT or SURF features, therefore there is the option to use either of these descriptors when finding the features in the images.

The feature database is populated in an iterative process. For the very first image,

all the features present in the image are entered into the database. For all subsequent images, the features are compared against the existing features in the database. If an image feature is matched to a database feature, it already exists in the database, and it is not newly added. However, if a feature is not found in the database, it is added as a new database feature. The boolean value c_{ij} is set to 1 each time a new feature is added to the database, or an existing database feature is found in a new image.

There are certain features in the database that only appear in one image, or alternately, in many images. These types of features are not useful in determining the best matched database images, and take up space in the database, resulting in a slower search time. Therefore, they are eliminated using *landmark indexing* which is already built into the feature database. A landmark is a feature that is repeatable (i.e., appearing more than once and in adjacent images) and distinguishable (i.e., not appearing in more than a certain number of images) [16]. Non-landmark features are removed using band-pass filtering, using an upper, U_{th}, and lower, L_{th}, threshold defining the number of images a feature must appear within. The value for the upper and lower thresholds may change based on the specific dataset. In addition, the upper threshold is selected to be a percentage of the total number of images.

Landmark indexing is performed directly after the features from all the database images have been loaded. Once the database has been created and landmark indexing is performed, all the information within the database is saved into YAML files. During all subsequent uses of the particular 3D map, the YAML file is simply loaded into the database to speed up the initialization time. Therefore the database needs to be created only once per map.

The feature database uses FLANN [63], a nearest neighbour search mechanism,

to find the closest matched feature between a query image and the database features. Once all the query features are matched to the database, the number of matched features found in each database image is summed up. Generally, the database image containing the most matched features is returned as the best matched image. Figure 3.3 shows the plot of the number of votes as a function of the database images. In the plot, there is an obvious peak representing the best matched database image.

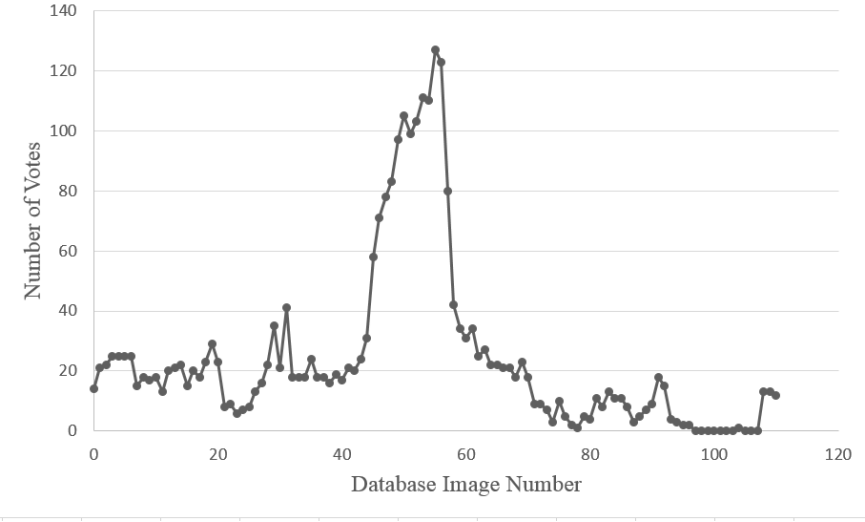
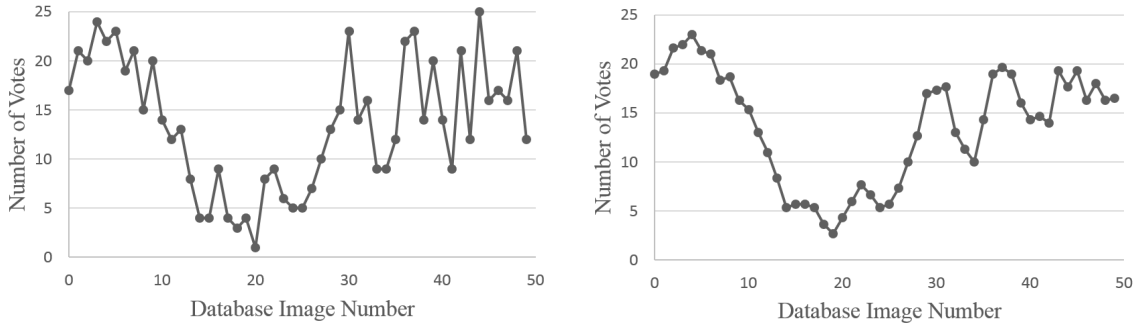


Figure 3.3: Plot of number of votes for database images. The x-axis represents the database image number and the y-axis represents the number of votes for each database image.

Occasionally, false positives may occur, or multiple images may have the same number of votes. An example of this case can be seen in the left image of Figure 3.4, where multiple peaks exist. Database image 44 has the highest number of votes at 25, but database images 3, 5, 30, and 37 also have many votes. Therefore it is not certain that image number 44 is indeed the best database image. However, due to the overlap between consecutive database images, it is known that the neighbouring images closest to the correctly matched image also have a higher number of votes. When multiple

peaks are present, by looking at the number of votes of the neighbouring database images, it is possible to eliminate the peaks that are incorrect. To account for this, a modification was made to the feature database to take a moving average of the votes in the neighbouring images. The image with the highest number of votes after the moving average is returned as the best matched image. In the case of a false positive, the votes in its neighbouring images should be small. Therefore a moving window average, taking into account the immediate votes to the left and right of the best matched image, reduces the likelihood of obtaining a false positive match.



(a) before windowing (b) after windowing

Figure 3.4: Plot of votes for database image before and after windowing. The x-axis represents the database image number and the y-axis represents the number of votes for each database image.

Figure 3.4 shows the effect of the moving window average on the number of votes per database image. The left image is the plot of the votes before averaging, and multiple peaks exist, causing uncertainty about the best matched database image. The image on the right is the plot of the votes after averaging, resulting in an obvious peak indicating the best matched database image. After applying the moving window average it is clear that the best matched database image in this example is image 4.

3.3 Camera Calibration

The monocular cameras need to be calibrated before they can be used to collect the query images. The camera calibration parameters are required by PnP to solve for camera position, and so calibration is a crucial step of the process. Calibration is performed using a checker board pattern and Zhang's technique for flexible camera calibration [64]. During camera calibration, the checker board is placed in the camera's field of view at different angles and distances, and images are captured of the checker board in each position. When adequate images have been captured, the calibration program analyzes the images and returns the camera's intrinsic matrix and distortion coefficients. The calibration is more accurate if there are many images of the checker board pattern in different configurations. Camera calibration must be performed each time the physical parameters of the camera are changed, for example, when the lens is re-focused or the aperture size is adjusted.

3.4 Data Collection

In the current implementation of the proposed system, the data collection and localization are performed at different times. Data is first collected, and then pre-processed before it is ready for localization. The data collection stage involves the creation of the 3D map and the collection of the query images using the calibrated monocular camera.

Data collection begins by running our modified version of the RGBDSLAM system and saving the map components. The RGBDSLAM system comes with its own GUI, which shows the map being built in real-time. During the map building process, the sensor is moved slowly to ensure that enough overlap exists between images to

recover the structure of the sensed environment. The user interface visually shows how many features are found in the current image and if it is added to the map. Sometimes the map building process is interrupted if the sensor is moved too quickly or if there are not enough features in the environment. An example of an environment with few features is a hallway with bare walls lacking texture. If the map building is interrupted, the user must backtrack and point the sensor at a part of the environment that already exists in the map. RGBDSLAM is able to recognise the area and pick up mapping from there.

The query images can be collected using different methods. Some of the query images presented in Chapter 4 were collected with a Logitech webcam, using OpenCV to save each image in a folder, naming them sequentially. The mine datasets were collected with Point Grey Blackfly cameras using ROS, saving the images in a *rosbag* file. These images were then extracted from the rosbag file upon returning to the lab. The current implementation of the proposed system takes images as input instead of video files, because with images it is easier to work on a particular portion of the dataset, and to analyze specific images. More information about the sensors used for collecting the data, as well as the sensor configurations and technique for data collection is provided in Chapter 4.

3.5 Pre-processing

A 3D map can contain from ten to hundreds of point clouds, depending on the size of the environment being mapped and on the overlap between the images. At runtime, these points clouds are visualized using the PCL visualizer. During the visualization process, the points in the point cloud are read, transformed using the

cloud's transformation matrix, and plotted. This is done for all the point clouds making up the map. Since each point cloud contains 307200 points, it takes a very long time to visualize the entire map. Additionally, if the 3D map needs to be oriented in the visualizer, it takes a long time for the visualizer to respond and orient the map due to the large number of data points.

The point clouds are therefore downsampled using a voxelized grid approach in order to speed up the map building process. The voxel size depends on the individual datasets, how distant the features are from the sensor, and the amount of overlap between the point clouds. Therefore trial-and-error may be required to determine the appropriate voxel size. A voxel length of 10 cm was used for all but one of the datasets presented in Chapter 4. The resultant downsampled point clouds contained between 1000 and 10000 points, depending on how close the original points were to each other. Even though the point clouds were downsampled, the key environment features remained visible in the downsampled map. If a larger voxel size was used for the datasets presented in Chapter 4, the risk would be that the downsampled map would contain gaps and that significant environmental features would be lost, as too many points would have been eliminated. Additionally, to increase the speed of reconstructing the 3D map during localization, the downsampled point clouds are also transformed before being saved. Therefore at runtime, the downsampled point clouds are simply loaded into the system and displayed using the PCL visualizer.

In addition to the map, the database is also pre-processed to prepare it for localization. First, the 3D point clouds that make up the map are loaded into the system one by one. These point clouds are organized, meaning that the data is split into rows and columns forming a matrix resembling an image. Therefore, the 2D image is

created by reading the intensity values in each matrix location. The intensity image is passed through the SURF detector. The detected features are extracted and the feature descriptors and their 2D features and 3D object points are saved. This information is saved in three files, which are loaded into the feature database during the database creation stage.

3.6 Initialization

Before localizing query images, the system must be initialized by loading the database, displaying the 3D map and loading the camera's calibration parameters from file. The larger the database, or the more point clouds there are in the 3D map, the longer the initialization process takes. However, this process is done once per run, and does not affect the speed of the localization process, when the query images are localized.

If it is the first time running a certain dataset, then the database must be created during the initialization stage. The database is created by loading in the files containing the descriptors, the 2D features, and the 3D locations of the features, which were created in the pre-processing step. The process of populating the feature database was described above in section 3.2. If the database for a certain dataset has previously been created, then the database is initialized by simply reading in the saved database files. Loading the database from file is much faster than the process of creating the database. The database, which is a k-d tree structure, is then trained and ready for use.

3.7 Localization

The localization process begins when the first query image is obtained by the monocular camera. The query image is processed and goes through the steps shown in Figure 3.5 to find the position of the camera. These steps are described in more detail in this section.

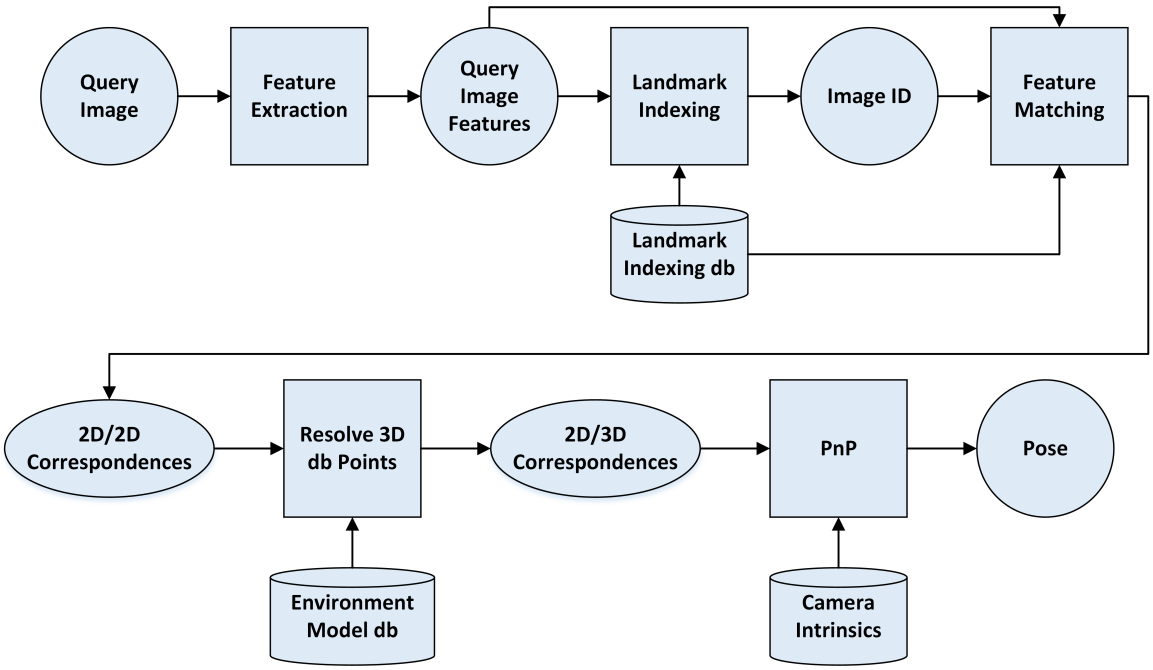


Figure 3.5: Block diagram showing the main steps in the localization process and the flow of data through the different components of the system.

The features in the current query image are detected and extracted to get a list of feature descriptors for the image. These descriptors are passed into the feature database, called landmark indexing, which compares them to the features in the database. The index of the database image containing the most query features is returned as the best matched image. In some cases when the query image is located outside the map, the database will not be able to find a closest match. In this case,

it returns a value of -1 as the best matched index, and the localization process skips to the next query image. It is important to note that the feature database and the query image must use the same type of feature detector, otherwise the feature matches would be incorrect. The localization process uses SURF features because it is faster than SIFT with similar matching accuracy. While other types of feature detectors are available, they are currently not supported by the feature database.

Knowing the index of the best matched database image, the properties of this image are read from file, represented by the environment model database in the block diagram above. In the pre-processing stage described above, all the features, their 2D image locations and 3D locations have been saved to file. Now, this is read from the file and loaded into the program.[2]

The query image features and the features in the returned database image are matched. The brute force descriptor matcher available in OpenCV is used in the current implementation. Since the number of features matched between the query image and the database image are relatively small, on the order of tens or hundreds, the brute force matcher is effective. If necessary, different types of feature matchers may be used in future implementations of the system. Since the query image contains only the 2D image points of the features, knowing the location of the same features in the database image is crucial. The matches are used to find the 3D location of the features, which is available from the database image.

To prepare for PnP, a list of the 3D object points, obtained from the database

[2]Reading large amounts of data from a file is not very efficient, and future work on the feature database should include modifying the database structure to contain this information. That way it could be passed back to the localization system along with the index of the best matched database image.

image, and their 2D image projection in the query image is created. This list of 2D-3D point correspondences, along with the camera's calibration parameters are passed into EPnP, which was described in Chapter 2. Figure 3.6 shows the geometry of PnP during the localization process.

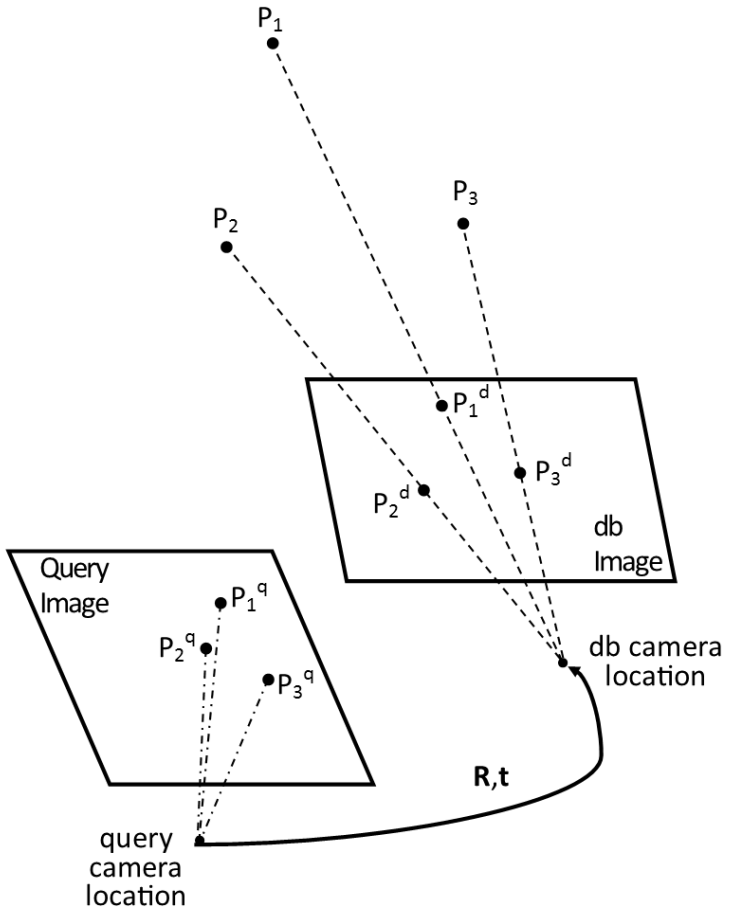


Figure 3.6: PnP geometry of the localization process. The location of the 3D points, P_1, P_2, P_3, and their 2D projection on the database image, P_1^d, P_2^d, P_3^d, are known, and the 2D location of the features on the query image, P_1^q, P_2^q, P_3^q, are also known. The 2D-2D feature matching provides knowledge of the 3D points for the query image. The position of the database camera is known, but the location of the query camera is unknown. PnP provides the rotation and translation that describes the position of the query camera with respect to the database camera.

OpenCV's implementation of EPnP was used, which runs PnP in a RANSAC framework. The function returns a rotation and translation vector that transforms the object points from their 3D coordinate space to the camera's coordinate system. The inverse of this transformation is found in order to obtain the motion of the camera. First the rotation vector is transformed into a rotation matrix using the Rodrigues transform [65]. Then the rotation and translation are combined as shown in (3.2) to get the transformation of the camera from the local coordinate frame of the database image.

$$T = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}. \tag{3.2}$$

PCL's visualizer is used to display the 3D map and the movement of the camera. All the point clouds located in the 3D map are described in the coordinate frame of the very first point cloud in the map. This is referred to as *the map's coordinate frame*. The database provides the transformation matrix to transform each point cloud into the map's coordinate frame. To display the camera's position in the 3D map, the transformation matrix describing the camera position in the local coordinate frame of the matched database image is multiplied by the transformation matrix of the database image.

The PnP function also returns a list containing the indices of the object points that are *inliers*. In order for a point to be an inlier, the following two conditions must be satisfied: First, the distance between the object point and its reprojected point must be less than the specified reprojection error threshold. Second, the z-coordinate of the transformed object point is checked to make sure that the point is in front of

the camera. The reprojection error is calculated by

$$\text{res} = \sum_i \text{dist}^2 \left(\mathbf{A}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} \mathbf{p}_i^w \\ 1 \end{bmatrix}, \mathbf{u}_i \right), \tag{3.3}$$

where $\mathbf{A}$ is the camera's intrinsic matrix, and dist($\mathbf{a}$,$\mathbf{b}$) is the 2D distance between point $\mathbf{a}$ and point $\mathbf{b}$, represented in homogeneous coordinates. While only a minimum of four inliers are required to solve for the camera position, it was found that increasing the number of inliers improved the accuracy of localization. Therefore a threshold for minimum inliers was set, and the camera's position is only visualized if this threshold has been met. The threshold value depends on the dataset and is discussed further in Chapter 4 with the results.

The entire process is repeated for each query image. The visualizer is updated for each query image, so that the motion of the camera is visible throughout the process. The final output of the system is the 3D map with the superimposed trajectory of the camera.

Chapter 4

Experimental Results

The experiments performed on the proposed system and their results are presented in this chapter. All the data used for testing the localization system was collected manually in various environments. Two sets of experiments were conducted. The ground truth experiments were used to compare the accuracy of the proposed system against the position of the images obtained by RGBDSLAM. The localization experiments were used to test the performance of the system in various environments using different cameras. This chapter begins by describing the sensors used to collect the datasets and providing information about all the datasets used for the experiments. It then describes the ground truth experiments, followed by the localization experiments. The description of the localization experiments is divided into two sections. The first section contains all indoor datasets, and the second section contains the datasets collected in underground mines. The chapter concludes with a discussion of the runtime of the system.

4.1 Sensors

The Microsoft Kinect (version 1) was used to collect the RGB-D data for all the datasets. The Kinect captures data at a maximum rate of 30 frames per second (FPS) with a resolution of 640×480 in an organized point cloud. To obtain the depth information, the Kinect's infrared (IR) emitter radiates from the sensor in a specific pattern. The rays are reflected off of the objects and sent back to the IR depth sensor. The depth sensor then determines the depth of the objects based on the disturbance of the received pattern [66]. The Kinect operates at a range of 0.8 m to 3.5 m. Any range closer or further than this results in an out of range (i.e. NaN) depth value.

Two different cameras were used to capture the query images for localization: the Point Grey Blackfly camera (model BFLY-PGE-13E4C-CS) and a Logitech Webcam C930e. The Point Grey Blackfly camera had a resolution of 1280×1024 and a maximum frame rate of 60 FPS, and was used to capture all the query images in the underground mine datasets. Many of the parameters of the Blackfly camera could be manually set, which was crucial for successfully collecting images in these dark environments. The Logitech webcam was used to collect all the query images in the indoor datasets. The resolution of the Logitech camera was 640×480, with a maximum frame rate of 30 FPS.

4.2 Datasets

The indoor datasets were all collected in buildings at Queen's University. Appendix B contains a selection of images from each of these datasets to provide the reader with a better idea of what these environments look like. The datasets were organized

as follows,

- *Office* - The Electrical & Computer Engineering department office, containing the reception and offices.
- *Lab* - A graduate student lab, containing computers and desks and lab equipment.
- *Office712* - A graduate student office made up of cubicles.
- *JDUC* - A large open area containing couches, a store, and the offices of different university clubs.
- *WLH* - The lobby of Walter Light Hall.

Table 4.1 contains more information about these indoor datasets, such as how many images were used to make the map, the number of features stored in the database, and the approximate size of the areas. Columns 2-4 provide information about the 3D map and the database. The last column describes the camera used for collecting the query images.

Table 4.1: Indoor dataset properties. All query and database images in these datasets had a resolution of 640×480.

Dataset	# Frames in Map	# Database Features	# post-Landmark Indexing	Approximate Area	Camera Used
Office	50	5930	3886	139 m^2	Kinect
Lab	33	12842	6437	37 m^2	Kinect
Office712	111	10918	8766	70 m^2	Logitech
JDUC	204	6987	5770	257 m^2	Logitech
WLH	167	10575	8608	130 m^2	Logitech

The mine datasets were collected at two different locations. The first location was at the NORCAT Underground Training Centre near Sudbury, Ontario. The data was collected in a dark area of the mine, using 1600 Lumens LED Worklights, one per camera, to illuminate the walls. A total of four Point Grey Blackfly cameras and two

Kinects were used to collect the dataset. The sensors were mounted onto a wood panel which was placed on top of a trailer. The trailer was driven around the mine by a Rugged Terrain Vehicle (RTV). The four cameras were located at each corner of the wood panel, looking at the wall at a 45° angle and looking up at a 45° angle. The Kinects were placed at the back corners of the wood panel, also looking out and up at 45°. A picture of the data collection system is shown in Figure 4.1. The data was collected using the Robot Operating System and processed upon returning from the trip. This dataset is referred to as the NORCAT dataset.

Figure 4.1: Version one of the vision data collection system used at the NORCAT Underground Training Centre.

The Hemlo datasets were collected at the Barrick Gold Hemlo site Northern Ontario. This round of data collection focused on collecting good quality images and maps of the mine, therefore only one Point Grey Blackfly camera and one Kinect was used. Five 1600 Lumens LED work lights were used to illuminate the walls. Both the camera and the Kinect were set up to face the ceiling of the tunnel, as can be seen in

Figure 4.2: Second version of the vision data collection system, used at the Hemlo underground mine.

Figure 4.2. The image shows that two other cameras were also part of the setup, facing outwards at the walls. These cameras were used for performing other experiments while at the mine, but all the data presented in the results was captured using the top facing camera. The camera setup for the Hemlo datasets are different from the NORCAT dataset because upon return from NORCAT, it was discovered that the side walls of the tunnel are not a good place for collecting data. More information about analysis of the data collection technique and improvements made is provided later in this chapter along with the results from these datasets. Two datasets were used from the Hemlo data collection trip, which are referred to as Hemlo1 and Hemlo2. A summary of the mine datasets is provided in Table 4.2, including the length of the area that was mapped.

Table 4.2: Mine dataset properties. All query images had a resolution of 1280×1024, and all database images had resolution of 640×480 in these datasets.

Dataset	# Frames in Map	# Database Features	# post-Landmark Indexing	Approximate Trajectory	Camera Used
NORCAT	194	13584	11178	30 m	Point Grey
Hemlo1	1148	170300	147741	202 m	Point Grey
Hemlo2	1131	248436	216150	175 m	Point Grey

4.3 Ground Truth Experiments

The accuracy of the proposed localization technique was tested by comparing the results against the camera positions obtained by RGBDSLAM. All the images that are part of the 3D map of the environment have a transformation matrix that describes their position. These transformation matrices were created by RGBDSLAM during the map building process. By using a subset of these database images as the monocular query images, a second transformation matrix was obtained for each image. This second transformation matrix is an estimate of the camera's position, calculated using the proposed technique. Using RGBDSLAM's transformation matrix as a ground truth, the position obtained by the proposed technique can be directly compared with the reference position, and the accuracy of the technique can be tested. This experiment, which is referred to as the *ground truth experiment*, was performed on the Office and Lab datasets. Some of the results and figures that appear in this chapter have been published in [67].

The two transformation matrices were compared by looking at their translation. The position of the camera is described by the translation vector, and the orientation is described by the rotation matrix, therefore the focus was on the translation component. The difference in translation between the calculated and the reference

position was found using the Euclidean distance formula

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2 + (q_z - p_z)^2} \tag{4.1}$$

where $\mathbf{p}$ is the position of the camera provided by RGBDSLAM and $\mathbf{q}$ is the position of the camera recovered through the proposed system.

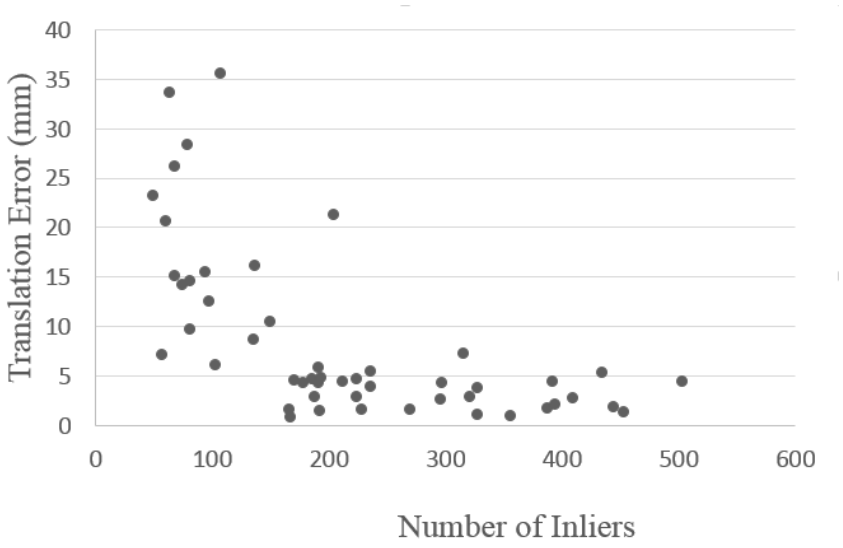


Figure 4.3: Plot of translation error as a function of the number of inliers for the Office dataset.

The translation error between the proposed technique and the reference position for the Office data set ranges between 0.9 mm to 35 mm. By plotting the translation error as a function of the number of inliers present in the image, such as in Figure 4.3, it can be seen that the more inliers there are, the better the accuracy of the system. A similar pattern is seen for the Lab dataset, which has translation errors ranging from 0.3 mm to 17 mm. The results of the ground truth experiment on the Lab dataset can be seen in Figure 4.4. The minimum inlier threshold was set to 100 for the Lab dataset because it is rich in features. Using this threshold for minimum

inliers eliminates three of the data points, which have a large translation error. These points are outliers, because they are similar distances from the camera, resulting in an inaccurate camera position and therefore would skew the plot.

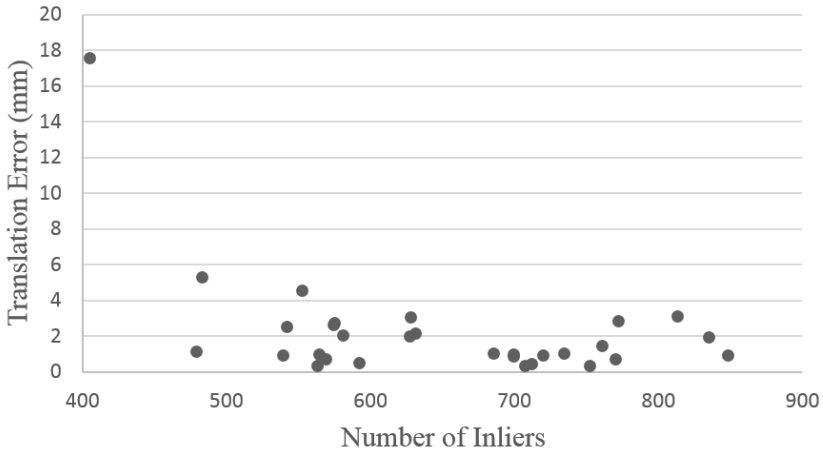


Figure 4.4: Plot of translation error as a function of the number of inliers for the Lab dataset.

If the inliers are mostly the same depth away from the camera, they cause large errors in the calculated camera position. It is difficult for PnP to determine information about the depth when all the reference points are coplanar. Figure 4.5 shows one of the images from the Office dataset that had a localization error of 21 mm. The query image is on the left side of the figure, and the database image is on the right side. The small coloured circles on both images represent the location of the image features, and the lines at the bottom of the figures represent the matched inliers in both images. In Figure 4.5, all the features determined to be inliers by PnP are located in the lower half of the image, and most of the features are coplanar.

Similarly, Figure 4.6 shows a set of images from the Lab dataset that had a high translation error. This image had 65 inliers and an error of 1697.3 mm, which

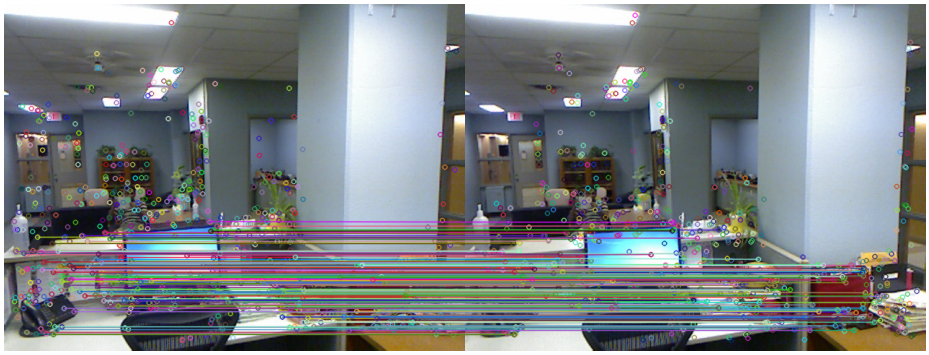


Figure 4.5: PnP inliers between the query (left) and the database (right) image from the Office dataset. The coloured lines match the inliers between the two images. This image had one of the higher translation errors in the ground truth tests.

was the largest for the Lab dataset. This data point was excluded from the plot in Figure 4.4 because it was a gross outlier. Even though the image in Figure 4.6 contains many inliers, these inliers are clustered in one part of the image and are once again similar distances away from the camera. This does not provide PnP with a large variety in feature locations and therefore affects the accuracy of the position estimate. However, even with the large error in the calculated position, the proposed technique still determined that the camera was located within the 3D map.

Images where the inliers are distributed more evenly throughout the image tend to have smaller localization errors. One example of such an image is shown in Figure 4.7 from the Office dataset. This image had a translation error of 0.9 mm, and it can be seen that the inliers are dispersed throughout the entire image.

The output of the proposed localization system is shown in Figure 4.8 for the Office dataset. A subset of thirteen images were localized in the figure, all of which were captured while walking through the office. These thirteen images are also present in the 3D map, and their reference position, calculated using RGBDSLAM, is known.

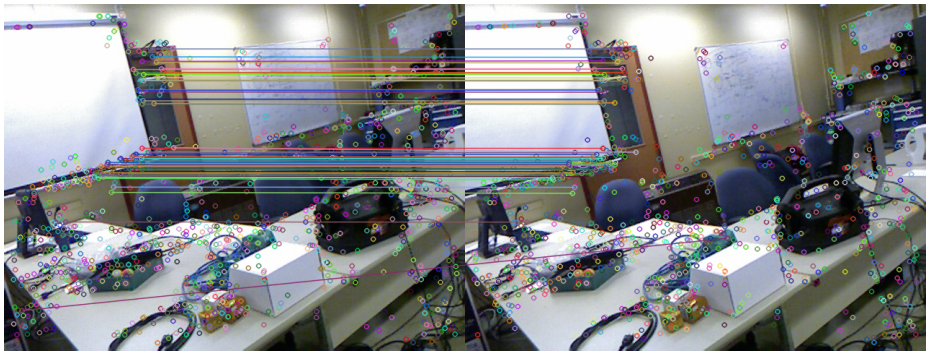


Figure 4.6: PnP inliers of one of the images from the Lab dataset, showing the query image on the left, and the database image on the right. This image had the largest error in the calculated position out of all the images used in the ground truth experiments.

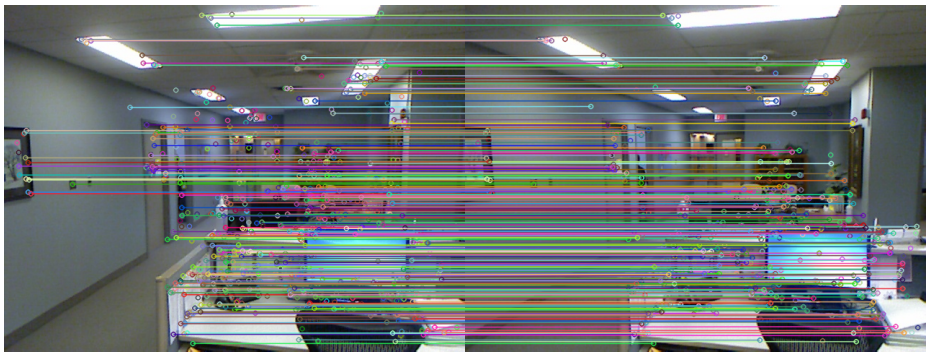


Figure 4.7: PnP inliers of one of the frames of the Office dataset, showing the inliers for an image with a small localization error. The query image is located on the right and the database image is located on the left.

Two different coloured spheres are used to show the position of the camera. The green spheres represent the reference position as calculated by RGBDSLAM, and the red spheres represent the camera position calculated by the proposed technique. The camera position is located at the centre of the spheres, but the size of the spheres were chosen to be large enough to be visible in the figure. Since the red and green spheres occupy the same area in the image, they appear to be within one another. Due to this, in cases where the calculated position is less than 1 mm away from the

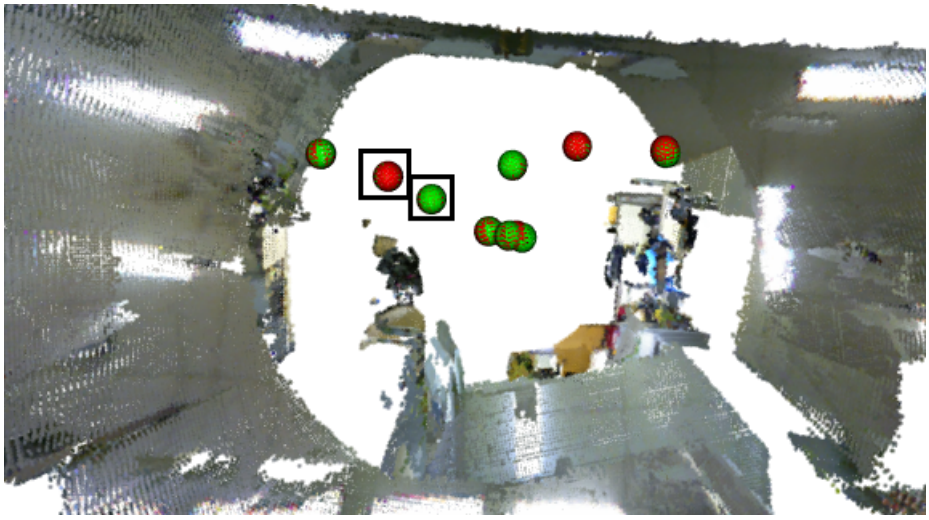


Figure 4.8: Camera position visualized in the 3D map during the ground truth experiment on the Office dataset. Each sphere represents the position of the camera at the time of image capture. The red spheres represent the position calculated using the proposed technique, while the green spheres represent the reference position provided by RGBDSLAM.

reference position, the spheres exhibit a single colour. These spheres are highlighted with a black square in Figure 4.8. For some of the positions it can be seen that the spheres are offset a little (containing both a red and green colour), which shows the translation error discussed above. However since most of this error is in the order of millimetres, a large offset of the spheres is not visible.

4.4 Localization Experiments

Localization was performed on the remaining indoor and mine datasets. In the following experiments, the query images used for localization were collected with a different camera than was used for the database images. In the case of the mine data, the resolution of the query and database images were different as well, adding a further challenge to the system. The results in this section demonstrate that the proposed

localization system is not constrained to a particular camera. As long as both the query and database images are clear and feature rich, the proposed localization system works. Some of the results presented in this section, mainly the indoor datasets, have been published in [67].

4.4.1 Indoor Datasets

Office712

The *Office712* dataset was taken in an area of a graduate student office that contained two sets of cubicles. One set contained larger cubicles in a 3×2 arrangement and the second set contained smaller cubicles in a 4×2 arrangement. The divider between the cubicles can be seen in the 3D map. The query images were captured in three of the large cubicles by slowly rotating the camera 360° while standing at the centre of each cubicle. Emphasis was placed on collecting the query images in areas of the room that were near objects appearing in the map, in this case, within the cubicles. Doing this made it possible to visually verify whether the calculated camera position was correct. Since localization was performed using only a single camera relative to the 3D map, it was difficult to obtain a ground truth for the actual camera position. However by performing localization near objects that appeared in the 3D map, it was easy to visually determine if the localization was correct. A minimum inlier threshold was set to ten, and only the camera positions of images containing inliers above this threshold were visualized.

The calculated camera positions for the query images captured in cubicle 1 can be seen in Figure 4.9. A total of 152 query images were localized, with each sphere representing the position of one of the query images. The green spheres in this

figure represent acceptable positions, as they are within the cubicle where the data was captured. The red spheres represent outliers, as these locations are outside the cubicle, and therefore not near where the images were captured.

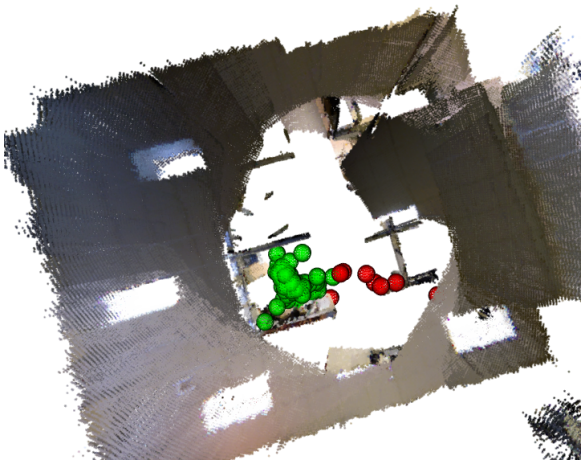


Figure 4.9: Calculated camera positions for query images captured in cubicle 1.

As mentioned before, the query images were collected from each cubicle by moving the camera around 360° from the centre of the cubicle. The camera positions calculated by the proposed localization system were averaged to get one camera position per cubicle. The average position of the camera in each cubicle can be seen in Figure 4.10, represented by the green sphere. A large sphere was chosen to make sure that the camera's position is visible in the image. The calculated camera position is located at the centre of the sphere. The average position for cubicle 1 and cubicle 2 is near the centre of the cubicle, and is located within the boundary of the cubicle, which is visible on the map. For cubicle 3, the average position is at the boundary of the cubicle, near the divider. The query images were collected in the centre of cubicle 3, so it can be seen that some error exists in the average position for cubicle 3.

(a) Cubicle 1 (b) Cubicle 2 (c) Cubicle 3

Figure 4.10: Average localization results for query images taken in three cubicles of the Office712 dataset.

To provide more information about the number of images localized and the distribution of the average camera position, Table 4.3 has been included. As can be seen from this table, not all the query images were localized because some of the query images did not meet the minimum inlier threshold. These query images did not contain enough good features to calculate a correct camera position. Of particular interest are the last three columns showing the standard deviation of the positions in the x, y, and z components. It can be seen that the localization is more accurate for cubicle 1, and the accuracy decreases as it moves to cubicle 3. One reason for this is that cubicle 1 and cubicle 2 are located closer to where the database images were collected. The database images and the map were collected in the walkway between the 3×2 cubicles and the 4×2 cubicles, near the divider between cubicle 1 and cubicle 2. Therefore the query images collected in cubicle 1 and 2 are more similar in distance and orientation to the database images than the query images captured in cubicle 3. More feature matches are present among similar images than between images taken at a different angle.

Table 4.3: Standard deviation of the camera's average position calculated in the Office712 dataset

Cubicle Number	Total Images	Images Localized	Percent Localized	Standard Deviation (mm)		
				x	y	z
1	200	152	76	478.825	345.871	171.33
2	150	99	66	619.23	376.961	468.211
3	155	110	71	1382.7	794.103	613.596

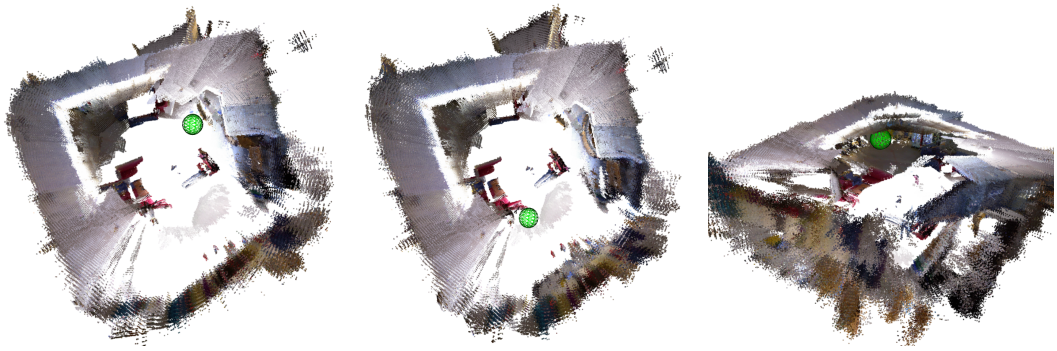
Another reason that cubicle 3 has the least accurate position estimate is that cubicle 3 is located right next to a wall. In this case the wall was bare, therefore containing very few features. On the other hand, cubicle 1 was located close to a feature-rich wall containing three bookshelves filled with books and other objects. In addition, the third wall of the office had five windows, which all look similar. Therefore having cubicle 3 further from the feature-rich environment makes it harder to localize and increases the error in the calculated position. This source of error is in the localization, as it is harder to match the features between the query image and the database image. Therefore fewer feature matches are input into PnP, and fewer inliers are used to calculate the camera's position.

JDUC

The JDUC dataset was collected on the lower floor of the John Deutsch University Centre at Queen's University. This is a large open space containing couches in the middle and surrounded by offices and a store. About half of the area is covered by a ceiling, while the other half is open to the upper floor. One of the corners of this environment was made up of a staircase and a stone wall. The stone did not contain many features, causing difficulty in both the map building and localization stages.

The 3D map was created by standing near the middle of the room and moving the

Kinect around 360°. Some areas, such as the corner with the staircase and the stone wall, were challenging for map building as they did not contain many features. The Kinect had to be moved through this area very slowly multiple times until enough features were detected in the images to be able to add them to the map. Another challenge faced when building the map was that, due to the size of the area, some of the boundaries and objects were beyond the range of the Kinect and therefore 3D data was not always available.



(a) Corner 1 (b) Corner 2 (c) Corner 3

Figure 4.11: Average localization results for query images taken in three corners of the JDUC dataset.

The query images were captured in three corners of the JDUC environment. The query images were also collected by slowly moving the camera around 360°. Figure 4.11 shows the average localization obtained in the three corners of the JDUC dataset. The green sphere represents the averaged camera position for each corner. Table 4.4 provides numeric results from the experiment, showing how many of the query images were localized in each corner, and the standard deviation for each corner.

Table 4.4: JDUC dataset average localization standard deviation

Location	Total Images	Images Localized	Percent Localized	Standard Deviation (mm)		
				x	y	z
Corner 1	229	98	42.8	933.654	1106.9	463.618
Corner 2	293	118	40.3	1058.77	1480.5	923.451
Corner 3	345	47	13.6	442.301	866.931	512.348

Corner 1, located at the back right corner of the room (when viewing the map from the top), was the closest to the location where the 3D map was created. Of the 229 query images, 98 were localized. Most of the localized images were of the back and right walls, which were the two closest walls to the camera location. Very few images of the front and left walls were localized, and if they were, these images were more likely to produce an incorrect localization result when compared to the images of the back and right walls.

Corner 2, located near the bottom left corner of the room, had 118 of the 293 query images localized. The localized query images collected from this location were mostly images of the front wall, containing many features, or images of the back left corner, which had a set of vending machines. The query images of the other parts of the room did not contain enough inliers to be localized. Additionally, corner 2 was located right next to a column, which obstructed important environment features in a number of the query images. Corner 2 had a larger standard deviation than corner 1, which can be explained by the fact that corner 2 was located further from where the map was built.

Corner 3, located in the back left corner, had 47 of the 345 query images localized. All the localized images were of the back wall, which was the closest wall to corner 3. The back wall contained a set of two vending machines which were very rich in

features, as well as a large poster. The other end of the back wall had recycling bins and a counter for a student run WalkHome service. These areas were rich in features and made it easy to localize. For the remainder of the query images taken from this position, there were not enough inliers to pass the minimum inlier threshold and to obtain a correct position for the camera. Lastly, of the three positions where query images were collected, corner 3 was the furthest away from the location where the map was built.

One of the challenges of localizing in the JDUC environment was that similar features appeared in multiple locations of the environment. One such feature was a set of garbage cans and recycling bins, with a poster above them indicating what waste goes into each bin. There were two such set-ups of recycling bins within the mapped environment, one near the back right corner, and one at the front right corner. From where the map was created, the bins in the first location were clearly visible, while the bins in the second location were partially obstructed by the stairwell. From the location where the corner 2 query images were collected, the bins in the first location were not visible because they were obstructed by a column, but the bins in the second location could be clearly seen. The query images from corner 2 containing the recycling bins in the second location were matched to the database images containing the recycling bins in the first location. This is obviously a wrong match, as the 3D location of these features are different. Passing in these wrong 3D coordinates to PnP resulted in an incorrect calculation for the camera position. Therefore having multiple similar features in the environment poses a challenge to localization using a single camera. Other examples of similar features experienced in the JDUC dataset included identical signs posted in different locations, identical

benches and couches in the centre of the room, and sets of windows.

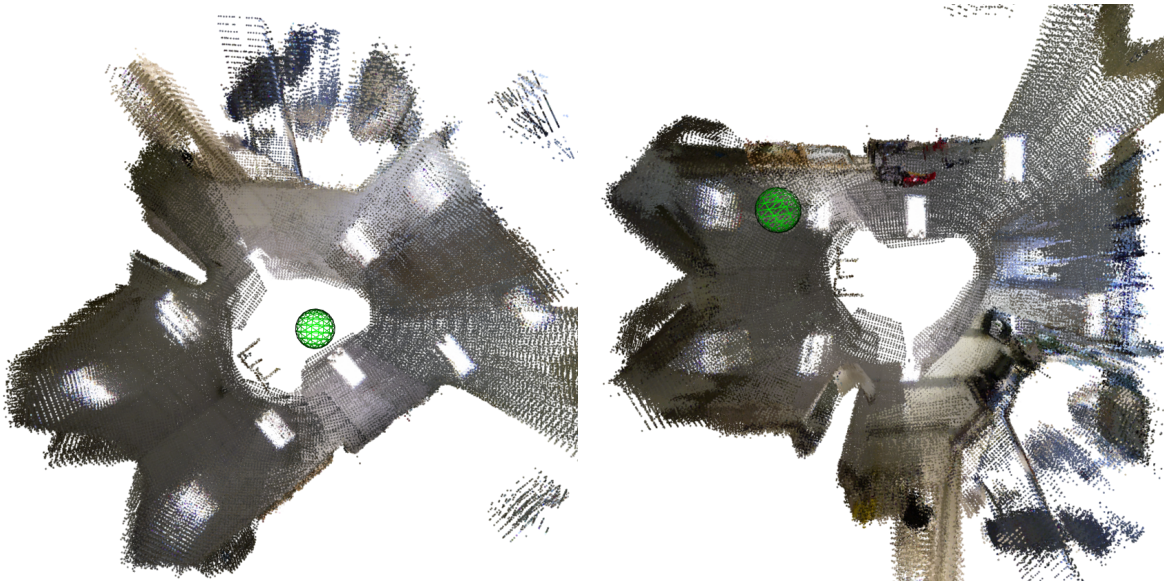
Since the JDUC was a much larger area than the Office712, the results from this dataset show that localization works better in areas where the features are closer to the camera. For all three corners of the JDUC dataset, most of the localized query images were images of the nearby walls or objects. Not many of the query images taken of features further from the camera had enough inliers to be localized.

WLH

The WLH dataset was collected in the lobby of Walter Light Hall at Queen's University. The lobby leads to two separate hallways, two doors of an auditorium, a classroom, a computer lab, and a wall of glass doors separating the main stairway of the building and lobby. A display case was located at the centre of the lobby, and some couches were placed by the wall separating the lobby from the computer lab. One wall of the lobby had a vending machine and some posters on display. The lobby's walls were made up of grey brick and did not contain many features. A few images from the WLH dataset have been selected and included in Figure B.5 in Appendix B to provide the reader with a better idea of the environment. The frames used for building the map were collected by standing in the centre of the lobby, right next to the display case and slowly rotating the Kinect 360° around.

The query images used for localization were captured in two locations which are referred to as position 1 and position 2. Position 1 was at the centre of the lobby, near the location where the 3D map was created. Position 2 was located right in front of one of the doors to the auditorium, next to the wall of posters. As with the other experiments in this section, the query images were captured by slowly moving the

camera around in a 360° motion. The average position computed by the proposed system for both locations is shown in Figure 4.12. Please note that the second image in this figure is shown upside down in order to make the sphere visible, as it would otherwise be covered by the ceiling of the 3D map.



(a) Position 1 (b) Position 2

Figure 4.12: Average localization results for query images captured in two locations of the WLH dataset.

In position 1, 117 of the 300 query images were localized. Most of the localized images were either images looking at the computer lab, the front stairway of the lobby, or at the vending machines and the posters. The other areas of the lobby did not contain enough inliers to be localized. Only 34 of the 290 query images captured in position 2 were localized. Most of the localized query images were of the computer lab or the front entrance of the building. Since position 2 is much further away from where the database was captured, the query images and the database images view the

same features from significantly different angles, leading to fewer feature matches, and ultimately the localization of fewer query images. Localization is challenging in the WLH dataset due to the limited number of features available in the environment. The standard deviation of the averaged camera locations for both positions are presented in Table 4.5.

Table 4.5: WLH dataset localization standard deviation

Location	Total Images	Images Localized	Percent Localized	Standard Deviation (mm)		
				x	y	z
Position 1	300	117	39	1087.92	508.827	719.084
Position 2	290	34	11.7	2870.82	2005.97	1707.77

4.4.2 Mine Datasets

This section presents the results of the localization experiments on the mine datasets. A discussion on the challenges of using cameras underground and lessons learned from collecting data in underground mines is also included.

The terminology used in this thesis may not be consistent with the terminology used in the underground mining environment. In mining, the main tunnels in underground mines are called *drifts*. However, in this thesis the term drift is used describe the accumulated error in measurement over time which affects the accuracy of the 3D map. Therefore the mine drifts are referred to as tunnels to prevent any confusion between the two terms.

NORCAT

For the NORCAT dataset, the 3D maps collected using the two Kinects were combined to create a partial reconstruction of the mine. Since there was no overlap in the field

of view of the Kinects, the geometry of their mounting location on the rig was used to position the two map segments together. Figure 4.13 shows the map of the tunnel collected at NORCAT, as well as the trajectory of the vehicle, which was calculated by the proposed algorithm. From the figure it may be observed that the walls of the tunnel appear to move apart over time. This is because combining the maps of the two segments together after the completion of the individual map building phases is not the most accurate solution. Ideally the mapping process should be modified in order to include the data from both Kinects into one map while it is being built, ensuring that the input of both Kinects are used to create one global map.

Figure 4.13: Calculated trajectory of the vehicle in the NORCAT dataset.

For all the mine datasets, the trajectory of the camera was represented using a red line. Since the camera was moving through the tunnel, rather than in a 360° motion as was the case for the indoor datasets, representing the camera movement using a line is more appropriate. The query images captured by the Point Grey Blackfly cameras that were facing the wall at the same angle as the Kinects were used. Therefore one camera was localized using one half of the map, and the second camera was localized

using the other half of the map. The vehicle's position was found for each instance of time when both camera's query images could be localized. The vehicle's position was taken to be the centre of the distance between the two query cameras. There are instances in time when only one of the two cameras could be localized. In this case, the position of the vehicle was not calculated since the position of the second camera was not available. A minimum threshold of six inliers was used for the NORCAT dataset.

This was the team's first experience collecting data using vision sensors in an underground mine, which was a particularly challenging environment due to the lack of lighting. There was no mine-like environment within the geographic proximity of Queen's University, so the cameras could not be tested in a similar environment prior to visiting the mine. The sensors were tested in the lab at night time, but the darkness in the lab at night is not comparable to the darkness in an underground mine. Collecting and analyzing the NORCAT dataset was a very important learning experience. Three main challenges were faced using visual sensors in underground mines.

The first challenge of using cameras in underground mines is that there is no light present in the environment, making a portable lighting source necessary. One portable light was used per camera to illuminate the rock surface during the NORCAT data collection. It was important to choose an external lighting source that did not create a spotlight effect in the image. Spotlight effects cause the features in the image to be clustered in the spotlight, causing the rest of the image to be dark and featureless. The ideal lighting source is diffuse, so that the entire field of view of the camera is illuminated evenly to ensure that features are present throughout the

entire image. The floodlights used during data collection were good, but having only one per camera was not sufficient. This was discovered once the Kinect and Blackfly images were analyzed and the images were found to be dark and blurry. The blurry images were caused by the camera moving while the shutter was still open. A faster shutter speed is required to reduce the blurriness, ensuring that the camera's motion is minimal while the shutter is open. Due to the dark and blurry nature of the captured images, not many features were detected in the NORCAT dataset.

The limiting factor for the small size of the map was the dark and blurry nature of the dataset. The map could only be built of the area shown, as after that point the vehicle moved faster, deteriorating the quality of the captured images. RGBDSLAM was no longer able to match the point clouds in the subsequent images to the existing map because it could not find feature matches between the images. Blurry images also caused a problem during localization because adequate features were not present in the query images.

The second challenge is that the angle of the query and database cameras facing the wall has an impact on the image quality, resulting in more accurate localization if the two cameras angles are identical. As mentioned in section 4.2, four Point Grey Blackfly cameras were used to collect the query images, one per corner, facing out at 45° at the tunnel walls, and the two Kinects were placed at the back two corners of the trailer. The vehicle was moving through the tunnel counter-clockwise in the dataset used for building the 3D map. It was noticed that when the vehicle was moving counter-clockwise through the tunnel, out of the query images collected using all four Blackfly cameras simultaneously, only the query images from the two cameras located at the back of the trailer could be localized. These two Blackfly cameras were

those facing the wall in the same direction as the two Kinects. On the other hand, for the same 3D map, when the vehicle was moving through this area in a clockwise direction, the query images captured by the two front Blackfly cameras could be localized, but the query image from the two back cameras could not. This shows that the query images and the database images should be looking at the features from a similar direction in order for localization to work effectively.

A possible reason behind this finding is that for the mine set-up, the lights used to illuminate the tunnel walls were placed at the same angle as the cameras in order to light up the entire camera's field of view. However, this means that taking an image of the wall with the camera and light facing the wall at one angle, and taking a second image with the camera and light facing the wall 90° apart results in substantially different features viewed in the image. In a well lit environment the shadows around the features are more uniform since the light comes from many directions. However in the mine, the direction of the light illuminating the features is completely different in the two images, changing the way the feature looks. It is not feasible to have the lights illuminating the features from different directions because that would mean that the lights would have to be placed quite far apart. Since the mapping and localization is performed through a large portion of the tunnel, the sensor set-up should be fairly compact to be able to move with the vehicle.

The third challenge with mapping and localizing in a mine is that most of the tunnels contain cross-cuts. Cross-cuts are horizontal passageways, nominally at 90° angles to the main tunnel, which provide access to mining operations. The lights are unable to illuminate cross-cuts when passing by them, creating large, dark, featureless areas in the captured images. Additionally, cross-cuts extend for a large distance, out

of range of the Kinect's 3D sensor. Mapping is interrupted when a cross-cut is passed, and localization cannot be performed in these areas. Therefore, the side walls of the tunnel are not a good location for collecting the data required for map-building and localization due to the presence of cross-cuts.

The knowledge obtained from collecting the NORCAT dataset allowed for improvements to be made to the data collection technique used in underground mines. These improvements resulted in a successful data collection at the Barrick Hemlo mine in Northern Ontario. Additionally, running the proposed localization technique on the NORCAT dataset provided a very good test for the limitations and challenges of using the proposed system in underground mines.

Hemlo

The second mine dataset was collected at the Barrick Hemlo underground mine in Northern Ontario. The data collection process and experiments were improved based on the lessons learned from the NORCAT data collection trip. Two areas of the Hemlo mine were mapped. The first one is an L-shaped map of part of the main tunnel turning into a perpendicular tunnel, and has been named *Hemlo1*. The second dataset, called *Hemlo2*, is of part of a ramp leading to an upper level of the mine.

The map building was done while underground for both Hemlo datasets to ensure that a significant part of the tunnel could be mapped. The 3D map is critical for the localization process as the proposed system would not work otherwise. While mapping underground, it was noticed that the computer would freeze after approximately 1100 keyframes were added to the map. For each image added to the map, many features are stored in memory, and the memory would be exceeded at such a high number of

keyframes. This was the limiting factor for the size of both the Hemlo1 and Hemlo2 maps, which are approximately 202 m and 175 m respectively.

The trailer had to be pulled very slowly during mapping to make sure that adequate feature matches existed between the existing 3D map and the current image. The Hemlo1 map was built while pulling the trailer manually, while the Hemlo2 map was built by dragging the trailer behind the RTV moving at its slowest speed. If the Kinect had moved too much from its previous location and the current image could not be added to the map, then the trailer would need to be reversed until the Kinect was once again in a location that already existed in the map. Therefore the map building process involved a lot of moving back and forth.

The first challenge for localization in underground mines was dark and blurry images. The dark images are a result of not having enough lighting, while the blurry images are due to moving the camera while the shutter is open. To overcome the lack of lighting available, a total of five lights were used to illuminate the field of view of one camera. Five lights were used to make sure that enough features could be seen in the image. Due to limited amount time in the underground mine, the optimal number of lights was not determined, therefore it is possible that four or three lights may provide adequate lighting as well. Determining the minimum number of lights required per camera and the optimal location of the lights would be part of the future work on this topic.

Since the camera parameters are crucial to collecting good quality query images in dark environments such as underground mines, a set of experiments were performed to find the optimal parameters. These experiments were performed for three different vehicle speeds (5 km/h, 10 km/h and 15 km/h) using trial and error, and the final

values were saved for future use. During the experiments, the lighting configuration was kept constant, using five lights to illuminate the camera's field of view. The *aperture size* was also kept constant, using the maximum opening. The aperture size determines how much light enters the camera, with a larger opening allowing for more light into the sensor. The downside of having a large aperture size is that the likelihood of having blurry images increases. The *exposure* value sets the exposure time for the image, and the *shutter speed* determines how long the camera's shutter is open. Reducing the shutter speed reduces the motion blur, but it also causes the captured images to be darker because the shutter is open for a shorter amount of time, limiting the amount of light that reaches the imaging sensor. The exposure value did not have a visible effect on the quality of the captured images, therefore it was kept consistent for all three vehicle speeds. After the shutter speed was set, the gain and brightness values were adjusted to make the captured image brighter. The *gain* is the amount of amplification applied to the pixels. Increasing the gain brightens the captured image, but also increases the amount of noise present. Lastly, the *brightness* parameter is used to brighten the overall image by controlling the level of black in the image [68]. These camera parameters were adjusted, the trailer was moved for a short distance at a particular speed, and then the captured images were visually analyzed. Based on the quality of the images, the parameters were adjusted and the steps were repeated until the images were clear. The optimal camera parameters obtained through these experiments are presented in Table 4.6.

Table 4.6: Point Grey Blackfly camera parameters used for collecting the query images in the Hemlo underground mine. The frame rate was kept constant at 30 FPS, the aperture was set to the maximum opening, and five lights were used to illuminate the camera's field of view.

Speed (km/h)	Exposure (EV)	Shutter Speed (ms)	Gain (dB)	Brightness (%)
5	0.250	25	5	0
10	0.250	5	15	30
15	0.250	3	15	30

To deal with the other two challenges mentioned in the previous section, the data was collected by pointing the cameras straight up at the top of the tunnel. In mining terminology the top portion of the tunnel is called the *back*, and is referred to as such for the remainder of the thesis. The back is a good place to perform localization because it is usually a constant distance away from the cameras for the majority of the time making it easier to illuminate, and it contains no cross-cuts. The back may also have a number of features such as pipes, ventilation tubing, and mesh. Even if the mine has movable objects such as equipment or crates parked in different locations, they are less likely to be seen when looking at the back. Therefore there is a smaller likelihood that this area of the tunnel would experience major change between the time of map building to the time of localization, making this a more consistent area of the mine.

The Hemlo1 dataset was collected on the 9975 level of the mine. Most of the mapping was done in the main tunnel of the mine, with a small portion of the map taken from a second tunnel, which was perpendicular to the main tunnel. Figure 4.14 shows the map of the tunnel with the calculated trajectory of the camera represented by the red line. The map shows the back of the tunnel because only one Kinect was used for creating the map. There is some distance between the calculated trajectory

and the 3D map. This is expected, because the 3D map represents the location of the rock surface, while the trajectory represents the the position of the camera. The camera was located at a height of 1.5 m from the ground, making it about 3 to 5 m away from the back, depending on the height of the tunnel in that particular area of the mine. It can be seen from the map that there is a little bit of drift in the data over time. Additionally, the map appears to twist on itself in some locations, which is caused by the accumulated error over time and the lack of loop closure.

Figure 4.14: Calculated trajectory of camera in the Hemlo1 dataset. The approximate trajectory of this dataset is 202 m, and 954 of the 2601 query images were localized.

In Figure 4.14 the trajectory of the camera calculated by the proposed technique is represented by the red line. A minimum inlier threshold of 8 was used to visualize the trajectory. Out of 2601 input images, 954 query images were localized. The calculated camera's trajectory appears to follow the shape of the 3D map, and appears to be a constant distance away from the back. Also, even though only 37% percent of the query images were localized, the localized images appear throughout different portions of the map, making it possible to re-create the entire trajectory.

Figure 4.15 shows the calculated trajectory against a reference trajectory for the same dataset. The trajectory of the Kinect, calculated by RGBDSLAM during map

building, is used as the reference and is represented by the green line. It can be seen from Figure 4.15 that the calculated trajectory follows the reference trajectory for the entire length of the 3D map, with only slight errors in the calculated position.

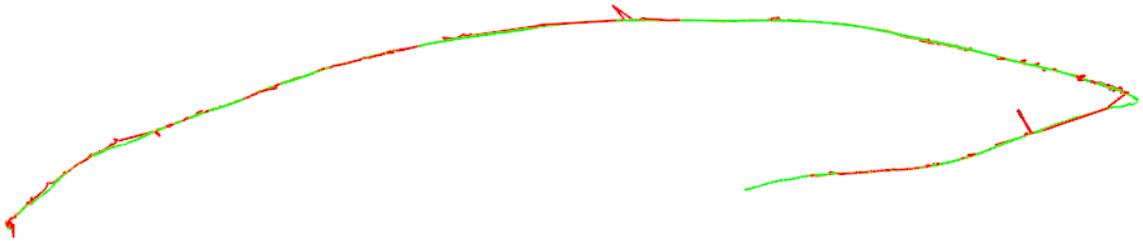


Figure 4.15: Calculated trajectory and reference trajectory in the Hemlo1 dataset. The red line represents the camera trajectory calculated using the proposed system, and the green line represents the trajectory of the Kinect, calculated by RGBDSLAM during map building.

The Hemlo2 dataset was also collected on the 9975 level of the mine, and is a map of the ramp located at the west end of the level, leading up to level 10030. The map building for this dataset began at the intersection of the main tunnel and the start of the ramp, and continued moving up the ramp until the next level was reached. The RGBDSLAM program crashed about three quarters of the way up the ramp because there were too many point clouds in the map. Therefore the final quarter of the ramp is missing from the map. The query images were captured by driving the trailer down the ramp at a speed of approximately 9 km/h.

The trajectory of the camera, calculated by the proposed system, can be seen in Figure 4.16. A total of 2251 query images were input into the system, and 411 of them were localized, which is about 18% of the query images. There are sections of the ramp where a series of query images did not contain enough inliers to be localized. In these sections the trajectory is a straight line which connects the current camera location to the last position that could be calculated. This section can be seen in

the right portion of the image where the map curves. Finally, Figure 4.17 shows the calculated trajectory against the reference trajectory.

Figure 4.16: Calculated trajectory of camera in the Hemlo2 dataset. The approximate trajectory of this dataset is 175 m, and 411 of the 2251 query images were localized.

Decreasing the minimum inlier threshold would allow more query images to be localized, but this would be at the risk of greatly reducing the accuracy. To see if it is worth decreasing the minimum inliers threshold, information such as the query image number, the matched database image number and the number of inliers found by PnP for each image was saved to a text file. Table 4.7 shows the distribution of query images over the number of inliers for both datasets. Analyzing this data would determine if the low number of localized query images is due to having the minimum inlier threshold set too high, or if it is because most of these images do not have good

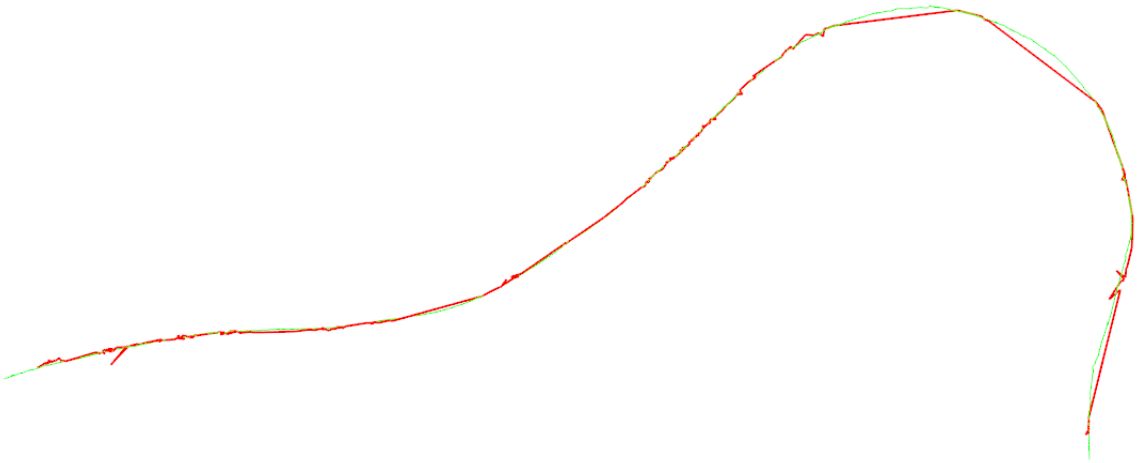


Figure 4.17: Calculated trajectory and reference trajectory in the Hemlo2 dataset. The red line represents the camera trajectory calculated using the proposed system, and the green line represents the trajectory of the Kinect, calculated by RGBDSLAM during map building.

inputs for PnP to determine the camera position.

Table 4.7: Number of inliers for both Hemlo datasets

Number of Inliers	Hemlo1		Hemlo2	
	Images Localized	Percentage	Images Localized	Percentage
4	840	32.3	968	43.1
5	92	3.5	90	4.0
6	39	1.5	7	0.3
7	22	0.8	11	0.5
$\geq$8	954	36.7	411	18.3

The Hemlo1 and Hemlo2 datasets had 652 query images (25.1%) and 758 query images (33.8%) respectively that were not localized because they did not contain any inliers. In addition, from Table 4.7, it can be seen that a large number of query images had only four inliers. While including these images would mean that more of them would be localized, most of the query images containing four inliers would have an error in their calculated position. In a sense the number of inliers returned by PnP can

be seen as a level of confidence towards the accuracy of the calculated position. Four is the minimum number of inliers possible, which means that four object points fit the position model. However, it is possible that the model could be wrong, as four points are not that many. On the other hand, if the number of inliers are higher, for example eight, it means that eight input points passed the reprojection error threshold and were projected back into 3D space very close to the location of the actual 3D point. If more points fulfil the model for the camera position, it means that the calculated position is more likely to be correct. While there are some query images containing five, six and seven inliers, it can be seen that this is a very small percentage of the overall query images. Even if the minimum inlier threshold was decreased to allow these images to be localized, it would not be worth it, as they would not provide significantly more information at the risk of allowing more erroneous positions to be localized.

4.5 Database Subset Experiments

One way to increase the number of localized query images and the speed of localization is to search only a subset of the feature database for a best matched image. The first query image input into the feature database would have to be matched against all the database features to find the best matched database image. However, it can be assumed that subsequent query images are located close to the current query image, therefore it is sufficient to search only features within the neighbourhood of the previous query image's best matched database image. Searching through a subset of the features in the feature database increases the speed of the system, and it also increases the accuracy of the match between the database and the query image, as it

limits the number of database images that could be returned. Therefore the likelihood of returning the correct database image is greater, which results in having more query images localized.

The current set-up of the database searches through all the database features for each query image. A "tracking" portion, where only a subset of the database images are matched to the query image has not been implemented. As the feature database is currently stored in a k-d tree structure and is ordered by features and not by database images, it would require a complete overhaul of the feature database structure to be able to implement a tracking component. This is a major change which is outside the scope of this work.

Instead a set of experiments, called *subset experiments*, were performed to see how such a change might improve localization. This experiment was conducted on the Hemlo datasets because they are significantly larger than all the other datasets presented, therefore the effect of the changes are more profound. For the purpose of these experiments, each subset contained 100 database images. This subset size was chosen arbitrarily, to be large enough to localize a decent number of query images, but to be small enough to see the effect of searching and localizing in a subset of the database. For each subset a new feature database was created, containing only the selected 100 database images.

The query images that corresponded to the part of the mine mapped by the subset databases were localized and the results obtained are presented in Tables 4.8 and 4.9. The first table shows the number of query images input for each subset and the number of images localized. It can be seen that for some of the subsets such as *Hemlo2 Subsets 1-4* the number of localized images increases significantly. For other

subsets the number of localized images does not increase too much, which means that the subset is not a good area for localization.

Table 4.8: Number of query images localized in subset of database vs. entire database.

Subset Name	Total Images	Subset		Entire database	
		Images Localized	% Localized	Images Localized	% Localized
Hemlo1 Subset1	192	123	64.1	115	59.9
Hemlo1 Subset2	231	70	30.3	25	10.8
Hemlo1 Subset3	241	120	49.8	94	39
Hemlo1 Subset4	196	144	73.5	115	58.7
Hemlo2 Subset1	206	158	76.7	43	20.9
Hemlo2 Subset2	186	73	39.2	7	3.7
Hemlo2 Subset3	246	156	63.4	72	29.3
Hemlo2 Subset4	216	97	44.9	35	16.2
Hemlo2 Subset5	301	28	9.3	19	6.3

The timing data for localizing the same query images within the subset database and the complete database is presented in Table 4.9. From this table it can be seen that localization is much faster in the subset than it is while searching through the features in the entire database. Therefore implementing a "tracking" portion of the feature database which only searches the features in a subset of the database images would not only improve the number of query images that are localized, but it would also greatly improve the performance of the system.

Table 4.9: Time for localizing query images.

Subset Name	Total Images	Subset		Entire database	
		Runtime (s)	# DB Features	Runtime (s)	# DB Features
Hemlo1 Subset1	192	360.32	19729	1609.9	147741
Hemlo1 Subset2	231	388.61	20143	2422.54	147741
Hemlo1 Subset3	241	482.78	23404	2505.58	147741
Hemlo1 Subset4	196	333.9	19168	1823.08	147741
Hemlo2 Subset1	206	540.04	43515	3465.69	216150
Hemlo2 Subset2	186	447	33977	3131.59	216150
Hemlo2 Subset3	246	536.72	36612	3832.41	216150
Hemlo2 Subset4	216	408.26	21310	3466.29	216150
Hemlo2 Subset5	301	452.49	18286	4600.45	216150

4.6 System Runtime

All the results presented in this chapter were run on one of two computers. All datasets except for the Hemlo ones were tested on an Intel Core i7 machine with 8 GB of RAM. The Hemlo datasets were run on an Intel Core i7 machine with 16 GB of RAM. Neither of the Hemlo datasets could be run on the first machine because they contained too many features in the database, exceeding the capacity of the RAM.

The implementation of the proposed localization system is a lab prototype code written in C++ using OpenCV [2] and Point Cloud Library (PCL) [3]. Care was put into designing the code to be as optimal as possible, but because it is research code, some processes can be redesigned to be more optimal. Due to the current structure of the feature database, some processes such as feature matching between the query and database image are performed twice, once in the feature database and once in the main code after the best matched database image is returned. Another unoptimized area of the code is when the 3D and 2D image features of the best matched database image are read from the file during each iteration. To speed up the process, the 3D

and 2D image features from the previous iteration are stored in the system and if the current query image is matched to the same database image as the previous query image, then the stored values are used instead of reading from file. However, if the matched database image is different for the current query image then this information needs to be read from file. Therefore the code has not been optimized for runtime and no attention was paid to performing optimization at the compiler level, leaving lots of opportunities to improve the performance of the proposed system at runtime.

Localization using the proposed technique was very efficient for the smaller datasets but became significantly slower as the number of features in the database increased. For every query image, the entire database was searched to find the closest matched database image, which was not efficient for large datasets. Figure 4.18 shows the average time for running the proposed system per frame as a function of the number of features in the database. The average time per frame was found by dividing the total runtime of the dataset by the number of query images input into the proposed system. From the figure it can be seen that the system runtime linearly increases as a function of the number of features present in the database.

The proportion of runtime the different processes of the proposed system require is presented in Figure 4.19. For each dataset the amount of time each process took is displayed along with the total runtime. Each process was divided by the total runtime to obtain a percentage. The percentage of each process for all datasets presented in this chapter were averaged to get the final numbers used in Figure 4.19.

By far the slowest process is reading the database information from file. With the current implementation, the feature database only returns the index of the best matched database image. The rest of the information about the database image is

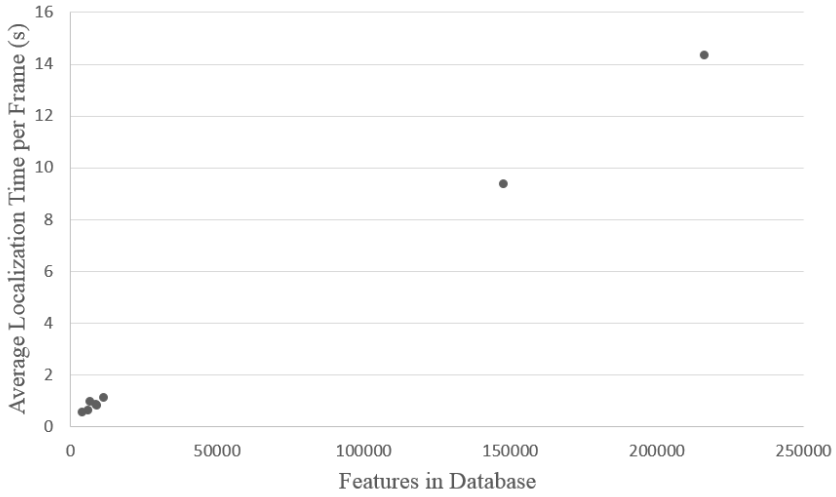


Figure 4.18: Localization runtime as a function of the number of features in the database.

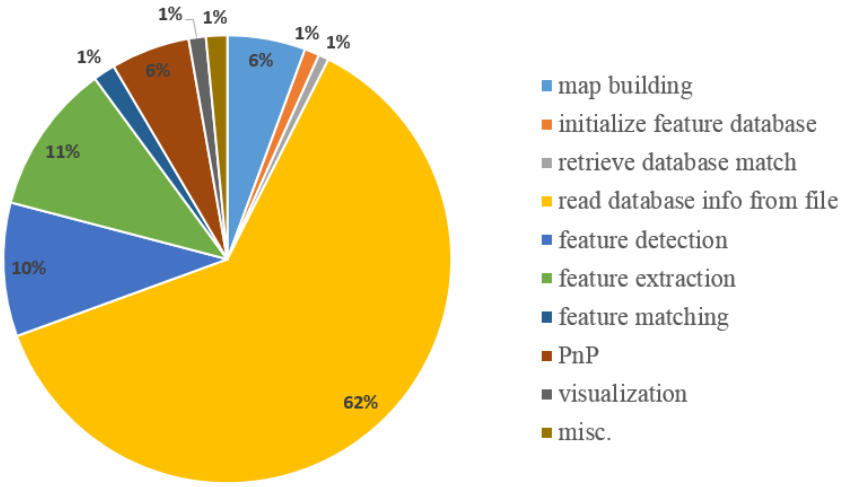


Figure 4.19: Breakdown of the percentage of runtime the different processes take to localize the query images in the current implementation. If the code is optimized, the amount of time spent on some processes would be reduced. For example, the time spent reading the database information from file could be reduced by restructuring the feature database to store this information. Similarly, the time spent on keypoint detection and extraction could be reduced by using a more efficient keypoint detector, and by using the OpenCV GPU module to speed up this process.

stored in text files, with one file containing all the feature descriptors, another file containing all the 2D feature locations and a third file containing the list of 3D feature locations. For each iteration, once the best matched database image is returned, these files are read to get the relevant information. The current database structure does not allow for the 2D and 3D feature locations to be stored, therefore they must be read from file. Future implementations of the feature database should include this information in the database structure. This would eliminate the slow process of reading from file and should speed up the entire system.

The second and third slowest processes are feature detection and extraction, which must be performed for each query image. This part of the process is required and therefore cannot be eliminated. Another option for future work is to explore different types of feature detectors and extractors and to find one which is more efficient while maintaining the accuracy of the current system. One possibility is to use ORB [62] which has comparable accuracy but is faster than SIFT and SURF. PnP only takes 6% of the total runtime, therefore it is an efficient way to determine the position of the camera. Information about the total runtime for all the datasets is presented in Table 4.10.

Table 4.10: Timing results.

Dataset	# DB Features	Total Runtime (s)	Runtime per Image (s)
Lab	6437	31.87	0.966
Office	3886	28.97	0.579
Office712	8766	141.65	0.836
JDUC	5770	189.25	0.651
WLH	8608	253.09	0.855
NORCAT	11178	525.03	1.136
Hemlo1	147741	24385	9.375
Hemlo2	216150	32356.40	14.374

4.7 Discussion about Uncertainty

There are a couple of sources of uncertainty in the proposed localization method. PnP relies on the 2D-3D feature correspondences for calculating the position of the camera. If the 3D positions of the features are incorrect, the solution obtained by PnP would be incorrect.

When using a database image as a query image, the database returns the correct match, as the same image is already present in the database. However, when using a query image that is not in the database, and therefore a bit different, there is room for error in the returned database image. Most of the time the returned database image is correct, however there are times when an incorrect image is returned. One example of this, which was discussed in the localization experiments above, was for the JDUC datasets when the query image of the recycling bins in the second location returned the database image of the recycling bins in the first location. When a completely incorrect image is returned from the database, any feature matches between the query image and the database image are incorrect, as the 3D points obtained from the database do not represent the location of the 2D features appearing in the query image. Therefore any position calculated by PnP is incorrect in this case.

Another source of error is caused during the feature matching process, when the query image features are matched to the returned database image features. During the feature matching process, some of the matches are incorrect. Typically, RANSAC should be able to determine the inliers and eliminate outliers. However, in certain cases there may not be enough features, or the number of wrong matches may be too much, and RANSAC can not determine the inliers correctly. In this case, the camera position calculated by PnP contains some error because some of the object points

used to determine the camera position are outliers. In addition, having only a small number of feature matches input into PnP increases the uncertainty in the camera position.

Since the proposed localization technique finds the position of the camera within a precomputed 3D map of the environment, some uncertainty also exists within the mapping process. However, the purpose of the work in this thesis is to localize accurately within a map. Therefore the assumption used is that an accurate 3D map of the environment is available. From the results presented in this section, it can be seen that the 3D maps used have some error. For example, in the JDUC dataset, loop closure was not performed, and the back right corner of the map drifts and is superimposed on itself. Also, in the Hemlo Mine datasets, the map twists over time, which happens because loop closure could not be performed in this environment, resulting in drift over time. The focus of this work was to obtain good localization results within the available 3D map, and not on improving the mapping technique. If a more accurate map is available, the accuracy of the localization would also improve.

Lastly, the limitation of the sensors also have an impact on the mapping and localization results. The error in the Kinect sensor when capturing the 3D point clouds could reduce the accuracy of the overall 3D map. Additionally, this error could also reduce the accuracy of the localization process if the Kinect does not correctly calculate the 3D feature locations. During localization, it is assumed that the 3D location of the features are accurate. Since the 3D features are an important part of calculating the camera's position, any error in the 3D feature location would reduce the accuracy of the localization. In the case of the camera used for capturing the query images, the camera calibration process is very important. Since PnP uses

the camera's calibration parameters to calculate the camera's position, if the camera is not properly calibrated there would be error in the localization results.

Chapter 5

Conclusions and Future Work

5.1 Summary of Conclusions

A visual localization system for underground mines and indoor environments was presented in this thesis. The lack of absolute positioning systems, such as GPS, makes localization in these environments challenging. The proposed system performs localization in a previously mapped environment using only a calibrated monocular camera and a feature database containing the 2D and 3D locations of features existing in the 3D map. The proposed system does not require a large set-up of sensor suites, or the installation of markers in the environment, and uses commercially available sensors for both map building and localization. An RGB-D sensor, the Kinect, is used to build the 3D map of the environment, and a calibrated monocular camera is used for capturing images for localization. Being able to use different cameras during localization means that the user is not constrained to a particular brand or model of camera, and the camera can easily be replaced if necessary. It provides the user the option to use a camera they are more familiar with or may already own. This is advantageous because the idea behind this technique is to perform localization

significantly more often than map building. Therefore it is important to be able to use inexpensive, commodity cameras during localization.

Two sets of experiments were performed to test the performance of the proposed system. The ground truth experiments showed that the localization results obtained by the proposed technique were close to the reference camera position obtained using the open source RGBDSLAM system. Increasing the number of inliers in the image decreases the translation error of the calculated camera position. The localization experiments showed that the proposed system can be used in different environments, such as small and large rooms, and even more extreme environments such as underground mines. This experiment was performed on three indoor datasets collected around the Queen's University campus, and in three underground mine datasets.

5.2 Future Work

This section presents a number of possible improvements and tests that can be performed on the work described in this thesis. The ultimate goal for localization is to be real-time and as accurate as possible. Therefore the recommended future work focuses on bringing the proposed localization system closer to this goal.

5.2.1 Feature Database

The current database structure only stores the feature descriptors and the index of the database images. It would be beneficial for the database to also store the 2D and 3D locations of the features. For the current implementation of the localization system, it was seen that reading the 2D and 3D feature locations from file for each iteration was by far the longest runtime process. If the feature locations could be

read and stored into the database during the initialization process, then it would significantly speed up the localization. This is because the feature locations would already be available in the feature database structure and their values would simply be passed to the localization process along with the best matched database image. Doing this would also put the time consuming portion of reading from file into the initialization of the program, reducing the time required to localize each image, and bringing it closer to the desired real-time localization.

The feature database should also be modified to handle different types of feature descriptors. Currently only SIFT and SURF features may be used in the database. By expanding the feature database to include other descriptors such as ORB and FAST, it would be possible to see how the different feature descriptors affect the accuracy and runtime of the localization process. Then the feature descriptor that has the best balance of accuracy and runtime can be used for localization.

Another avenue of future work on the feature database is to modify it to contain an *initialization* and a *tracking* component. When the first query image is passed into the feature database, it could be taken from anywhere in the map, therefore it is necessary to search through all the database images to find which is the closest to the query image. The process of searching through the entire database would be called *initialization*. Once the first query image has been localized, it is safe to assume that the subsequent query image is captured in an area of the map close to the first query image. This assumption is valid because during localization, the camera is moved with the user, therefore it is impossible for subsequent query images to jump from one part of the map to a completely different part. By knowing that the second query image is located near the first query image, there is no point in searching through

the entire feature database for the best matched image. Instead it is enough to search through only a small window of database images surrounding the previously matched database image, to find the one closest to the query image. This process of searching through only a subset of the database images could be called the *tracking* component. The size of the search window used for tracking would depend on the speed of the query camera, with faster camera movements requiring a larger window size. Additionally, a counter needs to be used during the tracking component. This counter would keep track of the number of consecutive query images that could not be localized. If the number exceeds a threshold, that means localization has been lost, and the database would revert to the *initialization* process to position the camera. As was seen in the results presented in Chapter 4, implementing a tracking component into the feature database is beneficial because it increases the accuracy and reduces the database search time, which in turn reduces the time required to localize each query image.

5.2.2 Localization

Future research in the localization process of the proposed system should focus on increasing the accuracy. One main area is in improving the feature matches between the query image and the database image. The assumption is that these matches are correct, and these feature matches are used to create the list of 2D and 3D feature correspondences that are passed into PnP to solve for the camera position. However in practice, incorrect feature matches are always present, with some images having more incorrect matches than others. Therefore it is beneficial to experiment with different feature detectors and feature matchers to see how the number of incorrect

matches can be reduced. Reducing the number of incorrect feature matches being passed into PnP would result in a more likelihood of solving for the correct camera position.

Another problem affecting the accuracy of localization is when the features used to calculate the position are clustered in one small area of the image, or are coplanar, as discussed in Chapter 4. One solution for this is to divide the image into quadrants, and ensure that a minimum number of features from each quadrant is selected for calculating the camera's position. This technique is used in some visual localization systems [42]. Implementing this system can ensure that features are dispersed throughout the image and may eliminate some of the incorrect camera positions.

5.2.3 Overall System

Other areas of future work include making the proposed system more general. A Microsoft Kinect version 1 was used for building the map of all the datasets shown in the results section. Research should be done to explore what other types of 3D mapping techniques and sensors can be used to create the map of the environment, such as the Kinect version 2. These maps should then be tested with the proposed system to determine if a particular sensor or mapping system provides better results for localization. Another area of future research is to explore how robust the system is to changing environments. After changes have been made to improve the speed of localization, the system should be used to perform localization in real-time. Additionally, future research could focus on developing a mechanism to make it possible to collect ground truth data in every environment where the query images are collected.

Another avenue of future work is to explore the possibility of including additional

sensors to the localization system, such as IMUs or wheel encoders. The data from this sensor could be used to check that the calculated camera position is correct. This would be particularly useful when the calculated position does not contain many inliers, making it more prone to localization errors. In this case the data from the other sensors can be used to determine if the calculated position is physically possible. Additionally, the data from the other sensors could be used to calculate the position of the vehicle in areas of the environment that do not contain enough features for PnP to calculate the position.

5.3 Recommendation for use in Underground Mines

The work presented in this thesis is proof of concept that localization can be performed in underground mines using only cameras. The modifications mentioned in the future work section should be studied and implemented to make the system real-time, and further experiments are required in underground mines. However, this is a promising technique and further research should be done on it to test its robustness and improve the localization speed. This section discusses three possible future research areas needed to make the proposed system work better in underground mines. The three areas are: creating better 3D maps of the mine, making the lighting system smaller and portable, and conducting further testing in underground mines.

The purpose of the proposed system is to allow the user to find out where they are within the underground mine. This is done by showing the user their position within a 3D map of the environment. The user then needs to apply that visual information to their present 3D surroundings to navigate. We have seen that the back of the tunnel is a good place for finding visual features and for calculating the user's position.

However, it is also easy to see from the mine datasets presented in Chapter 4 that a 3D map of a small portion of the back of the tunnel does not tell the user much about where exactly they are within the mine. Even though it is possible for the computer to determine where the camera is within the available 3D map, if this information is not presented in a way that the user can apply it to their current surroundings, it does not help them with finding their position within the mine. Therefore further research is required into creating better 3D maps of underground mines, which make it helpful for the user to find their location within the underground mines.

There are three options for future research into creating better 3D maps for underground mines. The first option is to continue using RGB-D sensors for map building, but to include the side walls of the tunnel in the map as well. By including these walls, the 3D map preserves the shape of the tunnel, making the map more realistic. In addition, experimenting with better RGB-D sensors, such as the Kinect version 2 might be beneficial, as it may result in a better map. The second option is to use existing mapping techniques used in mines to generate the 3D map. One technique is to use a laser scanner to get a dense 3D point cloud of the map. Laser scanners do not provide intensity information about the environment, therefore this option requires collecting the images separately and registering them with the 3D point clouds, to get the list of 2D and 3D feature correspondences. The third possibility is to continue collecting the 3D map of the back using the Kinect, and to overlay this map with a physical map of the mine. Paper maps already exist of each level of the mine, therefore the task would be to figure out how to overlay the 3D map of the back of the tunnel with the paper map. During localization, the user's position would then be shown on the map, which helps them get a better idea of where they are.

As mentioned previously, the computer's RAM presents a limiting factor for the size of the 3D map that can be used for the feature database. By creating a more efficient structure for the feature database, the hope is that it would allow more features to be added, allowing for a larger localization environment. Even with the modifications to the feature database, it is likely that the underground mines are too large to store into one database. In this case, it may be possible to have multiple feature databases, one for each level of the mine, and one for the ramps. When changing levels, the system would then simply load the database corresponding to the current level. These are just some suggestions for future work regarding how to solve the limitations of the proposed system.

Further research needs to be done on minimizing the hardware required for localization in underground mines in order to make the system more portable. Currently, a large part of the hardware is a network of lights which are powered by a car battery. The ideal lighting system should be small but bright and diffuse, with portable batteries. Another possibility is to research different types of cameras and experiment with ones that work better in dimly lit environments, such as high dynamic range sensors.

Lastly, it would be beneficial to further test the localization system in underground mines. The proposed future work for the current system, particularly with the feature database, should improve the localization speed and bring the system closer to running at real-time. Once the changes have been implemented, localization should be performed in real-time in underground mines to test the robustness and performance of the system. This thesis has shown that it is possible to perform localization in underground mines using a calibrated camera and a 3D map of the environment, and

the results presented have been promising.

Bibliography

[1] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE Int. Conf.*, pp. 3607–3613, May 2011.

[2] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 120–126, 2000.

[3] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[4] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, June 1981.

[5] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *Robotics, IEEE Transactions*, vol. 30, pp. 177–187, Feb 2014.

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

[7] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning - sensors and techniques," *Journal of Robotic Systems*, vol. 14, no. 4, pp. 231–249, 1997.

[8] A. Noureldin, T. B. Karamat, and J. Georgy, *Fundamentals of Intertial Navigation, Satellite-based Positioning and their Integration*. Springer.

[9] K. Yelamarthi, "An autonomous passive RFID-assisted mobile robot system for indoor positioning," *International Journal of Modern Engineering*, vol. 14, no. 2, pp. 28–37, 2014.

[10] N. Hughes, J. Pinchin, M. Brown, and D. Shaw, "Navigating in large hospitals," in *IPIN 2015 Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN 2015)*, (Banff, Canada), Oct. 2015.

[11] N. J. Lavigne and J. A. Marshall, "A landmark-bounded method for large-scale underground mine mapping," *Journal of Field Robotics*, vol. 29, no. 6, pp. 861–879, 2012.

[12] J. Biswas and M. Veloso, "WiFi localization and navigation for autonomous indoor mobile robots," in *Robotics and Automation (ICRA), 2010 IEEE Int. Conf.*, pp. 4379–4384, IEEE, May 2010.

[13] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios," *IEEE Signal Processing Magazine*, vol. 35, pp. 70–84, July 2005.

[14] D. Zhang, F. Xia, Z. Yang, L. Yao, and W. Zhao, "Localization technologies for indoor human tracking," in *Future Information Technology (FutureTech), 2010 5th Int. Conf.*, pp. 1–6, IEEE, 2010.

[15] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, pp. 155–166, February 2009.

[16] D. Sinha, M. T. Ahmed, and M. Greenspan, "Image retrieval using landmark indexing for indoor navigation," in *Computer and Robot Vision (CRV), 2014 Canadian Conf.*, pp. 63–70, May 2014.

[17] H. Alismail, B. Browning, and M. B. Dias, "Evaluating pose estimation methods for stereo visual odometry on robots," in *Proc. 11th Int. Conf. Intelligent Autonomous Systems*, August 2010.

[18] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau, "Localization in urban environments: monocular vision compared to a differential GPS sensor," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conf.*, vol. 2, pp. 114–121 vol. 2, June 2005.

[19] D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *International Journal of Computer Vision*, vol. 15, no. 1-2, pp. 123–141, 1995.

[20] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle, "An analytic solution for the perspective 4-point problem," in *Computer Vision and Pattern Recognition, 1989. Proc. CVPR '89., IEEE Computer Society Conf.*, pp. 500–507, Jun 1989.

[21] X.-S. Gao and J. Tang, "On the probability of the number of solutions for the P4P problem," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 1, pp. 79–86, 2006.

[22] C.-X. Zhang and Z.-Y. Hu, "A general sufficient condition of four positive solutions of the P3P problem," *Journal of Computer Science and Technology*, vol. 20, no. 6, pp. 836–842, 2005.

[23] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 25, pp. 930–943, August 2003.

[24] S. Zhang, X. Cao, F. Zhang, and L. He, "Monocular vision-based iterative pose estimation algorithm from corresponding feature points," *Science China Information Sciences*, vol. 53, no. 8, pp. 1682–1696, 2010.

[25] D. Leng and W. Sun, "Finding all the solutions of PnP problem," in *Imaging Systems and Techniques, 2009. IST '09. IEEE Int. Workshop*, pp. 348–352, May 2009.

[26] L. Quan and Z. Lan, "Linear N-point camera pose determination," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 21, pp. 774–780, Aug 1999.

[27] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, pp. 629–642, Apr 1987.

[28] C.-P. Lu, G. D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 22, pp. 610–622, Jun 2000.

[29] S. Li and C. Xu, "A stable direct solution of perspective-three-point problem," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 5, pp. 627–642, 2011.

[30] S. Li, C. Xu, and M. Xie, "A robust O(n) solution to the perspective-n-point problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 34, pp. 1444–1450, July 2012.

[31] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conf.*, pp. 2969–2976, June 2011.

[32] G. Schweighofer and A. Pinz, "Globally optimal O(n) solution to the pnp problem for general camera models," in *Proc. British Machine Vision Conf. 2008*, pp. 1–10, Sept 2008.

[33] J. A. Hesch and S. I. Roumeliotis, "A direct least-squares (DLS) method for PnP," in *Computer Vision (ICCV), 2011 IEEE Int. Conf.*, pp. 383–390, Nov 2011.

[34] Y. Guo, "A note on the number of solutions of the coplanar P4P problem," in *Control Automation Robotics Vision (ICARCV), 2012 12th Int. Conf.*, pp. 1413–1418, Dec 2012.

[35] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi, "Revisiting the PnP problem: A fast, general and optimal solution," in *Computer Vision (ICCV), 2013 IEEE Int. Conf.*, pp. 2344–2351, Dec 2013.

[36] R. M. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose estimation from corresponding point data," *Systems, Man and Cybernetics, IEEE Transactions*, vol. 19, pp. 1426–1446, Nov 1989.

[37] D. Xu, Y. Li, and M. Tan, "Visual positioning using four-point planar patterns," in *Automation Science and Engineering, 2006. CASE '06. IEEE Int. Conf.*, pp. 600–605, Oct 2006.

[38] T. Wang, Y. Wang, and C. Yao, "Some discussion on the conditions of the unique solution of P3P problem," in *Mechatronics and Automation, Proc. 2006 IEEE Int. Conf.*, pp. 205–210, June 2006.

[39] F.-M. Sun and B. Wang, "The solution distribution analysis of the P3P problem," in *Systems Man and Cybernetics (SMC), 2010 IEEE Int. Conf.*, pp. 2033–2036, Oct 2010.

[40] M. Q. Rieck, "An algorithm for finding repeated solutions to the general perspective three-point pose problem," *Journal of Mathematical Imaging and Vision*, vol. 42, no. 1, pp. 92–100, 2012.

[41] M. Q. Rieck, "A fundamentally new view of the perspective three-point pose problem," *Journal of Mathematical Imaging and Vision*, vol. 48, pp. 499–516, February 2014.

[42] Y. Wang, Q. Zhang, and Y. Zhou, "RGB-D mapping for indoor environment," in *Industrial Electronics and Applications (ICIEA), 2014 IEEE 9th Conf.*, pp. 1888–1892, June 2014.

[43] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Computer Vision ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of *Lecture Notes in Computer Science*, pp. 404–417, Springer Berlin Heidelberg, 2006.

[44] S. A. Scherer, D. Dube, and A. Zell, "Using depth in visual simultaneous localisation and mapping," in *Robotics and Automation (ICRA), 2012 IEEE Int. Conf.*, pp. 5216–5221, May 2012.

[45] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *Robotics & Automation Magazine, IEEE*, vol. 13, pp. 99–110, June 2006.

[46] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 32, no. 1, pp. 105–119, 2010.

[47] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *International Symposium on Robotics Research (ISRR)*, pp. 1–16, 2011.

[48] Y. Zou, W. Chen, X. Wu, and Z. Liu, "Indoor localization and 3D scene reconstruction for mobile robots using the Microsoft Kinect sensor," in *Industrial Informatics (INDIN), 2012 10th IEEE Int. Conf.*, pp. 1182–1187, July 2012.

[49] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Vision algorithms: theory and practice*, vol. 1883, pp. 298–372, Springer-Verlag, 2000.

[50] B. Clipp, J. Lim, J.-M. Frahm, and M. Pollefeys, "Parallel, real-time visual SLAM," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ Int. Conf.*, pp. 3961–3968, Oct 2010.

[51] Y. Sun, N. Ding, H. Qian, and Y. Xu, "Real-time monocular visual self-localization approach using natural circular landmarks for indoor navigation," in *Robotics and Biomimetics (ROBIO), 2012 IEEE Int. Conf.*, pp. 495–500, Dec 2012.

[52] S. Xu and M. Liu, "Feature selection and pose estimation from known planar objects using monocular vision," in *Robotics and Biomimetics (ROBIO), 2013 IEEE Int. Conf.*, pp. 922–927, Dec 2013.

[53] K. Choi, S. Lee, and Y. Seo, "A branch-and-bound algorithm for globally optimal camera pose and focal length," *Image and Vision Computing*, vol. 28, no. 9, pp. 1369–1376, 2010.

[54] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ Int. Conf.*, pp. 2531–2538, Sept 2008.

[55] A. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3D visual odometry," in *Robotics and Automation, 2007 IEEE Int. Conf.*, pp. 40–45, April 2007.

[56] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, 2007.

[57] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 172 – 187, 2009.

[58] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[59] L. E. Clement, J. Kelly, and T. D. Barfoot, "Monocular visual teach and repeat aided by local ground planarity," in *Proc. Field and Service Robotics (FSR)*, June 2015.

[60] P. Lin, Q. Li, Q. Fan, X. Gao, and S. Hu, "A real-time location-based services system using WiFi fingerprinting algorithm for safety risk assessment of workers in tunnels," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[61] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[62] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE Int. Conf.*, pp. 2564–2571, Nov 2011.

[63] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP (1)*, vol. 2, 2009.

[64] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, November 2000.

[65] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, ch. 11, pp. 401–403. O'Reilly, first ed., September 2008.

[66] M. Von Steinkirch, "Introduction to the Microsoft Kinect for computational photography and vision." May 2013.

[67] E. Deretey, M. T. Ahmed, J. Marshall, and M. Greenspan, "Visual indoor positioning with a single camera using PnP," in *IPIN 2015 Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN 2015)*, (Banff, Canada), Oct. 2015.

[68] Point Grey, "Blackfly PGE PoE digital camera technical reference," Tech. Rep. 12.0, Point Grey, 12051 Riverside Way, Richmond, BC, Canada, October 2015.

Appendix A

P3P Equations

What follows are the equations used to derive the coefficients G4, G3, G2, G1, and G0, used in the P3P solution presented by Fischler and Bolles [4]. These equations were taken from the appendix of their paper, and are shown here to provide the reader with more information about what these coefficients represent.

$$G4 = (K1 * K2 - K1 - K2)^2 - 4 * K1 * K2 * [\cos(\theta_{bc})^2] \tag{A.1}$$

$$\begin{aligned} G3 = {} & 4 * [K1 * K2 - K1 - K2] * K2 * (1 - K1) * \cos(\theta_{ab}) \\ & + 4 * K1 * \cos(\theta_{bc}) * [(K1 * K2 + K2 - K1) * \cos(\theta_{ac}) \\ & + 2 * K2 * \cos(\theta_{ab}) * \cos(\theta_{bc})] \end{aligned} \tag{A.2}$$

$$\begin{aligned} G2 = {} & [2 * K2 * (1 - K1)\cos(\theta_{ab})]^2 + 2 * [K1 * K2 + K1 - K2] \\ & * [K1 * K2 - K1 - K2] + 4 * K1 * [(K1 - K2) * (\cos(\theta_{bc})^2) \\ & + (1 - K2) * K1 * (\cos(\theta_{ac})^2) - 2 * K2 * (1 + K1) * \cos(\theta_{ab}) * \\ & \cos(\theta_{ac}) * \cos(\theta_{bc})] \end{aligned} \tag{A.3}$$

$$\begin{aligned} \mathrm{G1} = {} & 4 * (K1 * K2 + K1 - K2) * K2 * (1 - K1) * \cos(\theta_{ab}) \\ & + 4 * K1 * [(K1 * K2 - K1 + K2) * \cos(\theta_{ac}) * \cos(\theta_{bc}) \\ & + 2 * K1 * K2 * \cos(\theta_{ab}) * \cos(\theta_{ac})^2] \end{aligned} \tag{A.4}$$

$$\mathrm{G0} = (K1 * K2 + K1 - K2)^2 - 4 * (K1^2) * K2 * (\cos(\theta_{ac})^2) \tag{A.5}$$

where $K1 = \frac{R_{bc}^2}{R_{ac}^2}$ and $K2 = \frac{R_{bc}^2}{R_{ab}^2}$.

Appendix B

Images from Datasets

This appendix presents images from the different datasets to provide an idea of the different environments the localization system was tested in.

Figure B.1: Sample of images from the Office dataset.

Figure B.2: Sample of images from the Lab dataset.

Figure B.3: Sample of images from the Office712 dataset.

Figure B.4: Sample of images from the JDUC dataset.

Figure B.5: Sample of images from the WLH dataset.

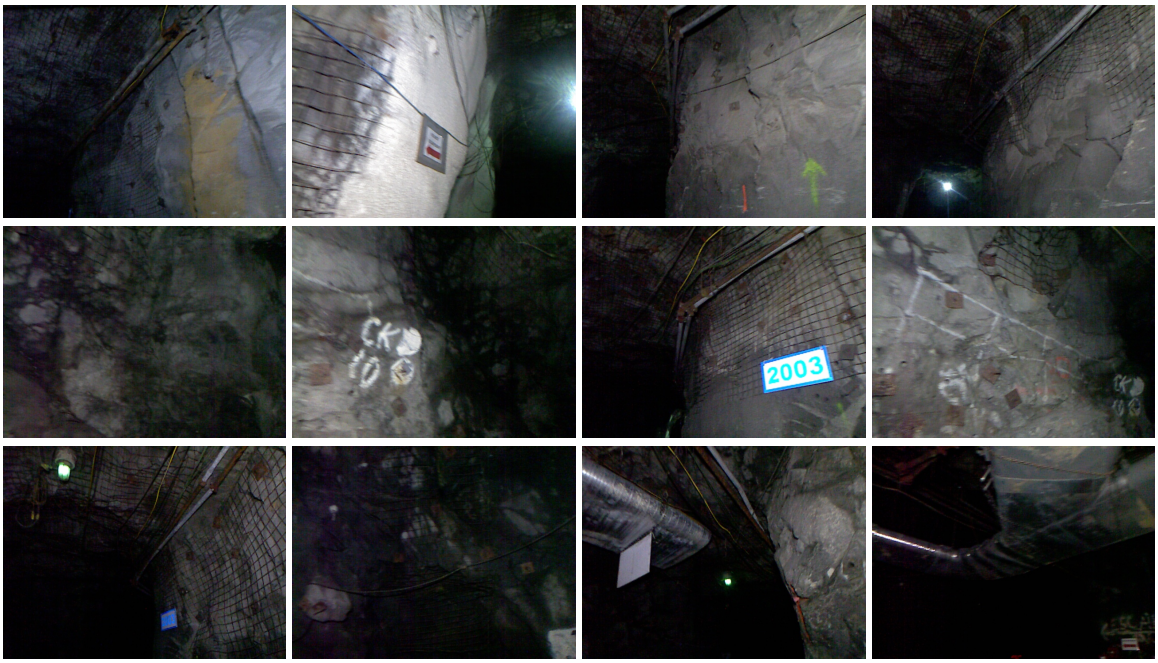


Figure B.6: Sample of images from the NORCAT dataset.