**Project Title:**

Robot Agnostic Fleet Manager Optimizer - Fleet Management Connectivity and Map Merging

**Submitted by:**

Lau Ler Young Ryan

**Department of Mechanical Engineering**

*In partial fulfilment of the requirements for the Degree of Bachelor of Engineering National University of Singapore*

**Session 2021/2022**

**Table of Content**

[Summary 4](#_Toc99014385)

[Acknowledgment 7](#_Toc99014386)

[List of Figures 8](#_Toc99014387)

[List of Symbols 12](#_Toc99014388)

[1. Introduction 13](#_Toc99014389)

[1.1 Background Information 13](#_Toc99014390)

[2. Literature Review – Map Merging 16](#_Toc99014391)

[2.1 Occupancy Grid Maps 16](#_Toc99014392)

[2.2 Occupancy Grid Map Merging Methods 18](#_Toc99014393)

[2.2.1 Probability Method 18](#_Toc99014394)

[2.2.2 Optimisation Method 20](#_Toc99014395)

[2.2.3 Feature-Based Method 22](#_Toc99014396)

[2.2.3.1 Feature Detector and Descriptor Algorithms 25](#_Toc99014397)

[2.3 Consolidation 26](#_Toc99014398)

[3. Literature Review – Communication Bridge 27](#_Toc99014399)

[4. Project Objectives 31](#_Toc99014400)

[4.1 Map Merging Tool 31](#_Toc99014401)

[4.2 Communication Bridge 32](#_Toc99014402)

[5. Project Scope 32](#_Toc99014403)

[5.1 Map Merging Tool 32](#_Toc99014404)

[5.2 Communication Bridge 33](#_Toc99014405)

[6. Methodology – Map Merging 33](#_Toc99014406)

[6.1 Pair-Map Stitcher Phase 33](#_Toc99014407)

[6.2 Feature-Detector-Descriptor Performance Evaluation Phase 42](#_Toc99014408)

[6.2.1 Dataset Acquisition 42](#_Toc99014409)

[6.2.2 Test Execution 44](#_Toc99014410)

[6.2.1 Test Observations 46](#_Toc99014411)

[6.2.1.1 Feature-Detection-Description Time 46](#_Toc99014412)

[6.2.1.2 Number of Inlier Matches against Applied Transformation 47](#_Toc99014413)

[6.2.1.3 Success Rate of Merging 48](#_Toc99014414)

[6.2.6 Test Conclusions 49](#_Toc99014415)

[6.3 Multi-Map Stitcher Phase 49](#_Toc99014416)

[6.4 Product Testing 55](#_Toc99014417)

[7. Methodology – Communication Bridge 57](#_Toc99014418)

[7.1 Architecture Design Proposal 57](#_Toc99014419)

[7.2 Product Testing 58](#_Toc99014420)

[8. Conclusions 60](#_Toc99014421)

[8.1 Map Merging 60](#_Toc99014422)

[8.2 Communication Bridge 61](#_Toc99014423)

[9. Recommendations for Further Work 62](#_Toc99014424)

[9.1 Map Merging 62](#_Toc99014425)

[9.2 Communication Bridge 63](#_Toc99014426)

[10. References 64](#_Toc99014427)

[Appendices 71](#_Toc99014428)

[Appendix A 71](#_Toc99014429)

[Appendix B 73](#_Toc99014430)

[Appendix C 75](#_Toc99014431)

[Appendix D 80](#_Toc99014432)

[Appendix E 81](#_Toc99014433)

[Appendix F 83](#_Toc99014434)

# **Summary**

Due to the advancement of technology, robots have become an affordable and capable alternative to solve the growing demand for a qualified workforce in the warehouse industry that was also hampered by Covid-19. With this warehouses employ a multitude of robotic solutions to assist with daily warehouse operations. For example, having Autonomous Mobile Robots (AMRs) to transport goods around the warehouse space, and having picking robots to place the goods on the AMR. The issue with the adopting of an extensive robotic solution is that interoperability is limited as different robot vendors utilise different communication protocols to operate their robotic systems. This makes it challenging for warehouse owners to adopt and control a “multi-diverse robotic system”.

This final year project (FYP) is provided by a warehouse robotics middleware software startup, Syncware Pte Ltd, that aims to build a connectivity platform for different robots to connect to regardless of the communication protocol utilised. In aspect, Syncware aims to create a robot fleet manager that is able to operate robots from different vendors. In addition, the platform will host a variety of utilities that can potentially add value to operations in a warehouse.

This project focuses on the creation of two deliverables that will be incorporated in the connectivity platform.

The first is a communication bridge that establishes connection and ensures constant exchange of information between external systems (i.e., robots) and the platform regardless of the communication protocol utilised. It was created from extensive tests with external collaborators that engaged with Syncware for its communication capabilities. One such collaborator is a local robotics solution company which needed Syncware to allow its own robot fleet to communicate with an external device. The architecture of the communication bridge was made of two components, the core and the plugin, whereby the core handles the communication with the external system while the plugin handles the communication with the platform. The main purpose of the proposed architecture was that for different communication protocols, the core is the only component that needs to be modified. This ensures that the communication bridge is robust and adaptable. However, further work is recommended to accommodate a greater variety of communication protocols.

The second is a map merging tool that can reduce the time spent in mapping out the enclosed area within a warehouse environment. This project sees the map merging problem as an image registration issue, and proposes a feature-based method to carry out the merging of maps. Through a modification of the Stitcher code pipeline, that was originally intended to generate a panorama from a set of overlapping photographs, the tool was capable of merging maps regardless of the order of input and the number of inputs. Furthermore, eight feature-detector-descriptors (FDDs) were evaluated for their performance in the merging process. This also allowed for the comparison of performance of handcrafted FDDs and learned FDDs, and it was found that handcrafted variants were more suitable due to the ability to find a greater number of matches between map images. Further work is recommended to evaluate the effectiveness of this tool as all tests were done without physical robots.

# **Acknowledgment**

The author wishes to express sincere appreciation to the following parties involved with this Final Year Project (FYP):

* Internal supervisors Prof. Marcelo H. Ang Jr. and Assoc. Prof. Chen Chao Yu, Peter, for their guidance and assistance.
* External supervisor Walter Frank Pintor Ortiz and Dr. Dirk-Jan van Veen for the proposal of the project and technical guidance.
* Dr. Patrick Gallo for providing extensive feedback and suggestion on how to present and write better.

# **List of Figures**

[Figure 1. Schematic diagram depicting Syncware’s position in bridging communication between two unique systems.](#_Toc99014435)

[Figure 2. Schematic diagram illustrating Syncware’s goal to create a fleet manager that addresses the lack of interoperability between different robot fleets.](#_Toc99014436)

[Figure 3. An example of an occupancy grid map showing three occupancy states.](#_Toc99014437)

[Figure 4. Flowchart of the general phases of image registration [24].](#_Toc99014438)

[Figure 5. Flowchart describing the steps taken by the basic stitcher class [56].](#_Toc99014439)

[Figure 6. Execution of original code on a map pair.](#_Toc99014440)

[Figure 7. (Top Row) Extracted masks indicating the locations of black pixels, and (Bottom Row) extracted masks indicating the positions of white pixels.](#_Toc99014441)

[Figure 8. Two masks denoting the locations of the black (Top) and white (Bottom) pixels.](#_Toc99014442)

[Figure 9. Close-up inspection of the blending effect for the merged masks for black pixels.](#_Toc99014443)

[Figure 10. Merged map after modifications to basic stitcher class.](#_Toc99014444)

[Figure 11. Diagrammatic description of the test sequence for each FDD algorithm utilised.](#_Toc99014445)

[Figure 12. Flowchart of steps to generate merged map from multiple images [69].](#_Toc99014446)

[Figure 13. Graph showing matches between four map images during merging.](#_Toc99014447)

[Figure 14. Merging four manually cropped out map images of the Freiburg Campus, Building 079.](#_Toc99014448)

[Figure 15. Masks representing the locations of white (Top) and black (Bottom) pixels.](#_Toc99014449)

[Figure 16. Merged map image after modifications.](#_Toc99014450)

[Figure 17. Simulated warehouse world created on Gazebo (Left) and a simulated MiR robot in its designated launch-point (Right)](#_Toc99014451)

[Figure 18. Overlapping maps produced with MiR robots using HectorSLAM.](#_Toc99014452)

[Figure 19. Merged map of simulated warehouse.](#_Toc99014453)

[Figure 20. Proposed architecture of communication bridge between Syncware and RNS.](#_Toc99014454)

[Figure 21. Sending of commands to a Turtlebot3 simulation world via RNS using the communication bridge.](#_Toc99014455)

[Figure 22. Snippet of dashboard showing information retrieve from sensor via communication bridge.](#_Toc99014456)

[Figure 23. Map Pair No.1](#_Toc99014457)

[Figure 24. Map Pair No.2](#_Toc99014458)

[Figure 25. Map Pair No.3](#_Toc99014459)

[Figure 26. Map Pair No. 4](#_Toc99014460)

[Figure 27. Map Pair No.5](#_Toc99014461)

[Figure 28. Pair 1 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).](#_Toc99014462)

[Figure 29. Pair 2 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).](#_Toc99014463)

[Figure 30. Pair 3 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).](#_Toc99014464)

[Figure 31. Pair 4 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).](#_Toc99014465)

[Figure 32. Pair 5 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).](#_Toc99014466)

[Figure 33. Pair 1 - Graph of Number of Inlier Matches against Rotation Applied.](#_Toc99014467)

[Figure 34. Pair 1 - Graph of Number of Inlier Matches against Scale Applied.](#_Toc99014468)

[Figure 35. Pair 2 - Graph of Number of Inlier Matches against Rotation Applied.](#_Toc99014469)

[Figure 36. Pair 2 - Graph of Number of Inlier Matches against Scale Applied.](#_Toc99014470)

[Figure 37. Pair 3 - Graph of Number of Inlier Matches against Rotation Applied.](#_Toc99014471)

[Figure 38. Pair 3 - Graph of Number of Inlier Matches against Scale Applied](#_Toc99014472)

[Figure 39. Pair 4 - Graph of Number of Inlier Matches against Rotation Applied.](#_Toc99014473)

[Figure 40. Pair 4 - Graph of Number of Inlier Matches against Scale Applied.](#_Toc99014474)

[Figure 41. Pair 5 - Graph of Number of Inlier Matches against Rotation Applied.](#_Toc99014475)

[Figure 42. Pair 5 - Graph of Number of Inlier Matches against Scale Applied.](#_Toc99014476)

**List of Tables**

[Table 1. Handcrafted Feature-Detector-Descriptors considered for map merging.](#_Toc99014477)

[Table 2. Handcrafted Feature-Detector-Descriptors considered for map merging.](#_Toc99014478)

[Table 3. Summary of functionalities of the seven layers of the OSI model [45] .](#_Toc99014479)

[Table 4. Map location and source details for each image pair.](#_Toc99014480)

[Table 5. Characteristic(s) and reason(s) for selection for each image pair.](#_Toc99014481)

[Table 6. Input map images and their respective masks.](#_Toc99014482)

[Table 7. Inlier STDEV/Mean for Rotation and Scale Segments.](#_Toc99014483)

[Table 8. Success states for each algorithm for rotation test segment.](#_Toc99014484)

[Table 9. Success states for each algorithm for scale test segment.](#_Toc99014485)

[Table 10. Table of number of matches, number of inlier matches, transformation confidence, and success state for each FDD in the merging of warehouse world.](#_Toc99014486)

# **List of Symbols**

Occupancy grid map (matrix form)

Occupancy grid map (raw form)

Robot’s pose

Transformation function

Translation factor

Odometrical estimate

Robot’s pose (x coordinate)

Robot’s pose (y coordinate)

LIDAR scan distance measurement

# **1. Introduction**

## **1.1 Background Information**

According to the Annual Warehouse and Distribution Center (DC) Operations Survey 2021 [1], the most significant challenges faced by warehouse operators were: (1) the inability to attract and retain a qualified hourly workforce, (2) limited space for inventory space, and (3) inaccurate inventories caused by human error. These challenges were exacerbated by the ongoing COVID-19 pandemic that disrupted the availability of a qualified workforce and triggered the surge in e-commerce volume that warehouses cannot accommodate effectively [1]. As such, warehouse owners needed a solution that has a consistent throughput with reduced reliance on manual processes, and warehouse automation could be the answer to these challenges.

With the continual progression of advanced technologies, robotic systems have experienced increases in their affordability and advances in their capabilities [2].Due to this, warehouse automation is presented as a legitimate solution to the needs of warehouse owners as it offers consistent services that can operate round-the-clock, operate in very compact spaces, and require little to no human intervention. Warehouse automation can consist of multiple types of robotic systems, which include Automated Storage and Retrieval Systems (AS/RS) and Automated Mobile Robots (AMRs) [3].

When an increasing number of segments in a warehouse operation become automated, companies will employ specific robotic systems that cater to particular functions and use-cases, resulting in a “multi-diverse robotic solution” [4]. For instance, some AMRs can only transport items around the warehouse, others are created for security and patrolling purposes, while some are made to carry out disinfection and cleaning. However, with multiple robotic system vendors, each devising their unique solution, warehouse operators face limitations in implementing this “multi-diverse robotic solution” [4]. This comes under a more prominent topic of interoperability in which there is a lack of compatibility in communication forms between different robotic systems. [5]Interoperability is the ability of multiple machine systems to exchange and process information, and in this aspect, it is a crucial driver for multi-diverse systems to collaborate. In the absence of interoperability, AMRs from different vendors can experience collisions and asynchronous transportation of goods that can delay warehouse operations.

This Final Year Project (FYP) is provided by a warehouse robotics middleware software startup [6], *Syncware*, that aims to build a connectivity platform for different robots to connect to regardless of the communication protocol utilised. With reference to Figure 1, the objective is to allow different systems to communicate through a central medium that is the Syncware Platform. Furthermore, information extracted from these systems will also be presented to users on the Graphical User Interface (GUI) of the platform.

A picture containing logo

Description automatically generated

Figure 1. Schematic diagram depicting Syncware’s position in bridging communication between two unique systems.

Relating back to the aspect of interoperability, *Syncware* aims to also create an optimized Fleet Manager (FM) that is able to handle a fleet of different robotic systems each utilizing different communication protocols. FMs are systems which allow for a centralized management of a robot fleet, usually from the same vendor utilizing a single standard communication protocol and are responsible for the scheduling and assignment of tasks, the planning of routes, and the monitoring of fleet status (i.e., robot battery level, mission completion status, and robot location coordinates) [7]. For these functions, it is evident that, FMs require an intensive exchange of information between each robot in the fleet and its system. With reference to Figure 2, *Syncware* needs to create its own communication bridge to replicate this process of data transfer between its platform and the robot fleets.

Diagram

Description automatically generated with medium confidence

Figure 2. Schematic diagram illustrating Syncware’s goal to create a fleet manager that addresses the lack of interoperability between different robot fleets.

Apart from the creation of an optimised fleet manager, *Syncware*’s platform will also contain a wide array of value-add utilities that could potentially supplement the overall user experience. One such utility is a map merging tool that is able to combine partially overlapping 2D robot maps which can potentially reduce time consumption in mapping out an enclosed area **[8]**.

# **2. Literature Review – Map Merging**

Before delving into the methods available for merging of maps, the type of map that is of interest here needs to be exemplified.

### **2.1 Occupancy Grid Maps**

The type of map of interest in the map merging section of this project is an Occupancy Grid Map (OGM). The reasons for targeting this type of map representation is that it is suitable for building unstructured environments (indicates robustness) [9], generation of a map is straight-forward [10], mapping is rapid as no additional environmental features are extracted [11], and map resolution can be adjusted by resizing the grids. Furthermore, occupancy grid maps are one of the most popular forms of map representations in the field of robotics [12]. An OGM is a binary representation of a mapped environment, in which each grid is assigned a probability value that signifies its occupancy state. In practice, a probability value within the range of 0.0 to 0.1 denotes that the area is “free”, while a value within the range of 0.9 to 1.0 denotes that the area is “occupied” [13]. With reference to Figure 2, the grey areas represent unknown regions, the white areas represent unoccupied space in which the robot can roam freely, and the black areas represent occupied regions in which there exists a physical obstacle (i.e., shelves and walls).

Diagram

Description automatically generated

Figure 3. An example of an occupancy grid map showing three occupancy states.

OGM are generated via mapping algorithms such as *GMapping* and *Cartographer*, whereby having known the change of position of the robot (wheel odometry) and the distance of a physical obstacle from the robot (LIDAR scanner), a grid is assigned a corresponding probability value [14]. When the mapping sequence is completed, a 2D grayscale image representation is generated and is akin to how the environment is seen when viewed directly from above.

Through the utilization of probabilistic localization algorithms such as the *Adaptive Monte Carlo Localisation* (AMCL) method, an autonomous robot can estimate the following key information: (1) the robot’s pose with respect to the generated occupancy grid map of the environment, (2) information regarding the robot’s traversal movement, (3) sensor measurements that relate the robot’s pose to objects (i.e., walls and shelves) in the occupancy grid map [15]. With these estimations, along with the generated map, robots are capable of autonomously navigating through the environment with little to no human intervention. Despite the advantages that the occupancy grid map present, the issue with which the optimisation of fleet managers is concerned, is the time consumed in generating a map. The time taken for a single robot to map out an area of interest is proportional to the size of the area. Mapping with a single robot can also be inefficient as the robot has only a single attempt to produce a map, and any errors during the mapping process, such as collision with the walls that causes erroneous readings by the odometry, will render the map unusable. Map merging can alleviate these limitations by generating a global map from multiple overlapping sub-maps created by multi-robotic systems, making the mapping process faster as users can run the robots simultaneously. In the event that one of the sub-maps is deemed unsatisfactory for usage (i.e., large presence of noise), the robot will only have to execute the mapping process for the corresponding area. An application that this can bring about, is that if an area were to be modified intentionally (i.e., addition of partitions), the global map can be simply updated by running the map merging process on the updated sub-maps.

### **2.2 Occupancy Grid Map Merging Methods**

The next three sections will describe existing map merging methods employed in actual mapping scenarios. The three map merging methods are: (1) Probability Method, (2) Optimisation Method, (3) Feature-Based Method. Note that method (1) describes the continuous process of merging maps while the robots map out an area of interest, while methods (2) and (3) describe processes which merge partially overlapping maps that have been extracted at the end of the mapping sequence.

#### **2.2.1 Probability Method**

This form of map merging takes on the fact that OGMs are generated based on probabilities.

This form of map merging method is employed when the initial poses of the robots are assumed to be known. Assuming the initial pose of each robot is known by every other robot apart of the map merging process, merging the maps with the probability method is analogous to estimating the trajectory of the robot and the global map. This map building process is generally described by the following equation [16]:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

With reference to Equation (1), represents the pose of robot *i* at time *t*, and is fully expressed as where and describe the robot’s coordinates and denotes the robot’s orientation. In addition, denotes odometrical estimate of the robot’s pose sampled at time *t*, while denotes the robot’s odometrical trajectory up to time *t*. Furthermore, is the distance measurement of the LIDAR scan acquired at time *t* and denotes the map of robot *i*. Assuming that the parameters between each robot are independent, the equation can be simplified to the right-hand side (RHS) of Equation (1). The starting term on the RHS of Equation (1), denotes the map density with known robot trajectories and LIDAR scan measurements.

An implementation of this general description was presented in the paper “Collaborative multi-robot exploration”. Having known the relative positions of the robots, the OGMs supplied by each individual robot are merged via the following equations [17]:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |
|  |  | (3) |
|  |  | (4) |

With reference to equations (2), (3), and (4), denotes the probability that the location of coordinates [*x*, *y*] in the global coordinate frame is occupied in the map of robot *i*, while denotes the probability of the grid at location [*x*, *y*] being occupied by obstacles. Ultimately, through this proposed method, the researchers observed an approximate reduction of 50% in mapping time when three robots were engaged as compared to a single robot [17].

#### **2.2.2 Optimisation Method**

Optimisation refers to the process of maximizing or minimizing the value of an objective function under specific constraints [18]. Optimisation is widely utilised in the field of Machine Learning (ML) and often serves as a method to evaluate the performance of a ML model with respect to the given data. Specifically, ML models learn by optimizing a loss function (i.e., minimizing the mean squared error) that represents the magnitude of error in the prediction of the model [19]. Focusing on the topic at hand, the optimisation method is also utilised in the merging of maps. However, a key step needs to first be fulfilled, and that is to design a suitable objective function to optimize. One such example will be described in this section.

Before delving into the objective functions utilised, the OGMs are first processed. Firstly, the probability values in each grid of the OGM is discretized and transferred to a matrix *m* of the same dimensions [20], and is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

where , and denotes the matrix that contains positive integers after discretization. On the other hand, the rigid transformation function between a pair of maps is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

where and represent the horizontal and vertical translations between the pair of maps respectively, while represents the angle of rotation between the pair of maps. The column matrix represents the grid coordinates in map .

The first of the two objective functions is based on the magnitude of overlap between a pair of maps. This equates the map merging problem to the process of estimating a satisfactory rigid transformation to maximize the level of coincidence between the map pair. The extent of coincidence is defined as [20]:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |
|  |  | (8) |

With the optimisation approach, the overlapping function  is designated as the objective function to be maximized. Solely maximizing the objective function of will yield the optimal rigid transformation function that can merge the pair of maps. However, the downside is that optimizing this objective function requires an iterative exhaustive search sequence which can drastically increase the computational costs [21].

#### **2.2.3 Feature-Based Method**

With the occupancy grid maps being presented as images, the problem of map merging becomes vastly similar to the stitching of overlapping images as seen in panorama stitching systems of phone cameras [22]. Panorama stitching comes under the umbrella of image registration, which is a process of “transforming different sets of data into one coordinate system” [23], and in this case, transforming a set of images into a common coordinate frame such that the overlapping regions superpose, and the images align with each other. This is the goal of image stitching, and hence map merging. Image registration is actively applied in areas such as aerial surveillance, satellite imagery, and medical imaging, whereby an overall consolidated image is produced from a series of multiple overlapping images.

Focusing on the topic of map merging via feature-based methods, there are five key stages in which the overlapping maps will have to go through before it comes out as a consolidated map image. With reference to Figure 3, the five stages are: (1) Feature Detection and Description, (2) Feature Matching, (3) Outlier Rejection, (4) Deviation of Transformation Function, and (5) Image Reconstruction and Stitching [24].

Diagram

Description automatically generated

Figure 4. Flowchart of the general phases of image registration [24].

In the first phase of “Feature Detection and Description”, features are extracted from the maps that are to be fused. These are points of interest that will play a crucial role in the occupancy grid map merging process [25]. The main idea is that features detected within the overlapping regions of the images are similar, and this will lay the foundation for the next phase of “Feature Matching”. The types of interest points detected depends on the type of feature detector utilized, and common ones include point features (i.e., Scale-Invariant Feature Transform (SIFT) features [26]), line features (i.e., line segments and arcs [27]), and geometric features (i.e., rectangles [28]). Once the detection sequence has been completed, the features are described with key information relating to the “unique patterns possessed by their neighbouring pixels” [24]. This creates a numerical identity that allows the differentiation of one feature from another, and these identities are known as descriptors.

The second phase of “Feature Matching” involves the matching of described features that are numerically identical. There are different approaches to matching features, and they are: (1) Threshold-Based Matching, (2) Nearest Neighbour Matching, (3) Nearest Neighbour Distance Ratio Matching [29]. Descriptors are compared across images. For threshold-based matching, two regions are matched only when the distance between their descriptors is below a specified threshold value. For nearest neighbour matching, two regions are matched if the distance between the descriptors are below a certain threshold value and the region is the nearest neighbour to the region to be matched to. In this case, a single match is returned. The nearest neighbour distance matching ratio differs from the former approach and applies the threshold to the distance ratio between the first and second nearest neighbour. The first nearest neighbour is only considered a match when the distance to the reference feature is smaller than that of a fraction of the distance to the second nearest neighbour [29].

Despite the measures put in place for the selection of the best feature matches, outliers will still be present, and the third phase of “Outlier Rejection” comes into play. This is a crucial step that ensures a generation of a cleaner and accurate transformation function. Some of the probabilistic methods for eliminating outliers include: (1) Random Sample Consensus (RANSAC) [30], (2) M-estimator Sample Consensus (MSAC) [31], and (3) Progressive Sample Consensus (PROSAC) [32]. The transformation function is also known as the homography matrix, and it transforms a map image with respect to the referencing map image such that they are aligned in the same coordinate frame [33].

Despite the straightforwardness of the feature-based method, it is not without its limitations. Liang et al. stated that current methods for merging maps are not robust to great differences in scale [34], while Luo et al. stated that false matches tend to be more prevalent when there exist high levels of repetitive patterns and symmetry [35].

##### **2.2.3.1 Feature Detector and Descriptor Algorithms**

Below are tables summarising some of the available algorithms that are capable of either keypoint detection and description or just feature description. Table 1 represents feature detectors and descriptors (FDDs) that are manually engineered [26, 36 – 40], while Table 2 represents FDDs that are learned through the means f neural network training [35, 41, 42]. The reason for considering the learned variants is because they have shown significant improvements over state-of-the-art traditional algorithms such as SIFT and are able to produce features that possess higher discriminative power [43]. This potentially enhances the matching process and could potentially lead to a smaller quantity of false matches which could jeopardize the map merging sequence.

Table 1. Handcrafted Feature-Detector-Descriptors considered for map merging.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm Name** | **Algorithm Function** | **Brief Description** | **Performance** |
| SIFT | Keypoint Detection and Feature Description | Detects and describes interests points in an image that are robustly invariant to (1) scale, (2) rotation, (3) illumination, (4) limited affine variations. High computational cost. | Ability to detect high quantity of features:  **ORB > BRISK > SURF > SIFT > AKAZE > KAZE**  Computational efficiency:  **ORB > BRISK > SURF > AKAZE > SIFT > KAZE**  Efficiency of feature matching:  **ORB > BRISK > AKAZE > KAZE > SURF > SIFT**  Speed of image matching:  **ORB > BRISK > AKAZE > KAZE > SURF > SIFT**  Accuracy in Image Matching:  **SIFT and BRISK overall best for all types of geometric transformations, with SIFT as the most accurate** |
| SURF | Keypoint Detection and Feature Description | Similar to SIFT but with lower computational cost. Little affine invariance |
| BRISK | Keypoint Detection and Feature Description | Detects and describes interests points in an image that are invariant to (1) scale, (2) rotation, (3) limited affine changes. |
| ORB | Keypoint Detection and Feature Description | Detects and describes interests points in an image that are somewhat invariant to (1) scale, (2) rotation, (3) limited affine changes. |
| KAZE | Keypoint Detection and Feature Description | Detects and describes interests points in an image that are somewhat invariant to (1) scale, (2) rotation, (3) limited affine changes, (4) distinct at varying scales. |
| AKAZE | Keypoint Detection and Feature Description | Detects and describes interests points in an image that are somewhat invariant to (1) scale, (2) rotation, (3) limited affine changes, (4) distinct at varying scales. |

Table 2. Handcrafted Feature-Detector-Descriptors considered for map merging.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm Name** | **Algorithm Function** | **Brief Description** | **Performance** |
| SIFT-GeoDesc | Feature Descriptor | A novel local descriptor learning approach that integrates geometry constraints from multi-view reconstructions. SIFT detector was used to incorporate scale invariance into the descriptor. | Median number of inlier:  **ContextDesc > GeoDesc by ~75%**  Recall evaluation on benchmark datasets:  **ContextDesc > GeoDesc by ~ 5%** |
| SIFT-ContextDesc | Feature Descriptor | A unified learning framework that leverages and aggregates the cross-modality contextual information. Seeks to enhance local feature description with extra prior knowledge from a global perspective. SIFT detector was used to incorporate scale invariance into the descriptor. |

### **2.3 Consolidation**

From the research literature of the methods available for the merging of OGMs, it was concluded that the feature-based method was the most suitable for this project’s purpose. The main reason for this selection is due to the straightforwardness and the simplicity of the steps utilised in this map merging method. With the steps required (Figure 3) being fixed in their sequence, this allows the option to explore and optimize a specific step without affecting the rest. In addition, unlike the other two methods which are research-based not publicly available for usage, resources are extensive and readily available for this particular method. For example, OpenCV is an open-sourced library for computer vision functions that provides tutorials that explain the execution of each step found in Figure 3 [44]. Furthermore, with the rise of neural networks, there have been new algorithms that attempt to incorporate deep learning into the step of “Feature Detection and Description”, some of which were indicated to overcome the limitations posed by handcrafted variants (Table 2). This presents itself as an opportunity to compare the performance of learned variants and handcrafted variants in the process of map merging.

# **3. Literature Review – Communication Bridge**

Due to the differences in communication protocols adopted by different robotic systems, understanding how communication is executed is crucial and is what will be covered in the next section.

**3.1 OSI Networking Model**

The Open Systems Interconnection model (OSI model) is a framework that describes and standardizes communication functions of a telecommunication or computing system. The interoperability of different communication systems with respect to communication protocols, is the purpose of the OSI model [45]. Structurally, the OSI model divides the flow of information in a communication system into seven layers, each having its own specific function and the type of data being transmitted. From the lowest level of physical implementation of transmitting bits across a medium (i.e., copper wire) to the highest-level representation of data of a distributed application [45]. A distributed application is a system with components located in different telecommunication systems, and intercommunication is carried out by transmission of messages from one system to another [46]. Between the top and bottom layers, each other layer is served by the layer directly below it, and serves the layer directly above it [45]. Table 3 is a summary of the functions each layer represents, the data unit of focus, and the examples of some communication protocols that belong to the layer [47].

Table 3. Summary of functionalities of the seven layers of the OSI model [45] .

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer** | **Data Unit** | **Function** | **Examples** |
| 7. Application | Data | High-level APIs, including resource sharing, remote file access, directory services and virtual terminals | HTTP, FTP, MODBUS |
| 6. Presentation | Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption | ASCII, JPEG |
| 5. Session | Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes | PAP, RPC |
| 4. Transport | Segments | Reliable transmission of data segments between points on a network, including segmentation, acknowledgement, and multiplexing | TCP, UDP |
| 3. Network | Packet / Datagram | Structuring and managing a multi-node network, including addressing, routing and traffic control | IPv4, IPv6 |
| 2. Data Link | Bit / Frame | Reliable transmission of data frames between two nodes connected by a physical layer | IEEE 802.3/802.2 |
| 1. Physical | Bit | Transmission and reception of raw bit streams over a physical medium | Copper twisted wire |

**3.2 Communication Protocols**

Out of the communication protocols available, this literature review of the communication bridge will look into the ones that are common in communication systems of robotic devices. The common protocols include HTTP [48] and TCP [49], and of which both will be covered in the following sections.

**3.2.1 HTTP**

The Hypertext Transfer Protocol (HTTP) belongs to the application layer of the OSI model, and is responsible for encoding and exchange of information between a webserver and a client (i.e., a web browser such as Google Chrome). The type of information that is dealt with by the protocol comes in the form of hypertext documents, which are ordered texts that utilises logical links between nodes containing text. These documents can be altered using the Hypertext Markup Language (HTML). In real-time application, clients are able to request for different kinds of content (i.e., images, videos, text) from the web servers that host the data requested [50]. Built alongside HTTP is an architectural framework called the Representational State Transfer (REST) which utilises HTTP for its application layer functionality [51].

REST is a set of conventions that are utilized for creating web services, and REST Application Programming Interface (API) is an approach of accessing these web services without any processing [52]. All communication executed through the use of REST APIs, utilise HTTP requests. Requests are sent to the server in the form of web URLs while specifying the nature of the request. The types of requests are GET, POST, PUT, and DELETE. Regardless of the request made, a response is returned to the client in a variety of forms which include XML and JSON data information [52]. For example, *Boston Dynamics’* autonomous four-legged robot, Spot, allows for the extraction of data (i.e., battery status) and control (i.e., teleoperation) through the use of REST API calls [53].

**3.2.2 TCP**

The Transmission Control Protocol (TCP) is a communication style that allows telecommunication devices to exchange messages via a network. The TCP communication protocol ensures the successful transportation of data packets over the Internet by first establishing a connection between the destination and the source of transmission [54]. After establishing a connection, large data is segmented down into smaller data packets and is transmitted to the specified destination.

The default style of data transfer over the Internet is the Transmission Control Protocol/Internet Protocol (TCP/IP), and it ensures the accuracy of data transmission between specified devices. The important aspect of this method is that the device’s IP address needs to be specified, and every device has its own unique IP address within a local network. Working together, TCP and IP ensures that data integrity is preserved throughout the communication process [54]. For example, OMRON’s LD-90 AMR utilises TCP/IP to communicate with external systems such as fleet managers [55].

**3.3 Consolidation**

For this project, the understanding of the communication protocols commonly used by robotic systems will aid in the construction of the architectural skeleton of the communication bridge. There exist multiple open-source libraries that support these communication protocols (i.e., requests and socket), and basically, wrappers need to be created to accommodate each protocol effectively. The types of the communication protocol that will be dealt with, will depend on the availability of collaborators that engage with *Syncware*.

Please note that this report cannot disclose certain aspects of the communication bridge (i.e., actual code, communication style used by Syncware, and etc.) because it is utilised by the company and releasing details will uproot the products integrity.

# **4. Project Objectives**

## **4.1 Map Merging Tool**

Before any autonomous navigation can be executed, a 2D map needs to be supplied. These maps are created via mapping algorithms such as GMapping, and through the combination and processing of data from the robot’s scanners and wheel odometry, a 2D map can be generated as a single robot traverses the environment. When efficiency is of a concern, having a single robot to map a whole area is inefficient. Instead of having a single robot to map the entire area, a map can be generated from multiple overlapping maps generated by a robot fleet. This shortens the time taken to create a map, and is the main motivation that drives the creation of the map merging tool.

The feature-based method for the merging of OGMs is selected for this project, and an evaluation of the performance of handcrafted FDDs and learned FDDs will be presented in section 6.2.3.

## **4.2 Communication Bridge**

Information from the fleet manager(s) will be extracted and mapped to the Syncware platform, and this will be processed and distributed to all robots for localization and navigation in an area of interest. In this context, the proposed communication bridge solution must be able to synchronise different forms of communication employed by fleet managers via the conversion of extracted information into a common data type that is processed and stored on the Syncware platform.

# **5. Project Scope**

## **5.1 Map Merging Tool**

The map merging tool needs to be capable of merging multiple OGMs regardless of three factors: (1) differences in orientation, (2) differences in scale, and (3) the ordering of input OGMs.

In order to accomplish this, the following preparatory and testing stages were needed:

1. Replicating a basic pair-image stitcher to understand the fundamental steps in stitching images.
2. Evaluation of merging performance of various FDDs that considered in section 2.2.3.1 with the pair-image stitcher created in step A.
3. Modify and optimize OpenCV’s code pipeline for the Stitcher module for the merging of multiple OGMs [56].
4. Carry out application testing on multiple overlapping OGMs created in a warehouse simulation world with the tool created in step C.

## **5.2 Communication Bridge**

The communication bridge needs to be created in such a way that it is not only able to accommodate several communication protocols, but to also handle the transmission between the robotic system and the Syncware Platform in a structured fashion. In addition, it needs to be robust …

In order to achieve this, this project requires the availability of an external collaborator to provide their system for in-house testing.

1. Retrieve robotic system from external collaborator for in-house prototyping and testing.
2. Propose a structured architecture that can be generalized for all external system to utilise.
3. Following the proposed architecture, create the communication bridge and test it out.
4. Apply the same architecture to another system for prove of concept and prove of ease of integration.

# **6. Methodology – Map Merging**

## **6.1 Pair-Map Stitcher Phase**

The opening phase of the map merging section of this project involved laying the foundation for a strong understanding of how images are stitched together, and this was done through the replication of a basic stitcher class that was available online [57].

The code was written in Python and utilised OpenCV’s open-source library for its image processing and computer vision functions. Figure 6 describes the steps that were required to successfully stitch two overlapping map images together, and these steps will be further explained.

Diagram

Description automatically generated

Figure 5. Flowchart describing the steps taken by the basic stitcher class [56].

The “Detect and Describe” step is the first step that an image must undergo. It involves having a feature-detector-descriptor algorithm, such as Scale Invariant Feature Transform (SIFT) and Speed Up Robust Features (SURF), to identify interest points in an image, and followed by generating a description of the local appearance around the detected interest points. Ideally, the description generated is invariant to changes such as illumination, translation, scale, and rotation, but are dependent on the feature-detector-descriptor algorithm itself. Each descriptor usually comes in the form of a vector [58]. As a product of this step, a set of descriptors as well as a set of keypoints are produced [58]. Focusing on stitcher class being replicated, two input images were required, and therefore, two distinct sets of descriptors and keypoints were generated from the “Detect and Describe” step. With reference to Table 3 and 4, the eight FDDs, handcrafted as well as learned variants, were incorporated in this step.

The two sets of descriptors then undergo the “Matching of Features” step, whereby the descriptor of one feature in the first set is matched with all the features in the second set using a specified distance calculation optimal for each algorithm [59]. The fundamental approach of descriptor matching in this step is based on the comparison of the distance between descriptors [59]. The smaller the value of the distance calculated with respect to the others, the greater the likelihood of the corresponding descriptors to be of a match. For specific details, real-valued descriptors (i.e., SIFT and SURF features), utilises the Euclidean distance while binary descriptors (i.e., ORB and BRISK features), utilises the Hamming distance [60]. In this stitcher class, the overall matching was executed via the 2-Nearest Neighbour Matching approach, which meant that every descriptor in the first set possesses two best matches after the comparison of distance values [61]. After the “Matching of Features” step, a set of pairs of match objects [62] is generated (Equation 9). Each object possesses public attributes such as the distance and the index of the matching descriptors in their respective sets [62].

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

where denotes the length of the set and . The first object of the pair denotes the best match, while the other denotes the second-best match [61].

The set of pair of match objects then undergoes a processing step which reduces the set into a set of single match objects. The processing step is known as “Lowe’s Ratio Test” which was proposed by David Lowe in 2004, when he introduced the FDD algorithm of SIFT [26]. The purpose of this test is to eliminate matches that are deemed unsatisfactory, which could be due to the presence of background cluttering [26]. This aids in the discarding of potential false positive matches that were picked up in the “Matching of Features” step. A match is considered a “good match” only if the distance of the closest neighbour is sufficiently smaller than the distance of the second-closest neighbour [26]. Lowe argued that this step was necessary as “correct matches need to have the closest neighbour significantly closer than the closest incorrect match to achieve reliable matching” [26]. For this stitcher class, the best match object is only accepted when its distance is less than 70% of the distance of the second-best match object. As such, as a product of the “Lowe’s Ratio Test” step, a set of satisfactory single match objects was obtained.

With this set of single match objects, the matching descriptor indexes are known. The next step of the “Homography Estimation” step requires the coordinates of the keypoints to compute the transformation matrix [63]. These coordinates can be retrieved from the keypoint sets with the descriptor indexes. The transformation matrix is also known as the homography, and it relates the transformation between two planes. The homography matrix possesses eight degrees-of-freedom (DOF) and only needs four correspondence points to sufficiently produce the transformation matrix [63].

If the number of match objects in the set is fewer than four, the homography estimation step fails and the stitching process is aborted. This is likely to occur when the overlapping regions between the two images is of insufficient size.

On the other hand, if the number of match objects exceed four, the RANSAC method is engaged to eliminate the outlier homographies. Through the application of the RANSAC method, four random good matches are first selected and a homography is computed. This homography is evaluated by checking the number of good matches that are consistent with it, and those that are consistent are considered inliers [30]. For example, and are a good match, and with the initial homography estimated, transforming should produce a point that is close to or equivalent to [64]. This whole process is repeated a fix number of times, with each iteration deriving a homography with another set of four good matches and counting the number of inliers [64]. The homography that has the highest number of inliers is selected to be the final matrix produced [64].

The homography matrix is applied to Map 2, and the pixels of Map 1 is transferred to Map 2 to overlap the corresponding regions and produce the merged map.

One of the limitations of this stitcher class was that it required the images to be supplied in their true order (true-left followed by true-right) as the homography is derived on the basis that Map 2 is transformed into the plane of Map 1 [57]. A work-around solution was to supply the inputs in two orders and execute the stitcher class twice. A failed merge and a potentially successful merge will both be presented.

Another limitation is that the stitcher class is not capable of stitching more two map images in an iteration. This compromises the robustness of the map merging tool, and its effectiveness will suffer when the user has many maps to be merged. To enable the tool’s scalability, this limitation was resolved latter in section 6.3.

In addition to these limitations, the stitcher class could not effectively merge maps that were centred in an area of grey pixels, this is because panorama stitching only involves overlapping of images and assumes that that is no extra padding background present. Figure 6 is an output of the unaltered stitcher class, with the grey background of the left map covering the right map.

A picture containing text

Description automatically generated

Figure 6. Execution of original code on a map pair.

The solution to this was to determine the locations of all black and white pixels after the stitching was executed.

Diagram

Description automatically generated

Figure 7. (Top Row) Extracted masks indicating the locations of black pixels, and (Bottom Row) extracted masks indicating the positions of white pixels.

With reference to Figure 7, the approach was to first create two kinds of image mask for each input image. One image mask that indicates the location of all black pixels, while the other indicates the location of all the white pixels. In these image masks, only pixels with the value of 255 (observed as white in the image) indicates the presence of the specific colour that the mask is responsible for.

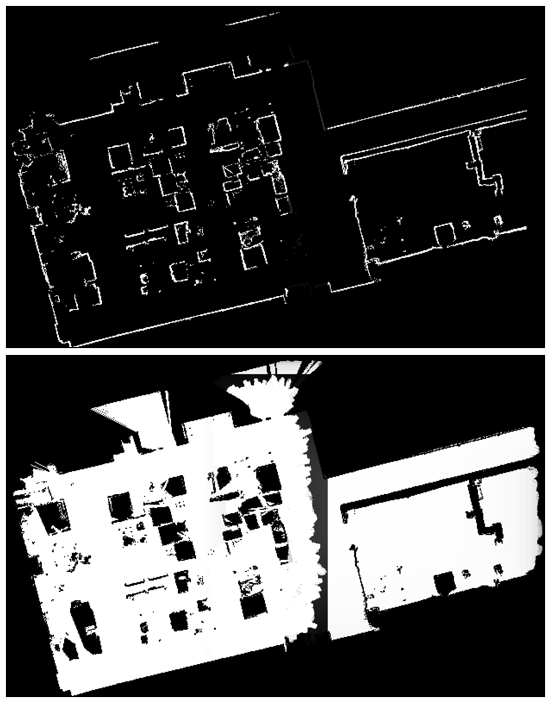


Figure 8. Two masks denoting the locations of the black (Top) and white (Bottom) pixels.

With reference to Figure 8, the homography matrix is also applied to these masks to overlap each other. However, there is an additional function applied to the overlapping step, and that is to add blending. This meant that the image will not completely cover the image that it overlapped, and there will a blend of colours that are contributed by each of the overlapping regions (Figure 9).

A picture containing application

Description automatically generated

Figure 9. Close-up inspection of the blending effect for the merged masks for black pixels.

With reference to Figure 9, it can be observed that there exists a slight blending of colours in the overlapping region.

With the merged masks, the location of all black and white pixels can be determined via the following logic. If the value of the pixel is non-zero (non-black regions), the location at which this pixel is at will contain a value corresponding to the colour the mask is responsible for. For example, with reference to Figure 8, the pixels in the merged white mask that possess non-zero values (non-black regions) indicate the presence of a coloured image pixel.

Having known the location of the black and white pixels in the final output, the merged image is altered by the following steps: (1) Overwrite the final output to contain only grey pixels, (2) overwrite the pixels at the locations that are indicated to be white, and (3) overwrite the pixels at the locations that are indicated to be black. With reference to Figure 10, the overlapping issue was resolved to produce a map that was close to the ground truth.

Diagram, engineering drawing

Description automatically generated

Figure 10. Merged map after modifications to basic stitcher class.

The product of this opening phase is a stitcher class that can stitch a pair of overlapping map images together. This class is called the “Pair-Map Stitcher”.

## **6.2 Feature-Detector-Descriptor Performance Evaluation Phase**

The main purpose of this phase is to evaluate the performance of each FDD that was implemented in the previous phase. For the execution of the evaluation, the testing will be carried out with the “Pair-Map Stitcher” class of the previous phase on a set of pair images.

### **6.2.1 Dataset Acquisition**

Due to absence of a benchmark dataset solely for the performance evaluation of FDDs in the stitching of overlapping OGMs, a dataset of five pairs of overlapping OGMs was manually created (See Appendix A).

Table 4. Map location and source details for each image pair.

|  |  |  |
| --- | --- | --- |
| Pair | Map Location | Source |
| 1 | Fort A.P. Hill (United States Army) | *RADISH* Repository |
| 2 | Advance Remanufacturing and Technology Centre (ARTC), Level 5, Arena | *Syncware Pte Ltd* |
| 3 | California Science Center, Mezzanine Level | *RADISH* Repository |
| 4 | DARPA’s Software for Distributed Robotics (SDR) program, Site B | *RADISH* Repository |
| 5 | Intel Lab (Hillsboro, Oregon) | *RADISH* Repository |

With reference to Table 4, the five image pairs were manually created out of available map images that were either retrieved online or provided by *Syncware* for the purposes of the project. Image pairs 1, 3, 4, and 5, were formed from OGMs that were publicly available on a repository called the Robotics Data Set Repository (RADISH). RADISH is a collection of standard robotics datasets which include logs of odometry, logs of laser scan data, and OGMs generated by actual robots [65]. Through the use of Microsoft Paint, the OGMs available in RADISH were manually cropped and drawn over to closely mimic the partial maps that a robot would generate. On the other hand, image pair 2 was provided by *Syncware* for the project. Table 5 contains information regarding the dimensions, perceived characteristics, as well as the reasons for selection of each of the five pairs of images.

Table 5. Characteristic(s) and reason(s) for selection for each image pair.

|  |  |  |  |
| --- | --- | --- | --- |
| Pair | Dimensions | Characteristic(s) | Reason(s) for Selection |
| 1 | True Left: (573, 536)  True Right: (581, 920) | * Simple environmental features * Small overlapping area | Tests the ability of the feature-detector-descriptor to generate sufficient number of feature descriptors for a higher likelihood of matching success. |
| 2 | True Left: (2240, 2176)  True Right: (1870, 2173) | * Ample distinctive features in the neighbourhood * Possesses large dimensions | Tests the ability of the feature-detector-descriptor to detect and compute descriptors efficiently. |
| 3 | True Left: (731, 2005)  True Right: (569, 1386) | * Simple structures with large areas of white spaces * Presence of symmetry | Tests the ability of the feature-detector-descriptor to create sufficiently discriminate feature descriptors for an effective matching of features. |
| 4 | True Left: (573, 645)  True Right: (573, 645) | * Medium degree of repetitive structures |
| 5 | True Left: (701, 602)  True Right: (855, 857) | * High degree of repetitive structures |

### **6.2.2 Test Execution**

This test serves to evaluate the performance of each of the eight FDD algorithms with respect to the merging of the five pairs of map images with different transformations applied. The purpose of the application of synthetic transformation is to mimic the difference in states between OGMs generated by different robots. In actual robot operations, although the mapping algorithms utilised are relatively standardized, different robotic systems may set different map parameters that can affect final state. For example, the Robotics Operation System’s (ROS) Gmapping package allows for the modification of the delta parameter which changes the resolution of the map produced. The resolution of the map refers to the distance per occupancy grid block of an OGM [66]. For this test, only rotation and scaling are applied to the map images, as maps commonly tend to possess differences in these transformation states when multi-robotic systems execute the mapping process [67].

The evaluation focuses on a few key aspects that can potentially indicate that a particular FDD is suitable for the merging of OGMs, and they are: (1) Invariance to differences in transformation states between images in each pair, (2) low computational costs in detection and description of features, and (3) high success rates of merging.

Diagram, text

Description automatically generated

Figure 11. Diagrammatic description of the test sequence for each FDD algorithm utilised.

With reference to Figure 11, for each FDD algorithm used in the “Detect and Describe” step of the “Pair-Map Stitcher” class, the merging is executed iteratively over a fixed set of transformation values to be applied to one of the images in the pair. The test was partitioned into two segments: The rotation segment and the scale segment. Rotation is applied in increments of 15° from 0° to 180° inclusive, while scaling is applied in increments of 0.5 from 1.0 to 5.0 inclusive.

After the completion of each segment, data such as the (1) duration of feature-detection-description per feature point detected, (2) number of inlier matches produced at each transformation application, and (3) success rate of merge, were extracted and processed. Note that for the success rate of merge, the merge is deemed successful when the final product closely represents the map images before it was manually cropped.

### **6.2.1 Test Observations**

#### **6.2.1.1 Feature-Detection-Description Time**

With reference to Figures 23, 24, 25, 26, and 27 (See Appendix A), which are the bar charts of feature-detection-description timing per point for each algorithm for each image pair, the following observations were made:

* It was observed that the speed of the handcrafted algorithms (ORB, BRISK, SURF, SIFT, AKAZE, and KAZE) were consistently faster than that of the learned variants (SIFT-GeoDesc and SIFT-ContextDesc) by at least 50%.
* The overall ranking sequence for speed (fastest to slowest) is as follows: ORB > BRISK > SURF > AKAZE > SIFT > KAZE > SIFT-GeoDesc > SIFT-ContextDesc.
* Amongst the handcrafted variants, the algorithms that employ binary descriptors (ORB and BRISK) were consistently faster than those that employ real-valued descriptors (SIFT, SURF, AKAZE, and KAZE).
* Overall, ORB had the fastest timing per point, and this can be attributed to not on­ly its binary approach to describing features, but also, its use of the FAST corner detector that is widely known for its high-speed execution in video processing applications [38].
* Amongst the handcrafted variants, KAZE was the slowest, which is due to high computational costs of a constructing non-linear scale space for its feature detection process [39].
* SURF is faster than SIFT, and this can be attributed to its use of the Hessian detector in its detection step which has a faster computational time than the Difference of Gaussian detector used by SIFT [36].
* AKAZE is faster than KAZE, and this can be attributed to its use of the Fast Explicit Diffusion (FED) mathematical framework which accelerates the construction of a non-linear scale space utilised by KAZE in its detection step [40].

#### **6.2.1.2 Number of Inlier Matches against Applied Transformation**

With reference to Figures 28, 30, 32, 34, and 36 (See Appendix C), which are graphs of inlier matches against the rotation applied, the following observations were made:

* The number of inlier matches peak at angles of 0°, 90°, and 180° in the merging of all the image pairs. This could likely be due to the magnitude of distortion of pixels when rotation is applied at angles other than 0°, 90°, and 180°. A vertical straight line of pixels was observed to retain its shape and position when rotated at 0°, 90°, and 180°, while at the other angles the pixels were observed to be out of line and had a seemingly jagged side profile.
* ORB consistently obtains the highest number of inlier matches. This is likely due a detection of a substantial number of features that can potentially raise the likelihood of a successful match.
* No obvious overall ranking sequence was observed for the number of inlier matches obtained for the algorithms other than ORB.

With reference to Figures 29, 31, 33, 35, and 37 (See Appendix C), which are graphs of inlier matches against the scale applied, the following observations were made:

* ORB, KAZE, and AKAZE were observed to have a decline in the number of inlier matches with ORB having the largest declination over the scale segment.
* BRISK, SIFT, SURF, SIFT-GeoDesc, and SIFT-ContextDesc were observed to have roughly stable numbers of inlier matches.
* For Image Pair 2, KAZE failed to complete the feature-detection-description step above scale application of 3.5. The large dimensional size of the images coupled with a large number of features detected, drove up the KAZE’s high total computational costs, which led to an abortion of operation.

With reference to Table 6 (See Appendix D), which is a table containing ratios of standard deviation to mean inlier matches for each algorithm over the two test segments. For this test, values below 0.25 were considered optimal, and are highlighted in green. The following observations were made:

* Only SIFT and SIFT-GeoDesc had values below 0.25 which can potentially indicate a satisfactory degree of invariance to differences in rotation and scale.

#### **6.2.1.3 Success Rate of Merging**

With reference to Table 7 and 8 (See Appendix E), which are tables containing the success states of each FDD algorithm for the rotation and scale segment respectively. Note that “1” denotes a successful merge, while “-1” denotes a failed merge. The following observation was made:

* Only SIFT and SIFT-GeoDesc were observed to successfully merge all image pairs throughout both test segments.

### **6.2.6 Test Conclusions**

On the basis of feature-detection-description time per feature point, ORB and BRISK were found to be the most computationally efficient algorithms, while SIFT-GeoDesc and SIFT-ContextDesc were found to be the least. The learned variants employ the use of neural networks which have more costly operations being executed as compared to the handcrafted variants.

On the other hand, on the basis of the evaluation of inlier matches against transformation applied and the ratio of standard deviation to mean values, SIFT and SIFT-GeoDesc were found to possess features that were to a degree invariant to rotation and scale differences.

Lastly, on the basis of success rates of merge, SIFT and SIFT-GeoDesc had the best performance in the merging of this particular dataset. However, it should not be concluded that these algorithms are the “best” for the merging of OGMs as the test dataset might not be representative of all the maps out there.

## **6.3 Multi-Map Stitcher Phase**

The main aim of this phase was to overcome the limitations that were present in the “Pair-Map Stitcher” class. The limitations were that it could only merge two images per iteration and that the images needed to be supplied in its true order representative of the ground truth.

The solution found was to replicate a panorama stitcher module that was available in OpenCV’s library. This particular module is an implementation of David Lowe’s and Matthew Brown’s proposal of an automatic panorama stitcher utilising SIFT features [22] and is stated to be able to stitch multiple images regardless of the order of input. The code pipeline was extracted from OpenCV’s repository [68] and modifications were made to suit the needs of a robust map merging tool.

Diagram

Description automatically generated

Figure 12. Flowchart of steps to generate merged map from multiple images [69].

Figure 12 depicts a flowchart of the key steps used in this stitcher class. Similar to the previous class, the merging process starts off with the detection and description of features in all the input images with a selected FDD. This is followed by the matching of features across multiple images via pairwise matching. Pairwise matching involves matching the features in one image with the rest of the features in the other images, and this is executed till each image is matched with every other. As such, the computational time complexity is , where is the number of input images, and this makes it the most resource demanding step of the entire process. After the end of the matching step, the number of match object sets would be the square of the number of input images [22]. After the matching step, the output can be organised as a graph as shown in Figure 13.

Diagram

Description automatically generated

Figure 13. Graph showing matches between four map images during merging.

With reference to Figure 13, is the number of matches, is the number of inlier matches from RANSAC, and is the transformation confidence. The transformation confidence makes use of the number of inliers derived from RANSAC to approximate the accuracy of transformation, and is given by [22]:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

If the maximum number of iterations is reached the specified limit in the RANSAC process, the confidence value will be 0 and a transformation cannot be estimated. A match between images is only accepted when the confidence exceeds a threshold value that is set manually. In this process, the threshold value of 1.0 was used.

The next step is to determine the images and matches that will participate in the construction of the final merged image. With reference to Figure 13, this is necessary as there could be multiple connect components that are independent of each other. The approach to this issue will be to find the largest connect component in the graph [22], and all the images and matches in this component will participate in the next step of the estimation of transformation function.

Similar to the “Pair-Map Stitcher” class, homographies are computed between the images that are present in the largest connected component. A single map image is selected as a global reference frame, and all the other map images are transformed onto this frame with the relative homographies estimated. However, before transformation, bundle adjustment is used to solve all the homographies jointly. This is necessary as a concatenation of relative homographies would cause the accumulation of errors [22]. As such, with the application of the new set of homographies, the final merged map image is produced.

Using the code pipeline for the Stitcher class provided by OpenCV, a set of four overlapping map images, that were manually created from a map in RADISH [65], were supplied to be merged. This result is as shown in Figure 14.

Diagram

Description automatically generated

Figure 14. Merging four manually cropped out map images of the Freiburg Campus, Building 079.

With reference to Figure 14, the output of the class had the same blending issues and extra areas of black pixels near the boundaries of the merged image that were faced by the “Pair-Map Stitcher” class. The same steps were undertaken to produce a satisfactory merged map image.

Table 6. Input map images and their respective masks.

|  |  |  |
| --- | --- | --- |
| **Input Images** | **Masks for Black Pixels** | **Masks for White Pixels** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

A picture containing diagram

Description automatically generated

Figure 15. Masks representing the locations of white (Top) and black (Bottom) pixels.

**Diagram, engineering drawing

Description automatically generated**

Figure 16. Merged map image after modifications.

Firstly, create masks to denote the locations of black and white pixels (Table 6). Secondly warp and blend the respective masks together (Figure 16). Thirdly, overwrite the pre-determined locations in output image with grey followed by the white and then black pixels (Figure 17).

The final product of this phase is called the “Multi-Map Stitcher” class, and the next and last phase will be to test it out in a simulated environment.

## **6.4 Product Testing**

The last phase of the map merging tool creation process was to carry out an application testing in a simulated warehouse environment.

Graphical user interface

Description automatically generated

Figure 17. Simulated warehouse world created on Gazebo (Left) and a simulated MiR robot in its designated launch-point (Right)

With reference to Figure 18, a simulated warehouse world was created on Gazebo for simulated robots to traverse and execute the mapping process. An open-source robot simulation provided by Mobile Industrial Robots (MiR) was utilised in the simulated world [70].

Diagram

Description automatically generated

Figure 18. Overlapping maps produced with MiR robots using HectorSLAM.

With reference to Figure 18, multiple overlapping OGMs were produced by MiR robots and supplied to the “Multi-Map Stitcher” class for merging.

**Diagram

Description automatically generated**

Figure 19. Merged map of simulated warehouse.

With reference to Figure 19, the final output of the “Multi-Map Stitcher” class is a merged map that closely resembled the actual simulated environment.

With reference to Table 10 (See Appendix E), it was observed that all FDDs, except SIFT-ContextDesc, were successful in the merging of the input map images. It was observed that the learned variants detected much fewer matches than the handcrafted variants, and this likely due to the training set upon which these learned variants were trained on. The small number of matches led to a failed merge by SIFT-ContextDesc as the homography estimation step did not have sufficient matches to effectively approximate an acceptable transformation matrix.

Note that the confidence threshold for this test was set to 0.3, so that all the images will be considered in the final merged image. From Table 10, it can be implied that the smaller the transformation confidence value, the higher the likelihood of a failed merge.

# **7. Methodology – Communication Bridge**

## **7.1 Architecture Design Proposal**

In collaboration with a local robotics solution company, Movel AI, a communication bridge was created between the Syncware platform and Movel AI’s Robot Navigation System (RNS). RNS is a single robot management system designed to simplify the user’s experience in controlling a robot [71]. Robotic functions such as mapping, teleoperation, and localization can be executed from the Syncware Platform through this management tool via REST API requests.

A picture containing text, device, meter

Description automatically generated

Figure 20. Proposed architecture of communication bridge between Syncware and RNS.

With reference to Figure 20, the proposed architecture of the bridge consists of two main components: (1) the core and (2) the plugin. The purpose of a dual-component architecture is so that one component handles the communication with the external system, while the other handles the communication with the Syncware Platform. The main idea is that the core only needs to be modified to accommodate different communication protocols of the external systems.

The core is responsible for: (1) gathering information from RNS, (2) sending information to RNS (i.e., commands), (3) storing information for extraction by the plugin, and (4) establishing communications with the external device. The plugin is responsible for: (1) retrieving information stored in the core for the Syncware Platform, (2) retrieving information from the Syncware Platform for the core, and (3) establishing communications with the Syncware Platform.

Graphical user interface, diagram

Description automatically generated with medium confidence

Figure 21. Sending of commands to a Turtlebot3 simulation world via RNS using the communication bridge.

With reference to Figure 21, a test was performed using a turtlebot3 world. Teleoperation commands were sent from the Syncware platform to RNS through the communication bridge, which was passed on to the Turtlebot3 simulation world. The test was deemed successful after the simulated robot moved and a feedback of status of movement was received back on the Syncware platform.

## **7.2 Product Testing**

The robustness of integration of the communication bridge was evaluated when it was utilised to connect an external device to the Syncware Platform. Another collaborator engaged with *Syncware* to utilise its platform to visualize data extracted from various air quality sensors. The core of the communication bridge was modified to accommodate the communication protocol of Message Queueing Telemetry Transport (MQTT), while the plugin was left untouched.

Chart

Description automatically generated

Figure 22. Snippet of dashboard showing information retrieve from sensor via communication bridge.

Figure 22 shows the data gathered from the sensors and presented as time-series graphs on the dashboard of the Syncware Platform. From this application, it was shown that the communication bridge was robust as minimal modifications were required to establish a constant exchange of data between the device and the Syncware Platform. As proposed, the core was the only component in the communication bridge was needed to be modified.

# **8. Conclusions**

## **8.1 Map Merging**

In the field of robotics, map merging is an ongoing problem in which researchers sought to find the most optimal method of merging several partial overlapping maps into a common global map. The advantage of map merging is that it can greatly reduce the time consumption of mapping out an enclosed environment. It is inherently faster to produce a single global map with a fleet of robots as compared to a single robot, as multiple robots can execute the mapping process of different regions simultaneously. In addition, map merging can reduce the time consumed in the correction of errors in the global maps. With a multi-robot approach, errors found in a specific region of the map can simply be resolved by re-executing the mapping process in the corresponding region, and then, executing the merging on the new set of partial maps.

The map merging problem was treated as an image registration problem, and a tool was created using feature-based methods. The final product was formed from the modification of the Stitcher module code pipeline that was originally intended for the generation of panoramas from multiple overlapping photographs. Provided that there are sufficient distinctive features and area of overlap, the tool is capable of merging OGMs regardless of the order and the number of inputs. This makes it robust as the size of robot fleets increase.

Various FDDs were incorporated and were evaluated for their performance in the map merging process. Based on the evaluation test done, the handcrafted FDDs tend to produce a higher number of inlier matches than learned variants, which raises the likelihood of computing a transformation function that is close to ground truth. On the other hand, it was observed that the learned variants were likely to produce features of stronger scale and rotation invariance properties. However, for the product testing, it was observed that a smaller number of matches is likely to lead to a failed merge. Therefore, it would be more suitable to use handcrafted variants for a higher chance of a successful merge.

With that being said, simulated environments cannot replicate the intricacies of an actual environment, and might downplay the number of features that can actually be detected in a feature-rich setting. Learned FDDs can still possibly be a strong alternative, but further testing with physical robots needs to be carried out.

## **8.2 Communication Bridge**

Interoperability is a major issue for warehouses or any entities that intend to adopt an increasing number of automation solutions, and it focuses on the ability of the entire system to exchange data without any hindrance. With different communication protocols used by different robotic systems, adopting a “multi-diverse” robotic solution is challenging. As such, a communication bridge was created to allow for the smooth exchange of data between these systems regardless of the communication protocols used. The communication bridge is a dual-component link that comprises of the “core” and the “plugin”. The core directly handles the communication with the external system, while the plugin handles the communication with the Syncware Platform. Together, these components work hand-in-hand to allow for data to be sent to and from the external system simultaneously. Being dual-component, the communication bridge required little modifications when dealt with another system using a different communication protocol. For the project, the protocols of REST API and MQTT were encountered in collaboration with external parties, and the communication bridge showed its robustness in establishing connections.

# **9. Recommendations for Further Work**

## **9.1 Map Merging**

Despite the ability of the map merging tool, it still has issues pertaining to ensuring the success of the merge. Due to the tool’s dependency on the establishment of matching features between overlapping partial maps, the size of the area of overlap will greatly affect the success of the map merge. In addition, when users carry out the mapping process, it is unclear as to how much overlap is required for the success of the merge of their partial maps. There does not exist a quantitative measurement for the minimum amount of overlap required for a successful merge as every environment will have different features. An approach could be to set up a distinctive rendezvous point (cones arranged in a star-formation) for which all participating robots will have to encircle before generating a partial map. However, this is merely a work around solution that purely lies on the users’ side.

For future works, the map merging tool can be converted into an iterating execution program, whereby the merge happens as the robots are executing the merging process. With this, users will not need to focus on determining the amount of overlapping area for a successful merge. For robots employing the use of ROS, there is a costmap\_2d that stores the OGM for extraction [72], and this can be leverage by the iterating program.

Furthermore, this project focuses only on the merging of 2-dimensional maps. For future works, a map merging tool can also be created for 3-dimensional maps. This could be beneficial for applications outside of warehouses, such as Search and Rescue (SAR) missions in enclosed areas like caves, whereby rescue personnel need to understand the terrain before entering it [73].

## **9.2 Communication Bridge**

The communication bridge showed that it was sufficiently robust to handle both communication protocols of REST API as well as MQTT, however, it has yet to be tested with many other protocols such as TCP/IP. The lack of collaborators for the testing of the communication bridge limits the number of communication protocols it can handle. Ideally, as long as the core is able to accommodate and handle the communication protocol of interest, connection should be established without many modifications and interventions.

For future improvements, it would be beneficial that the communication bridge is able to handle all sorts of communication protocols without the need to modify the core. For example, the core has several stored modes communication, and the user needs to only specify the communication protocol in use. In addition, the communication bridge is currently a one-to-one form of information exchange, and a possible future improvement will be to move towards a many-to-many form of communication. This can potentially save computational resources as fewer communication bridges will be needed.

# **10. References**

[1] "2021 Warehouse/DC Operations Survey: Automation as a Disruption Response", *Logisticsmgmt.com*, 2021. [Online]. Available: https://www.logisticsmgmt.com/article/2021\_warehouse\_dc\_operations\_survey\_automation\_as\_a\_disruption\_response.

[2] A. Jenkins, "Warehouse Automation: The How, When & Why", *Oracle NetSuite*, 2022. [Online]. Available: https://www.netsuite.com/portal/resource/articles/inventory-management/warehouse-automation.shtml. [Accessed: 21- Mar- 2022].

[3] C. Monroy, "Warehouse Robotics & Different Types of Robots | 6 River Systems", *6 River Systems*, 2022. [Online]. Available: https://6river.com/what-is-warehouse-robotics/. [Accessed: 21- Mar- 2022].

[4] "[Report] Interoperability in the Robotics Industry | Meili Robots", *Meili Robots*, 2022. [Online]. Available: https://www.meilirobots.com/resources-list/report/robotics-interoperability. [Accessed: 21- Mar- 2022].

[5] M. Schlechtriem, "Interoperability: A Key Challenge for the Robotics Industry", *#HowToRobot*, 2022. [Online]. Available: https://www.howtorobot.com/expert-insight/interoperability. [Accessed: 21- Mar- 2022].

[6] "Syncware.ai - Middleware software for warehouse robotics", *Syncware.ai*, 2022. [Online]. Available: https://www.syncware.ai/. [Accessed: 21- Mar- 2022].

[7] "How are Fleet Management Systems implemented in different robotic companies?", *Medium*, 2022. [Online]. Available: https://medium.com/@robosera.info/how-are-fleet-management-systems-implemented-in-different-robotic-companies-cdce178c12f. [Accessed: 21- Mar- 2022].

[8] C. Nieto-Granda, J. Rogers and H. Christensen, "Coordination strategies for multi-robot exploration and mapping", *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 519-533, 2014. Available: 10.1177/0278364913515309.

[9] H. Li, M. Tsukada, F. Nashashibi and M. Parent, "Multivehicle Cooperative Local Mapping: A Methodology Based on Occupancy Grid Map Merging", *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2089-2100, 2014. Available: 10.1109/tits.2014.2309639.

[10] Z. Jiang, J. Zhu, Y. Li, J. Wang, Z. Li and H. Lu, "Simultaneous Merging Multiple Grid Maps Using the Robust Motion Averaging", *Journal of Intelligent & Robotic Systems*, vol. 94, no. 3-4, pp. 655-668, 2018. Available: 10.1007/s10846-018-0895-4.

[11] Z. Jiang, J. Zhu, C. Jin, S. Xu, Y. Zhou and S. Pang, "Simultaneously merging multi-robot grid maps at different resolutions", *Multimedia Tools and Applications*, vol. 79, no. 21-22, pp. 14553-14572, 2019. Available: 10.1007/s11042-018-7109-8.

[12] R. Siegwart and I. Nourbakhsh, *Introduction to autonomous mobile robots*. Cambridge, Mass: MIT, 2004.

[13] A. Birk and S. Carpin, "Merging Occupancy Grid Maps From Multiple Robots", *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384-1397, 2006. Available: 10.1109/jproc.2006.876965.

[14] "ROS Mapping and Localization", *Robotics Knowledgebase*, 2022. [Online]. Available: https://roboticsknowledgebase.com/wiki/common-platforms/ros/ros-mapping-localization/. [Accessed: 21- Mar- 2022].

[15] P. Pfaff, W. Burgard and D. Fox, "Robust Monte-Carlo Localization Using Adaptive Likelihood Models", *Springer Tracts in Advanced Robotics*, pp. 181-194. Available: 10.1007/11681120\_15.

[16] A. Howard, "Multi-robot Simultaneous Localization and Mapping using Particle Filters", *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243-1256, 2006. Available: 10.1177/0278364906072250.

[17] W. Burgard, M. Moors, D. Fox, R. Simmons and S. Thrun, "Collaborative multi-robot exploration", *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Available: 10.1109/robot.2000.844100 [Accessed 21 March 2022].

[18] "Mathematical optimization - Wikipedia", *En.wikipedia.org*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Mathematical\_optimization. [Accessed: 21- Mar- 2022].

[19] A. Chatterjee, "Optimization in Machine Learning — Part 1", *Medium*, 2022. [Online]. Available: https://medium.com/swlh/optimization-in-machine-learning-part-1-e9da1aa1eedf. [Accessed: 21- Mar- 2022].

[20] A. Birk and S. Carpin, "Merging Occupancy Grid Maps From Multiple Robots", *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384-1397, 2006. Available: 10.1109/jproc.2006.876965.

[21] M. Locatelli, "Simulated Annealing Algorithms for Continuous Global Optimization: Convergence Conditions", *Journal of Optimization Theory and Applications*, vol. 104, no. 1, pp. 121-133, 2000. Available: 10.1023/a:1004680806815.

[22] M. Brown and D. Lowe, "Automatic Panoramic Image Stitching using Invariant Features", *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59-73, 2006. Available: 10.1007/s11263-006-0002-3.

[23] "Image registration - Wikipedia", *En.wikipedia.org*, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Image\_registration. [Accessed: 31- Dec- 2021].

[24] S. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK", *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2018. Available: 10.1109/icomet.2018.8346440.

[25] S. Saeedi, M. Trentini, M. Seto and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review", *Journal of Field Robotics*, vol. 33, no. 1, pp. 3-46, 2015. Available: 10.1002/rob.21620.

[26] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004. Available: 10.1023/b:visi.0000029664.99615.94.

[27] Y. Alnounou, M. Paulik, M. Krishnan, G. Hudas and J. Overholt, "Occupancy Grid Map Merging using Feature Maps", *IASTED Technology Conferences / 705: ARP / 706: RA / 707: NANA / 728: CompBIO*, 2010. Available: 10.2316/p.2010.706-074.

[28] J. Park, A. Sinclair, R. Sherrill, E. Doucette and J. Curtis, "Map merging of rotated, corrupted, and different scale maps using rectangular features", *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2016. Available: 10.1109/plans.2016.7479743.

[29] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615-1630, 2005. Available: 10.1109/tpami.2005.188.

[30] M. Fischler and R. Bolles, "Random sample consensus", *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981. Available: 10.1145/358669.358692.

[31] P. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry", *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138-156, 2000. Available: 10.1006/cviu.1999.0832.

[32] O. Chum and J. Matas, "Matching with PROSAC — Progressive Sample Consensus", *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Available: 10.1109/cvpr.2005.221.

[33] C. Kim, H. Sung and H. Yoon, "Camera Image Stitching using Feature Points and Geometrical Features", *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 12, pp. 289-296, 2015. Available: 10.14257/ijmue.2015.10.12.29.

[34] L. Ma, J. Zhu, L. Zhu, S. Du and J. Cui, "Merging grid maps of different resolutions by scaling registration", *Robotica*, vol. 34, no. 11, pp. 2516-2531, 2015. Available: 10.1017/s0263574715000168.

[35] Z. Luo et al., "ContextDesc: Local Descriptor Augmentation With Cross-Modality Context", *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. Available: 10.1109/cvpr.2019.00263.

[36] H. Bay, T. Tuytelaars and L. Van Gool, "SURF: Speeded Up Robust Features", *Computer Vision – ECCV 2006*, pp. 404-417, 2006. Available: 10.1007/11744023\_32.

[37] S. Leutenegger, M. Chli and R. Siegwart, "BRISK: Binary Robust invariant scalable keypoints", *2011 International Conference on Computer Vision*, 2011. Available: 10.1109/iccv.2011.6126542.

[38] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF", *2011 International Conference on Computer Vision*, 2011. Available: 10.1109/iccv.2011.6126544 [Accessed 31 December 2021].

[39] P. Alcantarilla, A. Bartoli and A. Davison, "KAZE Features", *Computer Vision – ECCV 2012*, pp. 214-227, 2012. Available: 10.1007/978-3-642-33783-3\_16.

[40] P. Alcantarilla, J. Nuevo and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces", *Procedings of the British Machine Vision Conference 2013*, 2013. Available: 10.5244/c.27.13.

[41] Z. Luo et al., "GeoDesc: Learning Local Descriptors by Integrating Geometry Constraints", *Computer Vision – ECCV 2018*, pp. 170-185, 2018. Available: 10.1007/978-3-030-01240-3\_11.

[42] Z. Luo et al., "ASLFeat: Learning Local Features of Accurate Shape and Localization", *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. Available: 10.1109/cvpr42600.2020.00662.

[43] J. Schonberger, H. Hardmeier, T. Sattler and M. Pollefeys, "Comparative Evaluation of Hand-Crafted and Learned Local Features", *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. Available: 10.1109/cvpr.2017.736.

[44] "Home - OpenCV", *OpenCV*, 2022. [Online]. Available: https://opencv.org/. [Accessed: 21- Mar- 2022].

[45] "OSI model - Wikipedia", *En.wikipedia.org*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/OSI\_model. [Accessed: 21- Mar- 2022].

[46] "Distributed computing - Wikipedia", *En.wikipedia.org*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Distributed\_computing. [Accessed: 21- Mar- 2022].

[47] F. Feltus et al., "The Widening Gulf between Genomics Data Generation and Consumption: A Practical Guide to Big Data Transfer Technology", *Bioinformatics and Biology Insights*, vol. 91, p. BBI.S28988, 2015. Available: 10.4137/bbi.s28988 [Accessed 21 March 2022].

[48] B. Wessling, "TERAKI and Foodora to bring autonomous deliveries to Stockholm - The Robot Report", *The Robot Report*, 2022. [Online]. Available: https://www.therobotreport.com/teraki-and-foodora-to-bring-autonomous-deliveries-to-stockholm. [Accessed: 21- Mar- 2022].

[49] S. Bouchard, "What Are The Communication Protocols Used In Industrial Robotics?", *Blog.robotiq.com*, 2022. [Online]. Available: https://blog.robotiq.com/bid/32559/What-Are-The-Communication-Protocols-Used-In-Industrial-Robotics. [Accessed: 21- Mar- 2022].

[50] "An overview of HTTP - HTTP | MDN", *Developer.mozilla.org*, 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview. [Accessed: 21- Mar- 2022].

[51] "REST — Stop calling your HTTP APIs as RESTful APIs", *Medium*, 2022. [Online]. Available: https://medium.com/the-sixt-india-blog/rest-stop-calling-your-http-apis-as-restful-apis-e8336e3e799b#:~:text=REST%20stands%20for%20REpresentational%20State%20Transfer.&text=REST%20doesn't%20add%20any,for%20its%20application%20layer%20protocol. [Accessed: 21- Mar- 2022].

[52] "REST API (Introduction) - GeeksforGeeks", *GeeksforGeeks*, 2021. [Online]. Available: https://www.geeksforgeeks.org/rest-api-introduction/. [Accessed: 31- Dec- 2021].

[53] “Spot SDK — Spot 3.1.0 documentation", *Dev.bostondynamics.com*, 2022. [Online]. Available: https://dev.bostondynamics.com/. [Accessed: 21- Mar- 2022].

[54] "What is Transmission Control Protocol TCP/IP Model?", *Fortinet*, 2022. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/tcp-ip. [Accessed: 21- Mar- 2022].

[55] "About OMRON TCP/IP Communications | CIMPLICITY 11 Documentation | GE Digital", *Ge.com*, 2022. [Online]. Available: https://www.ge.com/digital/documentation/cimplicity/version11/oxy\_ex-2/device\_communications/topics/g\_cimplicity\_device\_communications\_omrneth\_diag\_about\_omron\_tcpip\_communications.html. [Accessed: 21- Mar- 2022].

[56] "OpenCV: High level stitching API (Stitcher class)", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/4.x/d8/d19/tutorial\_stitcher.html. [Accessed: 22- Mar- 2022].

[57] A. Rosebrock, "OpenCV panorama stitching - PyImageSearch", *PyImageSearch*, 2021. [Online]. Available: https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/. [Accessed: 31- Dec- 2021].

[58] "OpenCV: cv::Feature2D Class Reference", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/3.4/d0/d13/classcv\_1\_1Feature2D.html. [Accessed: 22- Mar- 2022].

[59] "OpenCV: Feature Matching", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/3.4/dc/dc3/tutorial\_py\_matcher.html. [Accessed: 22- Mar- 2022].

[60] "OpenCV: Operations on arrays", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/4.x/d2/de8/group\_\_core\_\_array.html#gad12cefbcb5291cf958a85b4b67b6149f. [Accessed: 22- Mar- 2022].

[61] "OpenCV: Understanding k-Nearest Neighbour", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/3.4/d5/d26/tutorial\_py\_knn\_understanding.html. [Accessed: 22- Mar- 2022].

[62] "OpenCV: cv::DMatch Class Reference", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/3.4/d4/de0/classcv\_1\_1DMatch.html. [Accessed: 22- Mar- 2022].

[63] S. Agarwal, "Homography - And how to calculate it?", *Medium*, 2022. [Online]. Available: https://medium.com/all-things-about-robotics-and-computer-vision/homography-and-how-to-calculate-it-8abf3a13ddc5#:~:text=Thus%2C%20homography%20can%20be%20calculated,and%20translation%20between%20two%20cameras. [Accessed: 22- Mar- 2022].

[64] "How is RANSAC used to estimate a homography given the descriptors of both images computed with SIFT?", *Computer Science Stack Exchange*, 2022. [Online]. Available: https://cs.stackexchange.com/questions/70434/how-is-ransac-used-to-estimate-a-homography-given-the-descriptors-of-both-images. [Accessed: 22- Mar- 2022].

[65] "Radish: Robotics Research Datasets", *Dspace.mit.edu*, 2022. [Online]. Available: https://dspace.mit.edu/handle/1721.1/62236. [Accessed: 22- Mar- 2022].

[66] "gmapping - ROS Wiki", *Wiki.ros.org*, 2022. [Online]. Available: http://wiki.ros.org/gmapping#Parameters. [Accessed: 22- Mar- 2022].

[67] J. Park, A. Sinclair, R. Sherrill, E. Doucette and J. Curtis, "Map merging of rotated, corrupted, and different scale maps using rectangular features", *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2016. Available: 10.1109/plans.2016.7479743 [Accessed 22 March 2022].

[68] "GitHub - opencv/opencv: Open Source Computer Vision Library", *GitHub*, 2022. [Online]. Available: https://github.com/opencv/opencv. [Accessed: 22- Mar- 2022].

[69] "OpenCV: Images stitching", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/4.x/d1/d46/group\_\_stitching.html. [Accessed: 22- Mar- 2022].

[70] "GitHub - dfki-ric/mir\_robot: ROS support for the MiR Robots. This is a community project to use the MiR Robots with ROS. It is not affiliated with Mobile Industrial Robots.", *GitHub*, 2022. [Online]. Available: https://github.com/dfki-ric/mir\_robot. [Accessed: 22- Mar- 2022].

[71] "Seirios RNS: Robotic Navigation System | Movel AI", *Movel.ai*, 2022. [Online]. Available: https://www.movel.ai/seirios/rns. [Accessed: 22- Mar- 2022].

[72] "costmap\_2d - ROS Wiki", *Wiki.ros.org*, 2022. [Online]. Available: http://wiki.ros.org/costmap\_2d. [Accessed: 23- Mar- 2022].

[73] S. Topal, I. Erkmen and A. Erkmen, "A Novel Map Merging Methodology for Multi-Robot Systems", *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 2010. [Accessed 23 March 2022].

# **Appendices**

## **Appendix A**

Text

Description automatically generated with medium confidence

Figure 23. Map Pair No.1

Diagram

Description automatically generated

Figure 24. Map Pair No.2

A picture containing text

Description automatically generated

Figure 25. Map Pair No.3

Text

Description automatically generated with medium confidence

Figure 26. Map Pair No. 4

A picture containing text, antenna

Description automatically generated

Figure 27. Map Pair No.5

## **Appendix B**

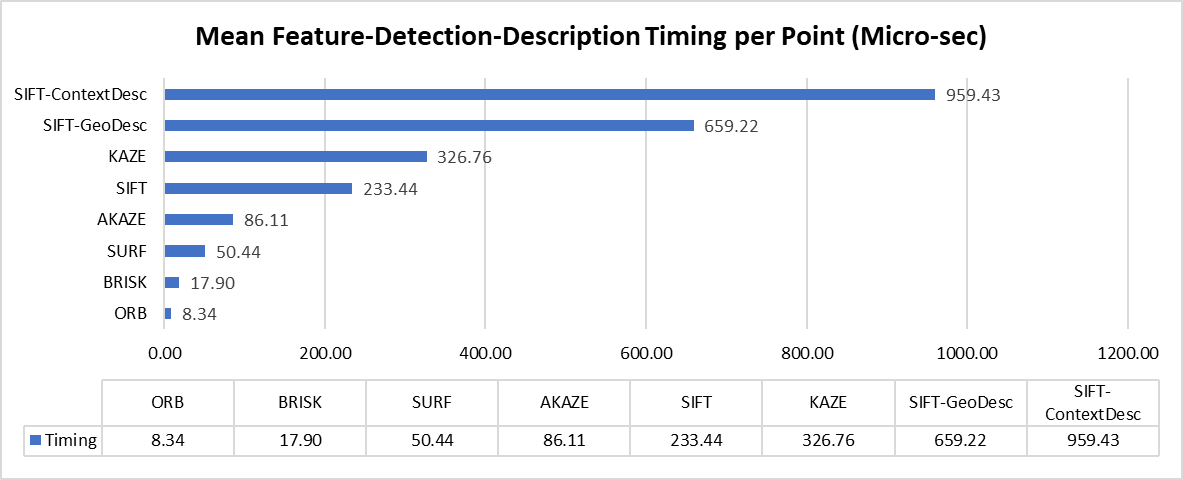
******

Figure 28. Pair 1 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).

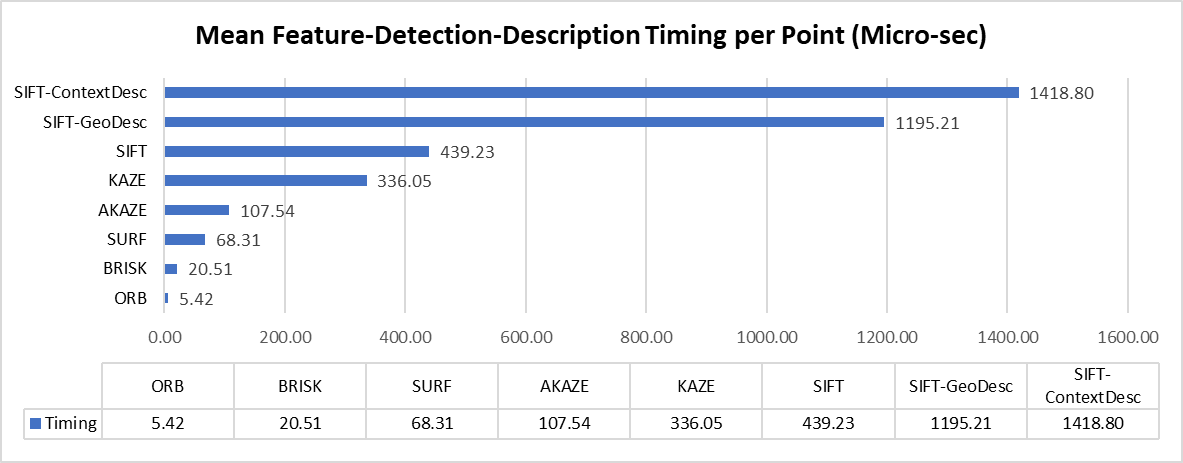
******

Figure 29. Pair 2 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).

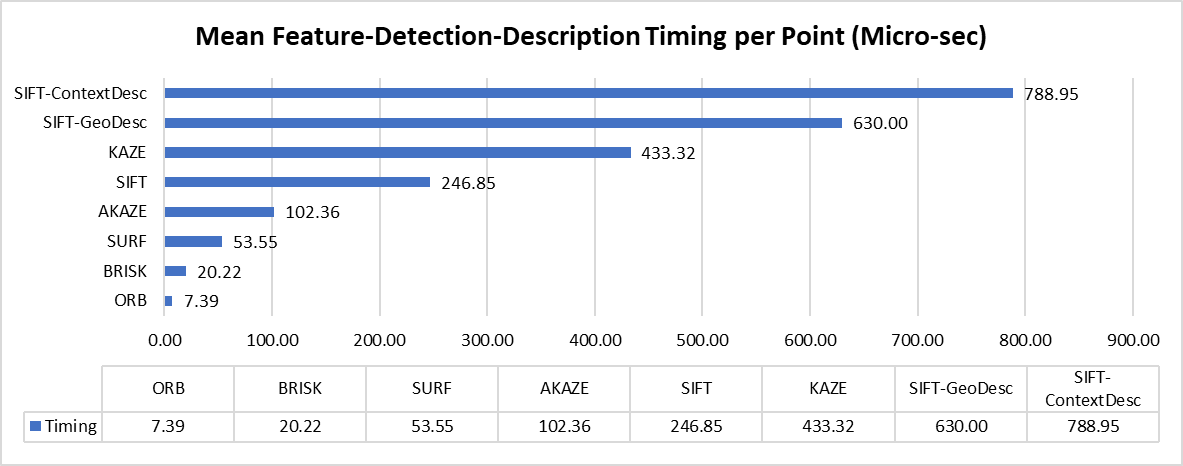
******

Figure 30. Pair 3 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).

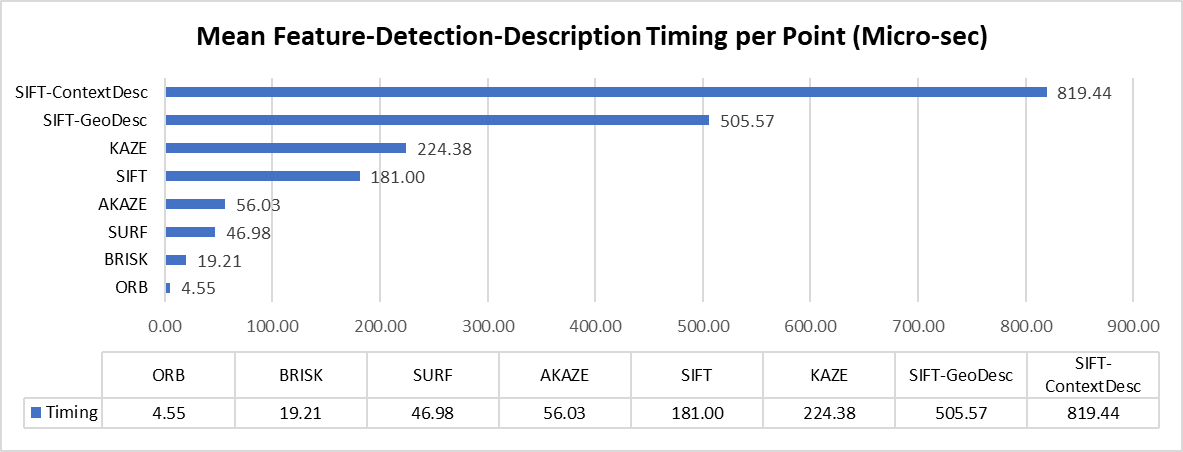
******

Figure 31. Pair 4 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).

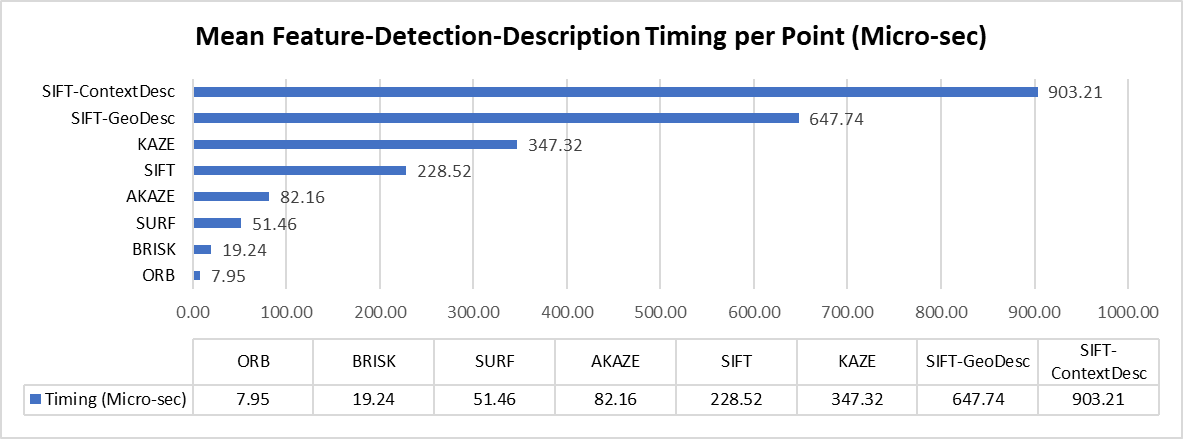
******

Figure 32. Pair 5 - Bar chart of mean feature-detection-description timing per feature point (Micro-sec).

## **Appendix C**

Chart, line chart

Description automatically generated

Figure 33. Pair 1 - Graph of Number of Inlier Matches against Rotation Applied.

***Chart, line chart

Description automatically generated***

Figure 34. Pair 1 - Graph of Number of Inlier Matches against Scale Applied.

***Chart, line chart

Description automatically generated***

Figure 35. Pair 2 - Graph of Number of Inlier Matches against Rotation Applied.

***Chart, line chart

Description automatically generated***

Figure 36. Pair 2 - Graph of Number of Inlier Matches against Scale Applied.

***Chart

Description automatically generated***

Figure 37. Pair 3 - Graph of Number of Inlier Matches against Rotation Applied.

***Chart, line chart

Description automatically generated***

Figure 38. Pair 3 - Graph of Number of Inlier Matches against Scale Applied

***Chart, line chart

Description automatically generated***

Figure 39. Pair 4 - Graph of Number of Inlier Matches against Rotation Applied.

***Chart, line chart

Description automatically generated***

Figure 40. Pair 4 - Graph of Number of Inlier Matches against Scale Applied.

***Chart, line chart

Description automatically generated***

Figure 41. Pair 5 - Graph of Number of Inlier Matches against Rotation Applied.

***Chart, line chart

Description automatically generated***

Figure 42. Pair 5 - Graph of Number of Inlier Matches against Scale Applied.

## **Appendix D**

Table 7. Inlier STDEV/Mean for Rotation and Scale Segments.



## **Appendix E**

Table 8. Success states for each algorithm for rotation test segment.



Table 9. Success states for each algorithm for scale test segment.



## **Appendix F**

Table 10. Table of number of matches, number of inlier matches, transformation confidence, and success state for each FDD in the merging of warehouse world.

