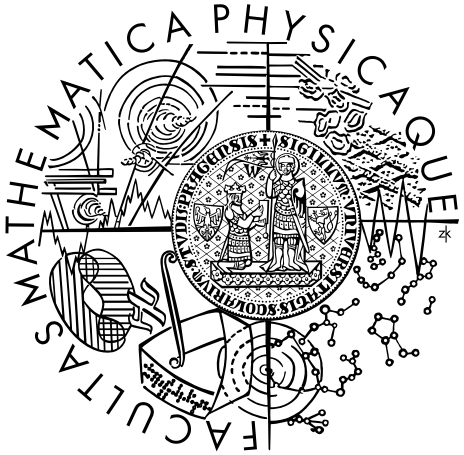
Charles University in Prague
Faculty of Mathematics and Physics

BACHELOR THESIS



Jiří Hörner

Map-merging for multi-robot system

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis: RNDr. David Obdržálek, Ph.D.

Study programme: Computer Science

Study branch: General Computer Science

Prague 2016

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: Map-merging for multi-robot system

Author: Jiří Hörner

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. David Obdržálek, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: A set of robots mapping an area can potentially combine their information to produce a distributed map more efficiently and reliably than a single robot alone. Multi-robot swarm coordination depends on a consistent, reliable map of the environment. Map-merging algorithms are therefore key komponents for such systems. In this work I present a novel algorithm for merging two-dimensinal maps created by different robots independently without initial knowledge of relative poses of robots. The algorithm is inspired by computer vision image stitching techniques for creating photo panoramas. Presented algorithm relies only on map data represented as occupancy grids, which allows great scalabity for heterogeneous multi-robot swarms and makes algorithm easily deployable with various simultaneous localization and mapping (SLAM) algorithms. The map-merging algorithm was implemented as publicily available Robot Operating System (ROS) package and was accepted in ROS distribution. Performance of the algorithm has been evaluated in ROS enviroment using Virtual Robot Experimentation Platform (VREP) simulator. For purposes of evaluation ROS package for exploring was developed as part of this work.

Keywords: map-merging multi-robot system ROS SLAM image stitching

I acknowledge my colleague Lukas Jelinek for sharing his insights into VREP simulator. I wish to thank to my family for support.

Contents

Introduction

Multi-robot exploring swarms have several advantages over a single robot. When properly coordinated performace of the multi-robot system is higher and multiple robots can possibly do tasks single robot could not. In fully distributed systems single point of failure is eliminated.

Multi-robot swarms can overcome imperfections in underlying navigation and mapping algorithms especially when using heterogeneous robot swarms, where one stucked robot can be replaced by another robot which uses different algorithm.

This work focuses on map-merging, which is a challenging problem, especially in heterogeneous robot swarms. In multi-robot systems shared map is required for effective coordination. Map-merging algorithm producing global map is therefore essential component of multi-robot systems.

This text is structured as follows: Key aspects of map-merging and related works are discussed in Section 1. In Section 2, I present a novel map-merging algorithm based on image stitching techniques, which can work with heterogeneous multi-robot swarms and is scalable to large number of robots.

Section 3 presents ready-to-use ROS packages implementing presented map-merging algorithm and frontier-based autonomous exploration. Section 4 discusses performace of presented map-merging algorithm achieved in several simulation experiments.

Documentation for ROS packages presented in Section 3 is attached as Appendices A, B.

1. Initial pose estimation problem

Key problem of map-merging is getting transformation between reference frames of robots. When the transformation is known, merging maps produced by robots is simple. In such a case we can compute transformation to chosen global reference frame and overlay maps in global reference frame. Errors in maps especially while mapping in dynamic environment may lead to different values across maps for specific global frame coordinates, but this can be solved by taking arithmetic mean, median or extremal values for such coordinates.

Transformation between grids can be acquired from initial poses of robots and vice versa, considering initial pose is represented as origin in the map (or generally any fixed point). This holds usually true for existing implementations of SLAM algorithms.

Problem arises when initial poses of robots are not known to merging algorithm. When robots are starting from the same place, initial relative positions can be measured with basic equipment either present on the site or mounted on robots. When robots are starting from distant locations measuring initial poses might require more sophisticated equipment, as widely available hardware such as Global Positioning System (GPS) sensors might not provide required precision. In indoor environments determining starting positions might be even more challenging.

Because of these difficulties, algorithms that can estimate transformations between maps and then merge maps without knowledge of initial positions have been developed. A comprehensive survey of map-merging techniques is done in [LLL+12], which classifies algorithms as Direct Map Merging and Indirect Map Merging.

1.1 Direct map merging

Direct map-merging algorithm relies on sensors to directly compute transformation between reference frame of robots. This includes techniques relying on direct robot rendezvous [ZR06] and similar. This techniques can provide a highly accurate transformation estimate, but are limited by relying on specific conditions to estimate this transformation such as the robot rendezvous or encountering specific landmark. These systems also exchange specific data related to selected feature or sensor to estimate transformation; some algorithms even require control over robot such as solution presented in [KFL+03]. Implementations therefore tend to be monolithic, because they rely on exchange of custom messages and specific sensors data, which makes them hard to implement especially for heterogeneous robots.

1.2 Indirect map merging

Indirect map-merging algorithms use overlapping areas in maps to estimate transformation between maps. Merging maps based on map data only naturally creates a common interface and works well for heterogeneous groups of robots. Each robot is only required to expose its map in a common format, sensor equipment mounted on robots may be different as well as a SLAM algorithm used for creating the map. This makes indirect map-merging algorithms more flexible than direct map merging: robots may visit overlapping area at different times (in contrast to robot rendezvous) and this approach is also absolutely passive requiring no control over robot.

Wide range of techniques has been employed for indirect map merging. In his work [LLL+12] mentions in this category techniques based on scan matching algorithms. These algorithms are working with SLAM representation of maps and they are usually tightly coupled with specific SLAM algorithm. This makes them share some disadvantages (such as difficult scalability through heterogeneous systems) with direct map merging algorithms. Algorithms in this category include [WJL+12] work based on visual scale-invariant feature transform (SIFT) features and topology nodes, approach of [TLKJ10] combining omnidirectional vision and laser scans, [CWBD12] using graph SLAM with condensed maps and non-linear constrained optimization to acquire transformation between maps. Data used by the SLAM algorithm to represent map may be too big to exchange all of them between robots, this was addressed in the work by [LPP+13], which uses condensed measurements and multi-robot graph SLAM.

Only a few algorithms works exclusively with portable map representation (maps represented as two-dimensional occupancy grids), despite this promises better scalability and by design supports heterogeneous multi-robot systems. Works using occupancy grids include spectra-based approach of [Car08]. [LL11] combines his approach with concept of virtual supporting lines to merge custom sparse maps of infrared features. [Mar13] used algorithm of [Sch12] based on image features to merge maps of 2 robots, which is limitation of this algorithm.

The novel algorithm for map merging presented in Section 2 is using only occupancy grids to produce the merged map. This algorithm is inspired by image stitching algorithms for creating photo panoramas. It is designed to merge maps from arbitrary number of robots, overcoming the limitation of both [Sch12] and [Car08]. It employs random sample consensus (RANSAC) for robust transform estimation and uses probability model to evaluate confidence of estimated transformation. Matching phase is accelerated using parallel hierarchical clustering trees proposed by [ML12] so that algorithm scales well to large number of robots.

2. Merging algorithm

In this chapter I present a map-merging algorithm for two-dimensional maps based on computer vision techniques. This approach is not completely new. First map-merging algorithm implemented for ROS based on image features is [Sch12]. Purpose of this package is stitching map created by SLAM to existing static map.

Although this package was not developed for map-merging in multi-robot configuration, algorithm and its original implementation were used by [Mar13] for merging maps of two robots and in coordinated multi-robot exploration solution presented in [ANB14].

Due to its original purpose, mapstitch algorithm shows some limitations for multi-robot map-merging setup. The algorithm was originally designed for offline use [ANB14]. Also, it was designed for stitching two maps, one them being large reference map covering most of the environment. Although it is possible to incrementally merge maps from multiple robots with this algorithm, global map quality generally decreases with increasing number of robots. Significant decrease in performance was observed for 4 robots [ANB14].

Algorithm 1 Mapstitch original algorithm. Implemented for ROS in [Sch12]

Input: 2 occupancy grids
Output: transformation between 2 grids

1: **procedure** StitchedMap($grid1, grid2$)
2: detect Oriented FAST and Rotated BRIEF (ORB) features
3: match keypoints with Brute-Force matcher
4: find matching point pairs with same distance in both images
5: estimate affine transformation
6: **end procedure**

Algorithm 1 is original algorithm used in [Sch12], version used in [ANB14] is only a slightly modified.

Although our algorithm also uses computer vision based approach, it deals with limitations of the Algorithm 1. Proposed algorithm is designed to work with arbitrary number of grids (while Algorithm 1 can only work with two grids). There is no need for iterative merging and moreover algorithm can determine optimal order of individual pairwise merges. I assume this might be the main reason for decrease in performance in 4-robot setup observed by [ANB14]. Also proposed algorithm deals with other problems arising for general n-map merge problem such as situations when it is not possible to merge some of the maps because transformation to others could not be reliably estimated, cases where map transformation can be estimated from more sources (multiple neighbours) etc.

2.1 Stitching pipeline

As discussed in Section 1.2 our algorithm is inspired by image stitching algorithms. Stitching algorithms are well-understood and implementations are broadly available. General concept of multi-step stitching pipeline is described

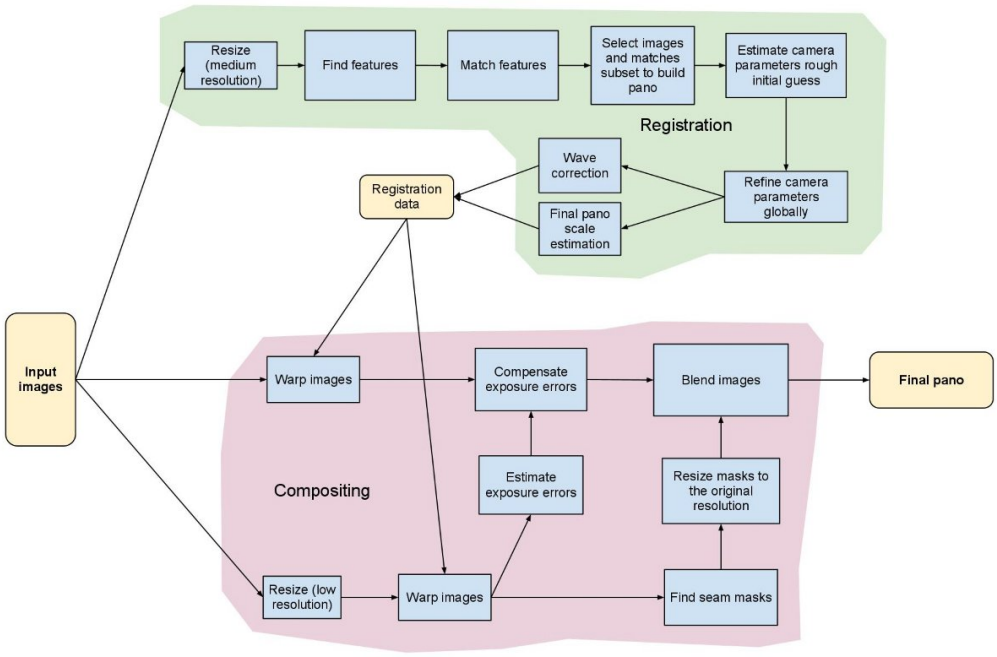


Figure 2.1: OpenCV Stitching pipeline.

in [BL06]. Stitching pipeline is also well established code in Open Source Computer Vision Library (OpenCV), mostly based on [BL06], along with [Sze04] [SS98] and others. Figure 2.1 highlights processing steps in stitching pipeline implemented in OpenCV.

Our algorithm will solve the registration part (estimating transformation between grids) according to Figure 2.1. Compositing part of stitching is relatively simple for occupancy grids compared to images from camera, because we don't need to compensate exposure errors, gain and other deficiencies. Registration solves the main problem of acquiring transformation between individual frames of robots and bridges the problem of merging maps with known initial positions and unknown initial positions.

ROS node for map merging described in Section 3.1 implements also compositing part of the pipeline, which is an easy problem when transformation is estimated.

For description of the algorithm we will assume maps are represented as occupancy grids, with each cell containing value in range $[0, 100]$ indicating probability that there is obstacle in the cell and -1 for indicating unknown probability. ROS uses the same representation. This representation can be mapped easily to greyscale image, hence using image processing algorithms is natural.

We will consider occupancy grids greyscale images through Algorithm 2 and vice versa. Values in the range $[0, 100]$ are the same, -1 is mapped to 255 in the image. This way we get standard 8-bit depth greyscale image.

Algorithm 2 offers overview of the proposed algorithm, detailed description is provided in following sections.

Algorithm 2 Proposed algorithm for estimating transformation between multiple occupancy grids. Uses Algorithm 3 to estimate final transformations.

Input: k occupancy grids

Output: for each grid: transformation between grid and global reference frame, or value indicating transformation could not be be estimated for current grid

procedure ESTIMATEGRIDTRANSFORM($grids$)
    detect ORB features (keypoints) for each grid
    for all (i, j) pair of grids **do** ▷ compute transform for each pair
        match features
        $n \leftarrow$ number of matches
        if $n \leq$ matches threshold **then**
            confidence $\leftarrow 0$
        else
            try find restricted affine transformation for features with RANSAC
            $\psi \leftarrow$ number of inliers in RANSAC
            if transformation found **then**
                confidence $\leftarrow \frac{\psi}{8+0.3 \cdot n}$
                $P_{(i,j)} \leftarrow$ restricted affine transformation
            else
                confidence $\leftarrow 0$
            end if
        end if
    end for
    matches $\leftarrow (i, j)$ for matches with confidence ≥ 1.0
    $g \leftarrow (grids, matches)$
    $h \leftarrow$ largest connected component in g
    $t \leftarrow$ maximum spanning tree in h
    ESTIMATEFINALTRANSFORM(t, $P_{(i,j)} \forall e \in$ edges of t) ▷ walk t and compute transformations to global reference frame. See Algorithm 3.
end procedure

2.2 Feature detection

Stitching pipeline proposed in [BL06] is using SIFT features. SIFT features have been used with success for stitching in many applications. Some of the recent approaches to stitching, improving traditional SIFT-based algorithm, are also building on top of SIFT features [XXL+15]. Use of SIFT features is limited by US patent [Low04].

I have decided to use ORB feature detector and descriptor, which was intrtoduced by [RRKB11]. ORB algorithm is patent-free, and available in OpenCV. Moreover ORB features has already been used with occupancy grid images [Sch12] and [ANB14].

Other alternatives for feature detection and feature description has not been tested yet. Performance of other detectors for map-merging and effect of choose of detector to overall merging performance remains to be evaluated. Some feature detectors and descriptors promising good performance are [AOV12], [AS11] and [CLSF10].

For image stitching, images are usually downscaled for further processing as seen is Figure 2.1. Feature extraction and feature matching on smaller images is considerably faster and overall accuracy is acceptable. I don't propose any such down scaling for occupancy grids. Occupancy grids acquired from mapping are usually smaller than multi-megapixel images from camera, so stitching time is reasonable even for full-scale grids. Also occupancy grids have usually much smaller number of features than photos making stitching harder and less accurate.

During online merging stitching can run with low frequency even if higher map update frequencies are required by simply using previously estimated transformation between grids. This further reduces cost of estimation over time. Transformation between grids is fixed in most cases (when SLAM algorithm works reasonably well), because transformation depends only on starting positions of robots. Therefore reusing previous transformations does not reduce map quality considerably.

In most scenarios estimated transformation change only during initial phase. After there is enough overlapping regions in the map, such that transformation can be estimated with high precision, transformation estimated with stitching algorithm remains stable over time. This property allows to run re-estimation with even lower frequencies if map quality in initial phase is not a problem.

2.3 Pairwise matching

Pairwise matching is the most resource demanding part of the algorithm. We do matching for all $\mathcal{O}(n^2)$ pairs of grids. For panorama images it is possible to push this down to $\mathcal{O}(n)$ matchings by expecting photos to be taken in ordered sequence. Then we can match only k neighbours (for small k) in image sequence, because only neighbours are expected to have overlapping area.

For occupancy grids in multi-robot mapping scenario it is impractical to assume any such ordering in initial poses of robots. It is not even possible in certain scenarios especially when robots are exploring given areas independently. Our algorithm therefore always match all $\mathcal{O}(n^2)$ pairs.

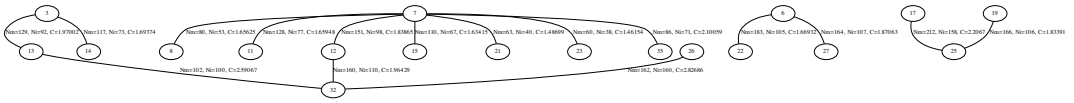


Figure 2.2: Graph showing matches between 36 occupancy grids during map merging. This graph was acquired for maps from MIT dataset described in Section 4.2. Grids without any matches are omitted. Legend: Nm number of matches, Ni number of inliers from RANSAC, C confidence.

Figure 2.2 shows matching results for 36 occupancy grids acquired during experiment presented in Section 4.6. Computations for each pair are independent, so it is easy to run matching of all pairs in parallel. This approach is used in map merging node presented in Section 3.1.

Because of non-linear number of pairs it is usually too computationally expensive to search for matches using simple brute-force search (even if it runs in parallel) unless it can be offloaded to GPU. For matching on CPU it is better to use approximate methods, which can be much faster.

For vector-based features, such as SIFT and Speeded Up Robust Features (SURF), the solution has been to use approximate nearest-neighbour search, but these existing algorithms are not suitable for binary features [ML12]. For ORB features, which are binary based, searching for nearest neighbours using parallel hierarchical clustering trees proposed in [ML12] can provide similar speed-up for ORB features. This method is used through the Fast Library for Approximate Nearest Neighbors (FLANN) by the same authors, which is now part of OpenCV.

When matching keypoints are found for pair of grids, algorithm estimates transformation between grids. Traditional image stitching algorithms are using homography in projective spaces, which is a good for modelling perspective affecting camera images. For occupancy grids this is not an expected transformation under normal circumstances, and even when there are errors in maps produced by SLAM algorithm, these are not errors produced by projective transformation.

For occupancy grids I propose a different model based on reduced affine transformation. This is a partial affine transformation with 4 degrees of freedom. This model extends the usual definition of map-merging problem as defined in [LLL+12] by allowing scaling. Scaling allows maps to have different resolutions, which may occur in heterogeneous multi-robot systems.

Definition 1 (Reduced affine transformation)**.** *For given matrices R (rotation), S (scaling), T (translation), where*

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{2.1}$$

$$S = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \tag{2.2}$$

$$T = \begin{pmatrix} tx \\ ty \end{pmatrix} \tag{2.3}$$

we define matrix of reduced affine transformation as

$$A = \big(RS|T\big) = \begin{pmatrix} \cos(\theta)s & -\sin(\theta)s & tx \\ \sin(\theta)s & \cos(\theta)s & ty \end{pmatrix} \tag{2.4}$$

As usual when representing translations we will work with reduced affine transformation in homogeneous coordinates, where this transformation is homomorphism. Therefore we extend A as

$$A' = \begin{pmatrix} \cos(\theta)s & -\sin(\theta)s & tx \\ \sin(\theta)s & \cos(\theta)s & ty \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

to represent reduced affine transformation in homogeneous coordinates space.

For pair of matched points $X = (x_1, x_2, 1)^\top$, $Y = (y_1, y_2, 1)^\top$ we can then solve

$$A'X = Y \tag{2.6}$$

$$\begin{pmatrix} \cos(\theta)s & -\sin(\theta)s & tx \\ \sin(\theta)s & \cos(\theta)s & ty \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} \tag{2.7}$$

for $(\cos(\theta)s, \sin(\theta)s, tx, ty)^\top$ to obtain transformation between grids. This system is easy to solve and we need only 2 points to get the transformation.

Reduced affine transformation is chosen to model transformation between initial robot poses in $2D$ space. Each robot can start at different position and have different orientation in terms of rotation in $2D$ space. Scaling enables occupancy grids to have different resolution.

We combine (2.6) with RANSAC [FB81] method to obtain final transformation. RANSAC is used to estimate homography in image stitching method proposed in [BL06]. We use the same method for robust estimation of reduced affine transformation. Using RANSAC has another advantage. We can use number of inliers from RANSAC to estimate transformation accuracy.

For each pair we compute transformation confidence as $\frac{\psi}{8+0.3\cdot n}$, where n is number of matches and ψ is number of inliers in RANSAC. Model for confidence is based on probabilistic model for image match verification proposed in [BL06].

For RANSAC I have chosen following parameters: maximum number of iterations 500, good ratio 0.5. Same parameters are used internally in OpenCV. If maximum number of iterations is reached and transformation therefore could not be found, its confidence is set to 0.

2.4 Finding largest connected component

As seen in Figure 2.2 it is common that some transformations can not be established between grids. Graph of matchings therefore can have multiple connected components. We need to deal with missing transformations between components to estimate transformations for grids.

We could include all components to resulting map and position them such they don't overlap. This setup would preserve all information, but resulting map

can't be topologically correct. Another approach is to choose only one connected component for final merge. This approach preserves map's topological accuracy. I have chosen the latter approach.

As a next step in the algorithm we filter out matches according to selected probabilistic model for accuracy. Matches with computed confidence less than or equal to 1.0 are not further considered.

After matches are filtered, largest component is found in the matches graph. Transformation will be established only for grids in largest connected component.

Choosing largest connected component might seem natural, but this approach has its caveats. Largest connected component represents matches between largest number of robots, however maps from the largest number of robots does not need to represent largest area in the map. This might be a problem especially in systems with heterogeneous robots, where mapping performance differs greatly between robots. Also some parts of maps might be much harder for robots to explore and despite large number of robots in such an area, produced map may be smaller than map produced by other group of robots (represented in graph as smaller component).

Modifications of this algorithm might choose to find weighted largest connected component in matchings graph. Weight could be based on discovered area in each map, such that the largest weighted connected component would represent largest discovered area.

I have chosen to use unweighted largest connected component, because it is less computationally expensive (weighting grids to represent discovered area in each grid require visiting each cell of each grid). Algorithm using unweighted largest connected component showed good results in tested scenarios, see Section 4.4. Approach using weighted components needs to be evaluated for more robots and larger environments.

2.5 Estimate final transformation

Remaining graph is connected and it is possible to estimate transformation for all grids. We have estimated transformations between all pairs of grids, but edges for some pairs in the connected component may be missing, because they were filtered out in previous steps.

Final part of the algorithm estimates transformation to global reference frame for each grid. We can choose reference frame of one of the grids as global reference frame, because we are interested in relative transformations between all grids for merging. This will be the reference frame of merged map.

Selecting global reference frame is not enough, because there may exist multiple paths from grid selected as reference frame to other grids. We construct maximum spanning tree to break these cycles. Edges are weighted with number of inliers to prefer stronger matches. This approach is routinely used for image stitching, the same construct is implemented in OpenCV.

Finally we can walk through spanning tree to obtain final transformations. There is now only one path from grid selected as reference frame to other grids. For each grid we can get the final by compositing pairwise transformations along the path. As we are working in homogeneous coordinates this is equivalent to

matrix product of pairwise transformations along the path. This can be done in linear time with Algorithm 3.

Algorithm 3 Algorithm estimating transformations to global reference frame from pairwise transformations on spanning tree.

Input: t maximum spanning tree on grids, $P_{(i,j)}$ pairwise reduced affine transformation in homogeneous coordinates between grids i, j.

Output: $T_i \forall i \in V$ transformations to global reference frame

1: **procedure** ESTIMATEFINALTRANSFORM($t = (V, E)$, $P_e \forall e \in e$)
2: $e \leftarrow$ edges of t sorted by discover time in breadth-first search (BFS) starting from grid with global reference frame ▷ using BFS or depth-first search (DFS) does not matter here
3: $\forall T_i : T_i \leftarrow I$ ▷ initialize transformations with identity
4: **for all** (i, j) in e **do** $T_j \leftarrow T_i P_{(i,j)}$
5: **end for**
6: **end procedure**

3. ROS packages

It this section I present ROS packages developed as part of this work. Merging algorithm presented in Section 2 was implemented in ROS package `multirobot_map_merge`. To evaluate performance of map-merging in multi-robot exploring scenarios I have developed the second package, `explore_lite` for autonomous exploring.

Both packages are now part of the ROS distribution. Documentation for packages is available online at the ROS wiki pages:

- `http://wiki.ros.org/multirobot_map_merge`
- `http://wiki.ros.org/explore_lite`

This documentation is also reproduced as Appendix A and Appendix B.

3.1 `multirobot_map_merge` package

`multirobot_map_merge` package solves several problems for merging maps from multiple robots. Dynamic robot discovery, initial poses estimation and map composition.

Dynamic robot discovery allows efficient easy-to-use auto-configuration of the package and also allows number of robots to change during exploring. This design allows robots to be launched and assigned to system based on exploration progress.

Initial poses estimation is the key feature of this package allowing merging maps for robots with unknown initial positions. For situations where robots initial positions are known (simulations) or can be measured with required precision (required equipment is available on robots or at the starting place) `multirobot_map_merge` package supports merging with user-provided initial robots positions. This was also used for producing a reference map to evaluate performance of estimation algorithm.

Regardless of how transformation between grids have been acquired (from user supplied initial poses or estimated by the algorithm) map composition is the final step to produce a merged map. This phase must be able to deal with different map sizes between robots, different map resolutions and be able to apply scaling, rotation and translation transformations.

3.1.1 Inter-robot communication

Running map-merging for multiple physical robots requires network connection between robots. Managing this connection is deliberately out-of-scope of this package. However, solutions exists in ROS to make this task relatively easy.

First of all ROS is designed to work natively across multiple computers. This setup requires almost no configuration and is supported by default. In this configuration one of the computers/robots runs a `roscore` service (also referred as ROS master), which acts as a directory listing (broker) service. All ROS topics and ROS services are available transparently through the whole network.

Main disadvantage of this setup is a single-point-of-failure `roscore` service. Although the communication in the ROS network is always peer-to-peer, `roscore` is required for advertisement, enumerating topics and establishing communication. This might not be acceptable for exploration robots communicating over unreliable link.

ROS community is aware that single ROS master is a limiting factor for many applications. There is a ROS Multi-master Special Interest Group (SIG) [Mul15] coordinating efforts for multi-master support.

As part of these efforts there exists the `multimaster_fkie` package, described in technical report [HH15], which allows setting a multi-master network transparently.

`multirobot_map_merge` can work transparently with both configurations as it is not tied to any particular communication between robots, allowing a great flexibility. It can take an advantage of native ROS communication in environments with reliable network link (such as a simulation running on a cluster) and use a user provided communication (such as the `multimaster_fkie` package) for exploring harsh environments with unrelible link. This is an important difference to framework presented in [ANB14], which always depends on custom ad-hoc messaging.

3.1.2 Dynamic robot discovery

This package allows merging maps from arbitrary number of robots. To make configuration of map-merging easy, robots are auto-discovered during the merging procedure. This also allows robots to be added or removed during exploring.

This package requires only maps produced by SLAM and does not depend on any additional info about robots. Robot discovery is implemented by scanning available ROS map topics, each map topic is being considered as one robot (one map to be added for merging). This approach is inspired by a discovery algorithm introduced in [YFLB14].

Robot discovery runs in parallel to map-merging at rate which is configurable. Robot discovery therefore does not negatively impacts map-merging performance, when configured at low rate.

3.1.3 Initial poses estimation

Initial poses estimation is necessary for situations where initial robot positions could not be measured with required precision by user. Package uses algorithm discussed in Section 2, which was specifically designed for this purpose. When robots are starting from different places, getting the initial positions might be difficult without proper equipment. Even when robots are starting exploration from common place, it might be more comfortable for users to let merging system estimate initial positions itself. In this situation merging algorithm can take the advantage of initial overlapping area and produce high-quality merges quickly.

Estimation is designed to run in parallel to other parts of this package. Estimation rate is user-settable.

3.1.4 Map composition

After estimating transformation between grids, map composition combines final merged map. For map-merging task I have adapted the relevant code from `occupancy_grid_utils` package [Mar14]. This package is no longer available in current ROS distribution, current version is maintained by Clearpath Robotics in github repository [Rob16]. I have contributed some fixes to this repository, but as the code of `occupancy_grid_utils` is mostly obsolete in current ROS distribution (most of the functionality is provided by ROS navigation stack), I have decided to incorporate merging-related code directly to `multirobot_map_merge` package.

The code is robust, it can deal with all differences in size and resolution to allow merging maps in heterogeneous systems. This code however does not provide expected performance for merging big maps from large number of robots. In future versions of `multirobot_map_merge`, this algorithm may be changed for more efficient one. Current solution works well for smaller groups of robots (up to 10 robots) on low end hardware.

Merging frequency is user-adjustable and can be increased to deliver faster update frequency.

3.2 `explore_lite` package

`explore_lite` package provides ROS node for autonomous exploration. Although there exists ROS packages for exploring [KMG+14], [DuH10], [Bov15] and exploring node from [ANB14], none of the existing packages met my requirements out-of-the-box. [KMG+14] is complex and includes custom navigation stack, [DuH10], [Bov15] are not available as of Apr 2016 for current version of ROS and [ANB14] offers some multi-robot coordination capabilities, but relies on custom ad-hoc communication between robots, which is an approach different from the map-merging node presented in Section 3.1.

For purposes of evaluation of the presented map-merging node, I have developed a new exploration node for ROS. This node is based on code of [DuH10] with major improvements. Main design goal of this node is light-weightiness. I have needed to allow running more robots in a testing environment with limited resources. Although the node does not have multi-robot coordination capabilities as this was not required for running selected simulation scenarios, this node must handle properly ROS namespaces and `tf` namespaces to allow running multiple robots under the same ROS master. As a result of these requirements major parts of the node have been redesigned.

3.2.1 Navigation

`explore_lite` uses ROS standard stack for navigation through the `move_base` node. The same approach is used by [DuH10], [Bov15] and [ANB14].

3.2.2 Map sourcing

Exploring packages, which use the ROS navigation stack [Bov15] and [DuH10] are using a local costmap provided by ROS navigation packages through `Costmap-`

2DROS. The local costmap is then used to search for frontiers and finding paths to frontiers from the robot's position.

Costmap2DROS is a feature-rich framework for building costmaps, allowing usage of plugins and several layers for costmaps. It can build costmaps from various sources including laser scans and handle inflating obstacles on-the-fly. Although all these features are great when building a costmap for robot navigation purposes it brings an unnecessary overhead when using the costmap for searching for frontiers and other exploration purposes.

For the explore_lite package I have introduced a custom costmap_client code, which subscribes to a map source in the ROS and provides a local costmap with only minimal processing. This reduces the overhead significantly compared to the situation when the costmap is build by ray-tracing from scans in explore node. Provided local costmap is then used for both frontier search and planing.

Choosing the right map source can also improve frontier discovery accuracy. Constantly better results has been achieved when the local costmap has been built from SLAM-constructed map, instead of a laser ray-traced costmap, such as the costmap created by move_base node.

3.2.3 Frontier search

Frontier search algorithm is based on the code in [DuH10]. Minor changes have been made to improve performance.

During frontier search frontiers are weighted, so that frontier with biggest weight could be forwarded to navigation node (move_base) as the next goal. To weight frontiers explore_lite needs to run at least basic path planning to get distance to frontier.

This planning is done through NavfnROS planner. During development it was apparent that this planner have several limitations, limiting its use to Costmap-2DROS as a source of costmap where planning happens. I have submitted a patch extending NavfnROS planner in the core ROS navigation stack. These changes have been accepted for the upcoming ROS Kinetic Kame release.

4. Evaluation

To evaluate map-merging algorithm introduced in Section 2 and to test performance of implemented map-merging node described in Section 3.1, I used 2 data sources. Data from simulation running several P3DX robots is the first source. Map-merging node was running through the whole exploring session testing online behaviour of the algorithm. Maps produced by SLAM on Massachusetts Institute of Technology (MIT) Stata Center dataset presented in [FJKL13] are the second data source. MIT dataset is produced by a single PR2 robot starting from different locations. Although this data does not come from multi-robot mapping, it is possible to test offline merging performance.

4.1 Simulation setup

I used VREP simulator for experiment. All simulated robots were Pioneers P3-DX, which formed a homogeneous exploring team. Robots were set up using `p3dx_robot` package available at [HJ16], which also configures SLAM and navigation for robots. Robots were using the `hector_slam` package [KMG+14] providing SLAM algorithm and `move_base` package [MELF16], part of the ROS navigation stack, providing navigation for robots.

Cluster of 5 computers was formed to run simulator, robots, map-merging and exploring nodes. ROS network was configured across all workstations using a single ROS master running `roscore`. Every robot was using its own ROS namespace for topics and was using a prefix for published `tf` frames to allow running multiple robots under the same ROS master. This setup is well supported in `p3dx_robot` package.

While VREP is powerful and feature-rich simulator, its usage for multi-robot simulation have some limitations. First of all, VREP support for headless mode (running without graphical environment) is not complete. Virtual framebuffer or similar technology is required, which adds performance overhead. Further VREP does not scale properly to large number of threads, limiting number of robots for which simulation runs at bearable speed. For this reason it wasn't possible to test more than 4 robots with this setup.

4.2 MIT dataset

MIT dataset is data available online in the form of rosbags [FJKL13]. Data was captured by a PR2 robot mapping multi-floor MIT building. I have used only datasets from the second floor.

For all rosbags I have created maps using the `hector_slam` package. It is the same SLAM algorithm, which was used in simulation. This resulted in 36 occupancy grid maps with sizes ranging from 2048×2048 cells to 5585×4895 cells.

Produced maps have been statically served in ROS. This setup is therefore limited to test offline merging, but allows a greater number of maps to participate in merging.

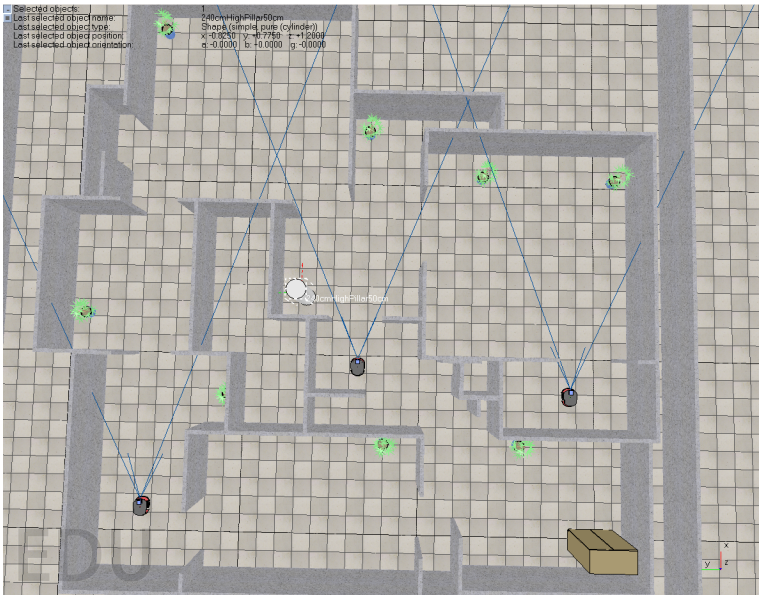


Figure 4.1: Scene for experiment with 3 robots in VREP simulator. Robot positions in scene are used as initial positions of robots for experiment.

It is important to note that presented maps has been created by a single robot in a multi-session mapping. It is not a result of multi-robot mapping, although the robot initial positions vary between sessions and produced maps are similar to maps we would expect from multi-robot mapping.

4.3 Merging with known initial positions

Presented merging node can work with both known and unknown initial positions of robots. In the fist case the node uses initial positions to obtain transformation between grids. This setup was used in this experiment. Maps in this mode are not required to have any overlapping area.

Simulation scene is shown in Figure 4.1. Figure 4.2 shows initial maps, Figure 4.3 shows maps after simulation ended. Note that Figures 4.2, 4.3 shows maps rotated compared to Figure 4.1.

Video capturing map-merging during whole simulation, rosbag with map topics, scene with 3 robots for VREP simulator and full quality graphics are attached.

4.4 Minimal overlapping area

Map-merging algorithm presented in Section 2 relies on overlapping areas of occupancy grids to produce a merged map. Minimal overlapping area to produce a reliable merge depends on environment being explored. Areas with high number of features in occupancy grids require small overlaps and vice versa.

Experiments showed that a reliable merge requires only about 90 inliers, sometimes only 80 inliers is enough to produce a correct transformation as seen in Figure 4.4. This excerpt is from the log of map-merging node presented in Section 3.1 acquired during simulation.

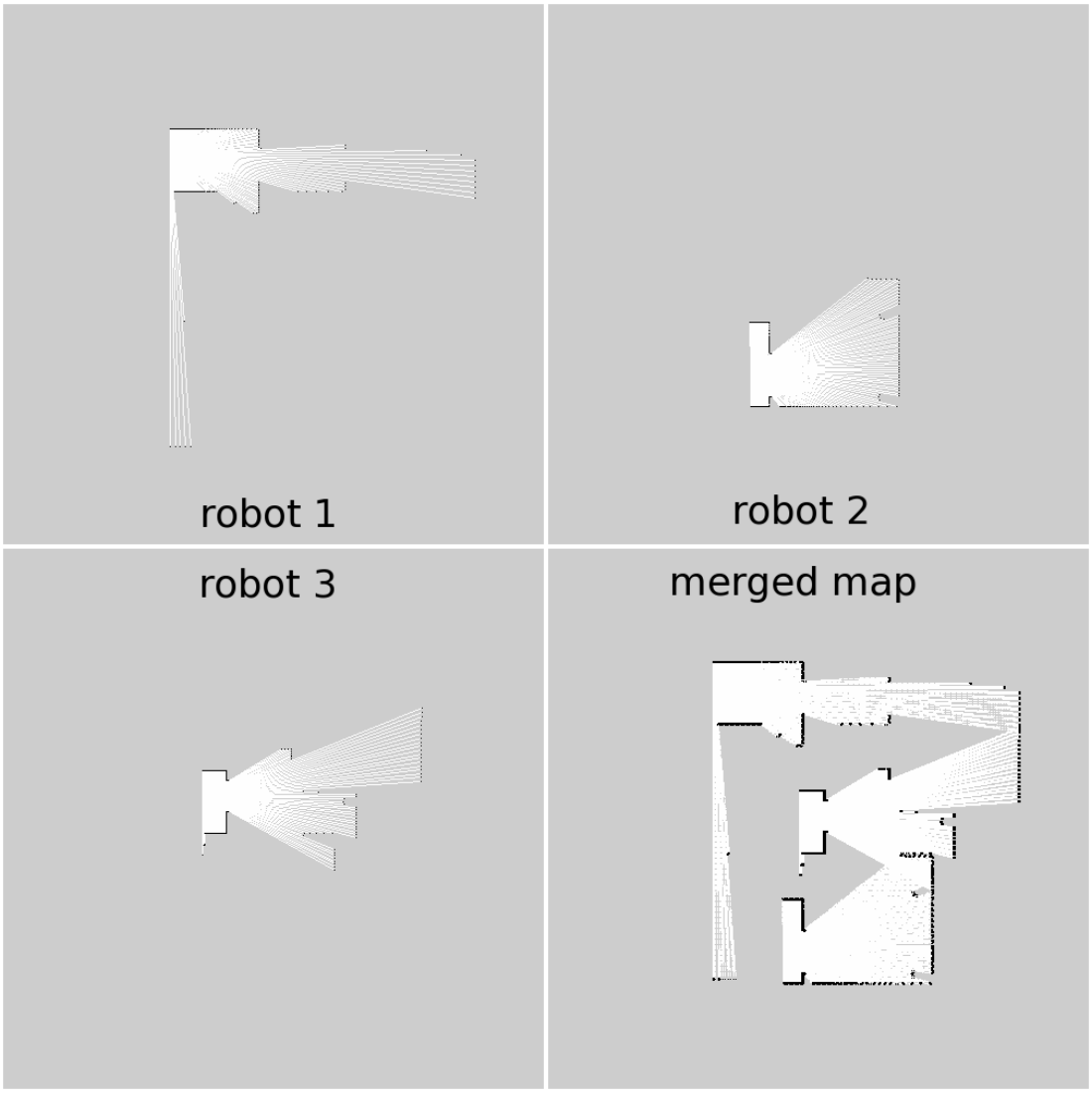


Figure 4.2: Initial maps produced by robots during experiment in the simulator. Merged map is produced with knowledge of initial positions and can be therefore produced even without overlapping areas. In this situation merged map can be sparse.

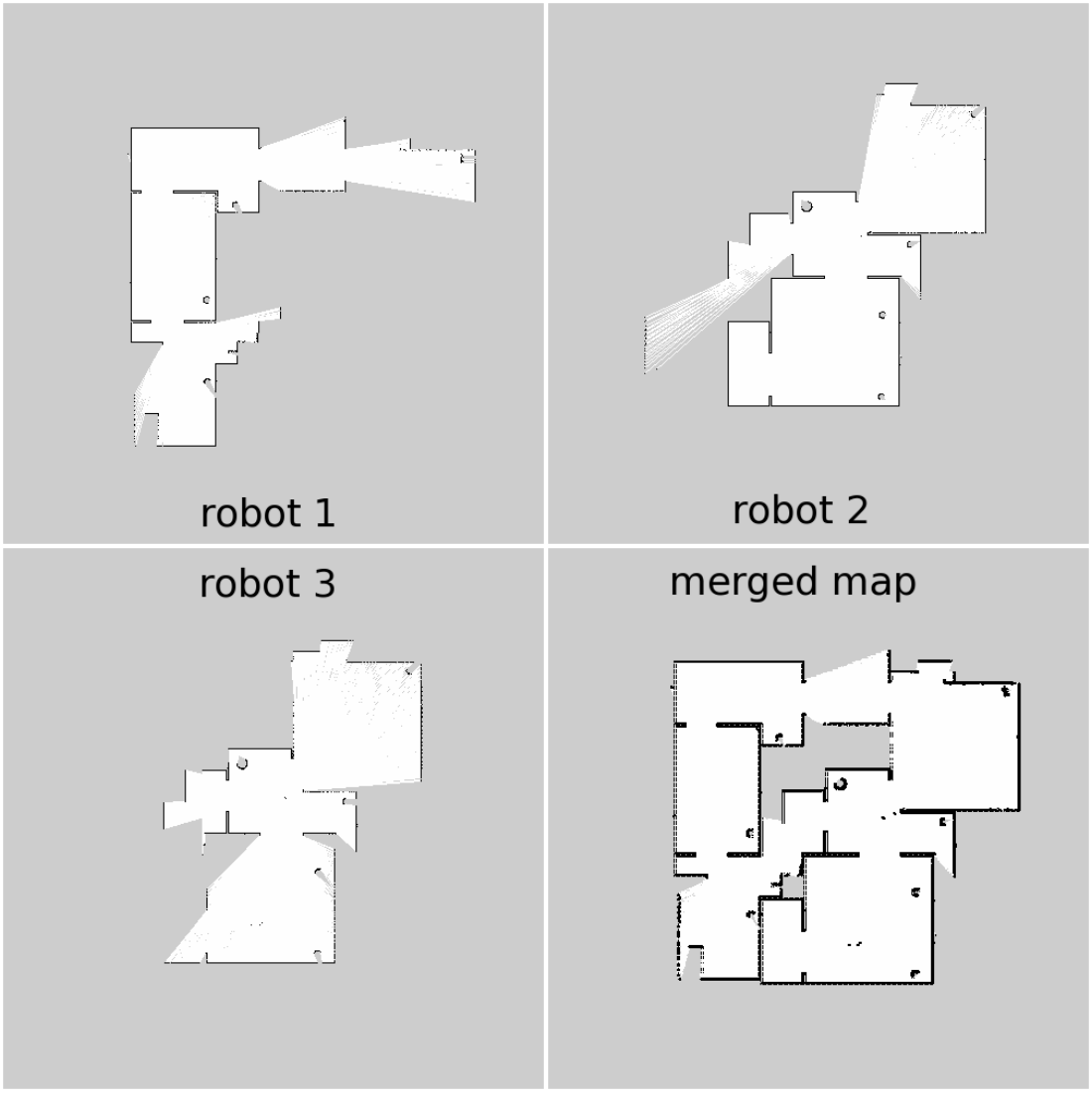


Figure 4.3: Final maps produced in the simulator. Merged map is produced with knowledge of initial positions.

```
AffineMatcher: have 121 matches
estimate:
[1.002576035215622, 0.00299917716022613, 62.49276756175958;
 -0.00299917716022613, 1.002576035215622, -240.2015971108993]
num_inliers 83
AffineMatcher: have 147 matches
estimate:
[1.002175802299877, -0.0004136975345276905, -15.26120294301828;
 0.0004136975345276905, 1.002175802299877, -120.7595895934327]
num_inliers 95
AffineMatcher: have 193 matches
estimate:
[1.000933706668138, 0.001232315845354937, 78.01218357952351;
 -0.001232315845354937, 1.000933706668138, -119.6792960984003]
num_inliers 157
```

Figure 4.4: Excerpt from the attached log of the map-merging node captured during simulation. Shows output of matching phase of the algorithm for 3 pairwise matches along with number of inliers.

Full log containing number of matches and number of inliers required to produce a transformation along with other details is available in the attachments. Scene for VREP simulator, merged map and maps produced by robots are also attached. Maps has been taken as screenshots in rviz visualiser.

Simulation featured 3 robots exploring common area. Figure 4.1 shows the scene used in the experiment and initial robots positions. Figure 4.5 shows maps produced by SLAM and the merged map produced by a map-merging node with unknown initial positions after mapping finished.

Simulation showed that a reliable merge between maps can be produced for $5 - 6$ overlapping rooms. After transformation is estimated, produced map is comparable to the reference map.

4.5 Retaining largest transformation

During online merging the merging algorithm presented in Section 2 is launched repeatedly on growing grids. It might seem natural to preserve the transformation between largest number of grids. If the algorithm was able to produce a merge between n grids in one point of time, it might seem like a good idea to preserve this transformation and use it for merging grids when the transformation is estimated for less than n grids.

Experiments showed that this approach produce worse results than approach always using the newest transformation. Largest transformation retaining behaviour is problematic when there is not enough overlapping area between grids. Experiment presented in Section 4.3 have maps that have very small overlaps.

Map-merging with unknown initial positions with largest transformation retaining launched in the same experiment as presented in Section 4.3 shows the key problem of a such approach. Problem occurs when incorrect merge is produced

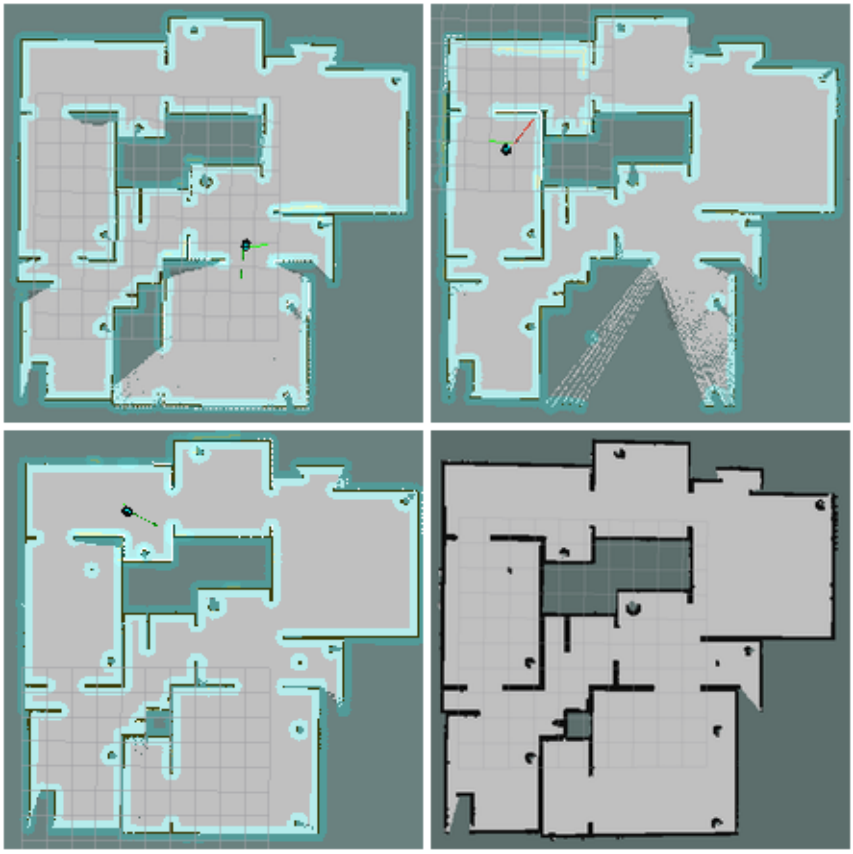


Figure 4.5: Maps produced by a multi-robot mapping in the simulator with 3 robots. The merged map (bottom right) is estimated by the map-merging node without knowledge of initial positions. Simulated scene can be seen in Figure 4.1. Positions of the robots are final positions where robots finished mapping.

Figure 4.6: Map-merging with largest transformation retaining. From the left: Maps have no overlaps, unable to merge more than a single map. Incorrect transformation estimated due to too small overlaps. Overlapping space is still too small to produce a merge, but incorrect transformation was rejected. Due to largest transformation retaining incorrect merge is still produced.

due to too small overlaps. As shown in Figure 4.6, incorrect unstable transformations, which are usually corrected quickly with small maps changes, are preserved due to largest transformation retaining after the estimation algorithm is no longer producing incorrect transformation. Incorrect transformations can be preserved for a long period of time with largest transformation retaining behaviour, when there is not enough overlapping area in the maps, so the incorrect transformation can be replaced with correct larger transformation.

Based on this and others experiments largest transformation retaining behaviour has been removed from map-merging node presented in Section 3.1.

Data for this experiment including rosbag with map topics, scene for VREP simulator and presented maps is available in the attachments.

4.6 Probability model evaluation

Data from MIT dataset is complex and difficult for SLAM to produce a map without errors. In some cases overlapping areas does not exists. Such data is therefore suitable to test capabilities of map merging node to reject incorrect or non-mergeable maps.

Figure 4.7 shows 36 maps obtained from MIT dataset, all of them are from second floor. Empty maps are caused by SLAM node failure. Note that maps contain many errors, some of them are completely broken as SLAM localization failed (this can't be prevented by the map-merging node). `hector_slam` node uses only scans from base-mounted laser, better results might be achieved with different SLAM approaches using stereo cameras and 3D scanning.

This dataset is very difficult to merge properly, because most of the maps are broken. It is possible to filtrate broken maps from merge, because the area where mapping failed should not match any other maps, but this is very difficult when most of the maps are broken. Furthermore broken maps tend to have more features than correct maps of the same area, these features than may cause invalid matches between two broken maps to be generated making rejecting broken maps even more difficult.

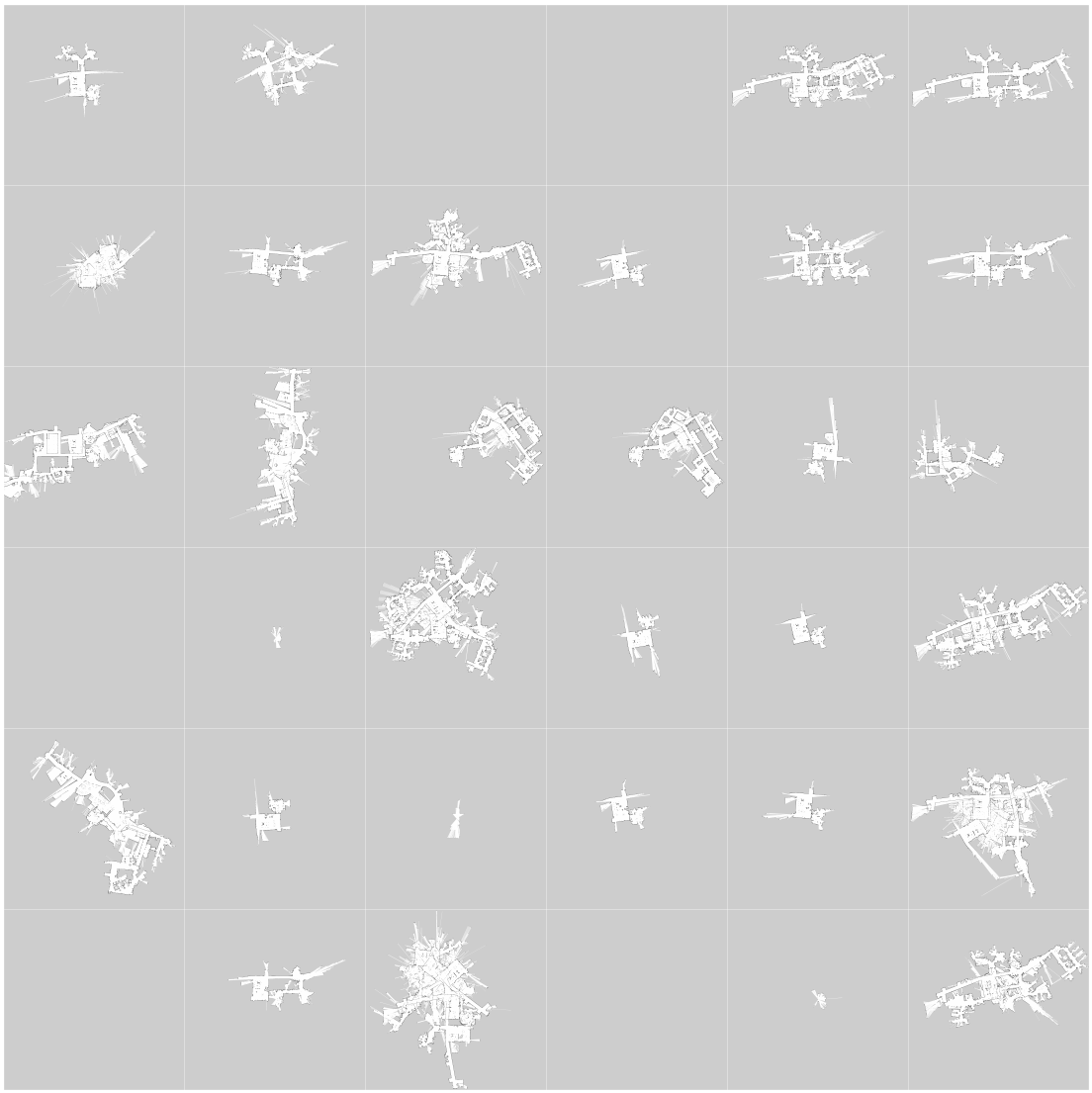


Figure 4.7: 36 maps created by `hector_slam` from MIT dataset. Note that the most of the maps contain serious mapping errors. These errors usually come from SLAM invalid estimation of rotation, generating misaligned walls and corners. Because of the new walls and corners, broken maps tend to have more features, especially in broken areas, making filtering of broken maps hard.

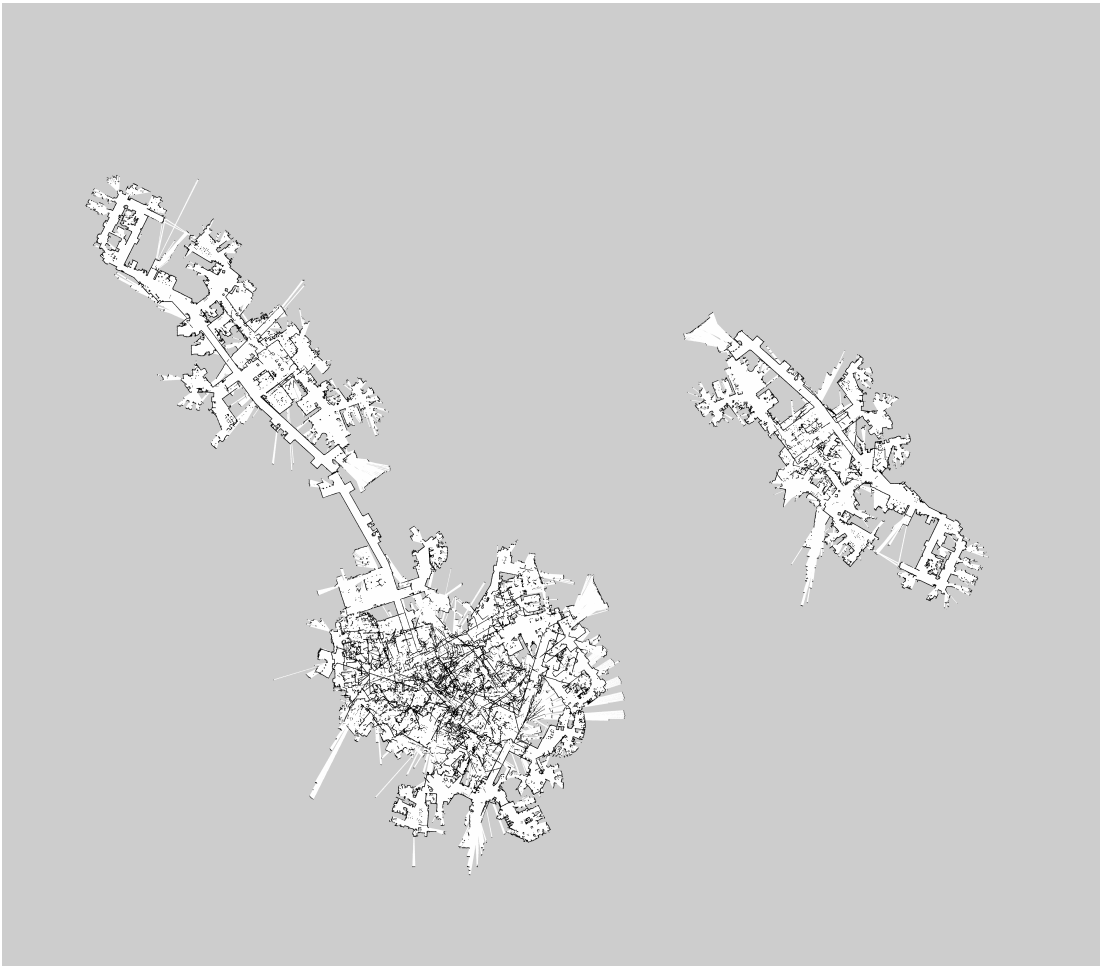


Figure 4.8: Merged map created from 36 maps shown in Figure 4.7. 12 maps are included in the merged map. Confidence threshold is set to 1.0. Although the map-merging algorithm was able to reject 24 maps, the merged map still includes severely broken maps. Broken areas are feature-rich and thus a wrong transformation is estimated.

Merging algorithm uses thresholding described in Section 2.4 based on probabilistic model to reject low-confident matches. I have experimented with confidence threshold, which directly affects which matches will be present in merging (matches with low confidence are not considered). Figures 4.8, 4.9 and 4.10 show maps merged with confidence thresholds 1.0, 1.5 and 2.0 respectively. Note that number of maps included in the merged map goes from 12 to 3.

Experiment has showed that increasing the confidence threshold to values greater that 1.0 may lead to better results when maps are difficult to merge. Increasing threshold decreases number of maps merged significantly.

Data used for this experiment are available in the attachments. This data consist of maps produced by SLAM on MIT dataset and merged maps for tested thresholds. Raw data of MIT dataset are available online [FJKL13].

Figure 4.9: Merged map created from 36 maps shown in Figure 4.7. 9 maps are included in the merged map. Confidence threshold is set to 1.5. This map contains even less maps than Figure 4.8, but threshold is not high enough to reject all broken maps. Severely broken maps are still included in the merged map. Broken areas in the maps were matched together. These areas are relatively feature-rich and matches between them has got enough RANSAC inliers to be considered confident enough.

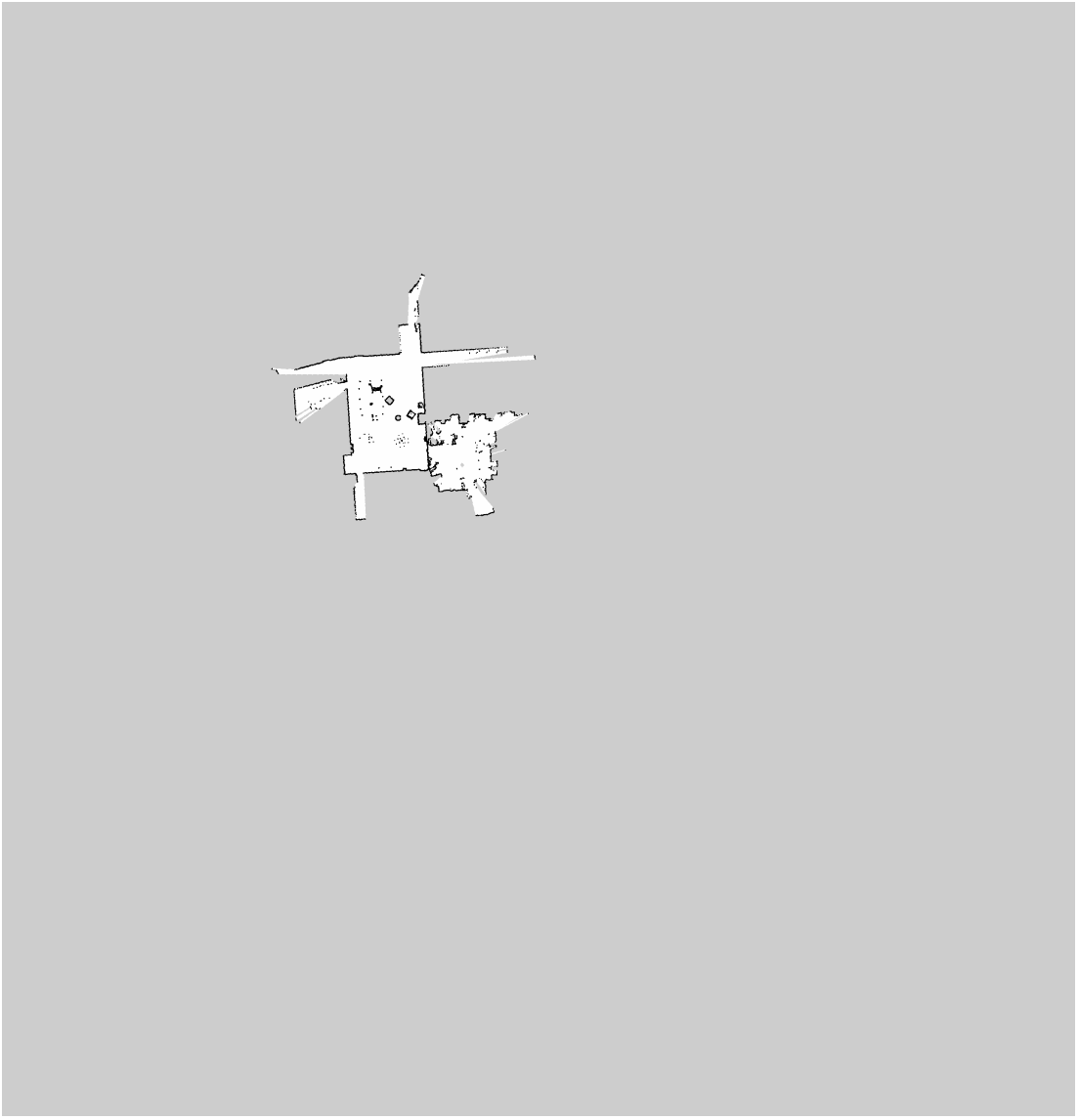


Figure 4.10: Merged map created from 36 maps shown in Figure 4.7. 3 maps are included in the merged map. Confidence threshold is set to 2.0. All broken maps have been rejected. The merged map is correct, transformation between grids was estimated correctly. Maps included in this merged map are small, for larger maps the SLAM algorithm mostly failed on presented dataset as seen in Figure 4.7.

5. Future works

This work has focused on map-merging of two-dimensional maps. Map-merging for tree-dimensional maps is a challenging problem in robotics. Despite two-dimensional maps are still routinely used for navigation in systems using tree-dimensional SLAM algorithms, tree-dimensional maps are becoming more common in robotics. While two-dimensional maps may provide a better overview for a human eye, three-dimensional maps contain more information and therefore might provide better accuracy for map-merging. A hybrid solution capable working with both two-dimensional and tree-dimensional maps might bring the best from both worlds together.

Although the merging algorithm has been partially evaluated on data captured by a PR2 robot, online properties of the merging algorithm has been evaluated only in the simulator. Evaluation with physical robots is yet to be done; building a multi-robot exploring system is purposely out of scope of this work. However I expect performance for real-world applications to be comparable to the experiments in the simulator or even better, because indoor physical environments are usually more feature-rich than scenarios in the simulator.

Conclusion

Map-merging algorithm presented in Section 2 can efficiently work with arbitrary number of robots. It scales well to large multi-robot systems and is designed with parallel processing in mind. The algorithm is suitable for heterogeneous multi-robot swarms and is easily deployable with various SLAM algorithms.

Proposed algorithm is based on image processing techniques and uses OpenCV through implementation. I have proposed a project implementing presented affine transformation estimation for OpenCV stitching pipe. This project have been accepted for Google Summer of Code 2016.

The algorithm is implemented in `multirobot_map_merge` ROS package presented in Section 3.1. This implementation is flexible, imposes low requirements on participating robots, does not depend on any particular communication between robots and does not have any presumptions on underlying mechanisms of SLAM algorithms used by robots. Those properties allows easy deployment in ROS environment for both existing systems and systems build from scratch. Performance of the implemented initial pose estimation algorithm is sufficient to merge maps from large number of robots (more than 30) on a laptop-grade processor. For large number of robots map composing is the bottleneck for map-merging node, this will be addressed in the next version of the map-merging node.

For purposes of evaluation I have created `explore_lite` package presented in Section 3.2 for autonomous exploring. Both packages have been accepted by the ROS project and are included in the ROS binary distribution.

During development I have make contributions to the ROS project including changes to the ROS navigation stack and ROS wiki. Changes for the ROS navigation stack have been accepted and will be included in the upcoming ROS release, changes to the ROS wiki (support for HTML5 videos) are already online.

The map-merging algorithm has been evaluated in the simulation and showed reliable estimates for maps with enough overlapping area. Necessary overlaps range from 5 to 7 rooms in a feature-poor simulation. Behaviour for online merging has been studied and implementation has been adapted to produce a consistent map when there is not enough overlapping space.

Bibliography

[ANB14] T. Andre, D. Neuhold, and C. Bettstetter. Coordinated multi-robot exploration: Out of the box packages for ROS. In *IEEE GLOBECOM WiUAV Workshop*, December 2014.

[AOV12] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517, June 2012.

[AS11] Pablo F. Alcantarilla and TrueVision Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2011.

[BL06] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2006.

[Bov15] Paul Bovbel. frontier_exploration. `http://wiki.ros.org/frontier_exploration`, 2015. [Online; accessed 2016-05-20].

[Car08] Stefano Carpin. Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25(3):305–316, 2008.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792, 2010.

[CWBD12] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert. Fully distributed scalable smoothing and mapping with robust multi-robot data association. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1093–1100, May 2012.

[DuH10] Charles DuHadway. explore. `http://wiki.ros.org/explore`, 2010. [Online; accessed 2016-05-20].

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FJKL13] Maurice Fallon, Hordur Johannsson, Michael Kaess, and John J Leonard. The mit stata center dataset. *The International Journal of Robotics Research*, 32(14):1695–1699, 2013.

[HH15] S. Hernández and F. Herrero. Multi-master ROS systems. Technical Report IRI-TR-15-01, Institut de Robòtica i Informàtica Industrial, CSIC-UPC, 2015.

[HJ16] Jiri Horner and Lukas Jelinek. p3dx_robot. `https://github.com/hrnr/robo-rescue`, 2016. [Online; accessed 2016-05-01].

[KFL+03] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart. Map merging for distributed robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 212–217, October 2003.

[KMG+14] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, and Oskar von Stryk. *RoboCup 2013: Robot World Cup XVII*, chapter Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots, pages 624–631. Springer Berlin Heidelberg, 2014.

[LL11] Heon-Cheol Lee and Beom-Hee Lee. Improved feature map merging using virtual supporting lines for multi-robot systems. *Advanced Robotics*, 25(13–14):1675–1696, 2011.

[LLL+12] H. C. Lee, Seung-Hwan Lee, Tae-Seok Lee, Doo-Jin Kim, and B. H. Lee. A survey of map merging techniques for cooperative-slam. In *9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 285–287, November 2012.

[Low04] D.G. Lowe. Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image. `http://www.google.com/patents/US6711293`, March 2004. US Patent 6,711,293.

[LPP+13] M. T. Lázaro, L. M. Paz, P. Piniés, J. A. Castellanos, and G. Grisetti. Multi-robot slam using condensed measurements. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1069–1076, November 2013.

[Mar13] J.A.S. Martins. *MRSLAM – Multi-Robot Simultaneous Localization and Mapping*. Master of Science Dissertation. University of Coimbra, 2013.

[Mar14] Bhaskara Marthi. occupancy_grid_utils. `http://wiki.ros.org/occupancy_grid_utils`, 2014. [Online; accessed 2016-05-20].

[MELF16] Eitan Marder-Eppstein, David V. Lu, and Michael Ferguson. move_base. `http://wiki.ros.org/move_base`, 2016. [Online; accessed 2016-05-20].

[ML12] Marius Muja and David G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV)*, pages 404–410, 2012.

[Mul15] Multimaster special interest group. `http://wiki.ros.org/sig/Multimaster`, 2015. [Online; accessed 2016-05-20].

[Rob16] Clearpath Robotics. occupancy_grid_utils. `https://github.com/clearpathrobotics/occupancy_grid_utils`, 2016. [Online; accessed 2016-05-20].

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, November 2011.

[Sch12] Philipp M. Scholl. mapstitch. `http://wiki.ros.org/mapstitch`, 2012. [Online; accessed 2016-05-20].

[SS98] Heung-Yeung Shum and Richard Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *Sixth International Conference on Computer Vision (ICCV'98)*, pages 953–958, Bombay, January 1998. IEEE Computer Society.

[Sze04] Richard Szeliski. Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, October 2004.

[TLKJ10] F. Tungadi, W. L. D. Lui, L. Kleeman, and R. Jarvis. Robust online map merging system using laser scan matching and omnidirectional vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7–14, October 2010.

[WJL+12] K. Wang, S. Jia, Y. Li, X. Li, and B. Guo. Research on map merging for multi-robotic system based on rtm. In *International Conference on Information and Automation (ICIA)*, pages 156–161, June 2012.

[XXL+15] Xin Xie, Yin Xu, Qing Liu, Fengping Hu, Tijian Cai, Nan Jiang, and Huandong Xiong. A study on fast sift image mosaic algorithm based on compressed sensing and wavelet transform. *Journal of Ambient Intelligence and Humanized Computing*, 6(6):835–843, 2015.

[YFLB14] Zhi Yan, Luc Fabresse, Jannik Laval, and Noury Bouraqadi. Team size optimization for multi-robot exploration. In *Simulation, Modeling, and Programming for Autonomous Robots: 4th International Conference, SIMPAR 2014, Bergamo, Italy*, pages 438–449. Springer International Publishing, October 2014.

[ZR06] X. S. Zhou and S. I. Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1785–1792, October 2006.

List of Figures

List of Abbreviations

BFS breadth-first search. 12

DFS depth-first search. 12

FLANN Fast Library for Approximate Nearest Neighbors. 9

GPS Global Positioning System. 3

MIT Massachusetts Institute of Technology. 17, 23–25

OpenCV Open Source Computer Vision Library. 6, 8–11, 29

ORB Oriented FAST and Rotated BRIEF. 5, 7–9

RANSAC random sample consensus. 4, 7, 9, 10, 26

ROS Robot Operating System. ii, 2, 5, 6, 13–17, 29, 33, 34, 39

SIFT scale-invariant feature transform. 4, 8, 9

SIG Special Interest Group. 14

SLAM simultaneous localization and mapping. ii, 3–5, 8, 9, 14, 16, 17, 21, 23–25, 27–29

SURF Speeded Up Robust Features. 9

VREP Virtual Robot Experimentation Platform. ii, 17, 18, 21, 23

List of Attached Files

This is a list of files attached to this work. Source code is also available online, see Appendices A B.

- attachements.zip
  - merging-with-known-initial-positions ..... experiment data, see 4.3
  - minimal-overlapping-area ..... experiment data, see 4.4
  - retaining-largest-transformation ..... experiment data, see 4.5
  - probability-model-evaluation ..... experiment data, see 4.6
  - m-explore ..... source code for ROS packages, see 3
    - explore ..... see 3.2
      - doc ..... package and code documentation
      - include
        - explore
          - explore.h ..... ROS node
          - navfn_ros.h ..... ROS planner with extended API, see 3.2.3
          - explore_frontier.h ..... see 3.2.3
          - costmap_client.h ..... see 3.2.2
      - launch
      - src
    - map_merge ..... see 3.1
      - doc ..... package and code documentation
      - test ..... gtest based tests
      - include
        - combine_grids ..... implements algorithm presented in Section 2
        - occupancy_grid_utils ..... see 3.1.4
        - map_merge
          - map_merge.h ..... ROS node
      - launch
      - src

Appendices

A. multirobot_map_merge

A.1 Package Summary

Merging multiple maps with knowledge of the initial relative positions of robots.

- Maintainer status: developed
- Maintainer: Jiri Horner <laeqten AT gmail DOT com>
- Author: Jiri Horner <laeqten AT gmail DOT com>
- License: BSD
- Source: git `https://github.com/hrnr/m-explore.git` (branch: master)

A.2 Overview

This package provides global map for multiple robots. It can merge maps from arbitrary number of robots. It expects maps from individual robots as ROS topics. If your run multiple robots under the same ROS master then `multirobot_map_merge` will probably work for you out-of-the-box. It is also very easy to setup an simulation experiment.

If your run your robots under multiple ROS masters you need to run your own way of communication between robots and provide maps from robots on local topics (under the same master). Also if you want to distribute merged map back to robots your communication must take care of it.

`multirobot_map_merge` does not depend on any particular communication between robots.

A.3 Architecture

`multirobot_map_merge` finds robot maps dynamically and new robots can be added to system at any time.

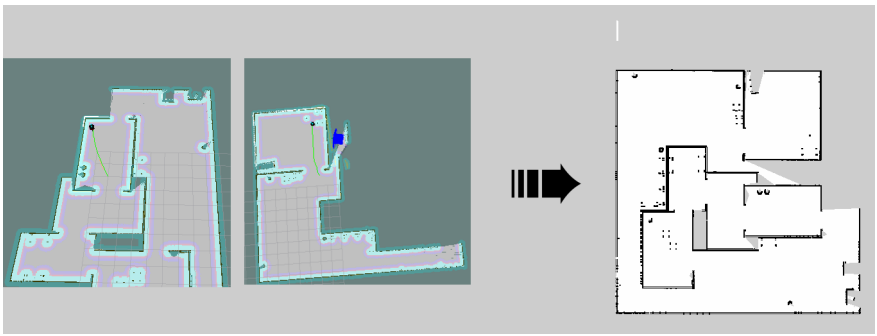


Figure A.1: Output of `multirobot_map_merge`, map merging node for ROS. The merged map for 2 robots. Robots in the environment visualised on the left, merged map on the right.

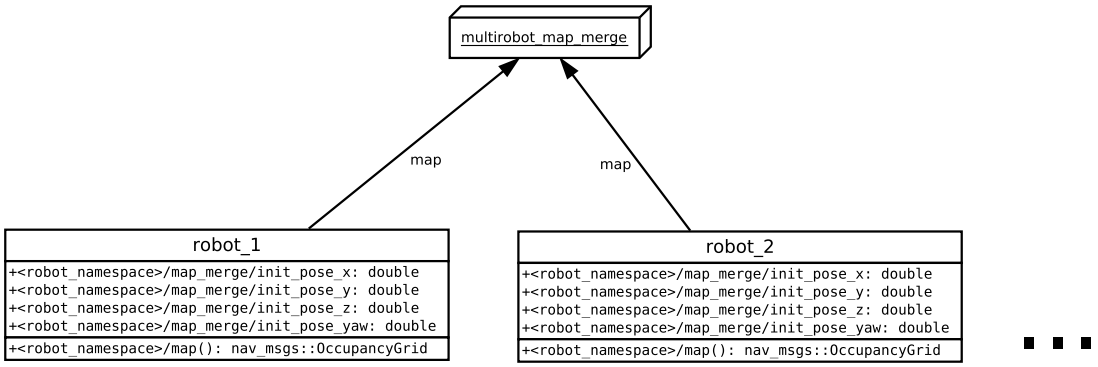


Figure A.2: Architecture of `multirobot_map_merge`, proposed map merging node for ROS.

To make this dynamic behaviour possible there are some constrains placed on robots. First all robots must publish map under `<robot_namespace>/map`, where topic name (`map`) is configurable, but must be same for all robots. For each robot `<robot_namespace>` will be of cause different.

This node support merging maps with known initial positions of the robots or without. See below for details.

A.4 Merging modes

Two merging modes are currently supported as orthogonal options. If you know initial positions of robots you may preferably use the first mode and get exact results (rigid transformation will be computed according to initial positions). If you don't know robot's starting points you are still able to use the second mode where transformation between grids will be determined using heuristic algorithm. You can choose between these two modes using the `known_init_poses` parameter.

A.4.1 merging with known initial positions

This is preferred mode whenever you are able to determine exact starting point for each robot. You need to provide initial position for each robot. You need to provide set of `<robot_namespace>/map_merge/init_pose` parameters. These positions should be in `world_frame`. See Section A.5.

In this merging these parameters are mandatory. If any of the required parameters is missing robot won't be considered for merging (you will get warning with name of affected robot).

A.4.2 merging without known initial positions

If you can't provide initial poses for robots this mode has minimal configuration requirements. You need to provide only map topic for each robot. Transformation between grids is estimated by feature-matching algorithm and therefore requires grids to have sufficient amount of overlapping space to make a high-probability match. If grids don't have enough overlapping space to make a solid match, merged map can differ greatly from physical situation.

Estimating transforms between grids is cpu-intesive so you might want to tune `estimation_rate` parameter to run re-estimation less often if it causes any troubles.

A.5 ROS API

A.5.1 map_merge

Provides map merging services offered by this package. Dynamically looks for new robots in the system and merges their maps.

Subscribed Topics

`<robot_namespace>/map` (nav_msgs/OccupancyGrid)
Local map for specific robot.

`<robot_namespace>/map_updates` (map_msgs/OccupancyGridUpdate)
Local map updates for specific robot. Most of the `nav_msgs/OccupancyGrid` sources (mapping algorithms) provides incremental map updates via this topic so they don't need to send always full map. This topic is optional. If your mapping algorithm does not provide this topic it is safe to ignore this topic. However if your mapping algorithm does provide this topic, it is preferable to subscribe to this topic. Otherwise map updates will be slow as all partial updates will be missed and map will be able to update only on full map updates.

Published Topics

`map` (nav_msgs/OccupancyGrid)
Merged map from all robots in the system.

Parameters

Robot Parameters Parameters that should be defined in the namespace of each robot if you want to use merging with known initial poses of robots (`known_init_poses` is `true`). Without these parameters robots won't be considered for merging. If you can't provide these parameters use merging without known initial poses. See Section A.4

`<robot_namespace>/map_merge/init_pose_x` (double, default: `<no_default>`)
`x` coordinate of robot initial position in `world_frame`. Should be in meters. It does not matter which frame you will consider global (preferably it should be different from all robots frames), but relative positions of robots in this frame must be correct.

`<robot_namespace>/map_merge/init_pose_y` (double, default: `<no_default>`)
`y` coordinate of robot initial position in `world_frame`.

`<robot_namespace>/map_merge/init_pose_z` (double, default: `<no_default>`)
`z` coordinate of robot initial position in `world_frame`.

`<robot_namespace>/map_merge/init_pose_yaw` (double, default: `<no_default>`)

`yaw` component of robot initial position in `world_frame`. Represents robot rotation in radians.

Node Parameters Parameters that should be defined in the namespace of this node.

~`robot_map_topic` (string, default: `map`)
Name of robot map topic without namespaces (last component of topic name). Only topics with this name will be considered when looking for new maps to merge. This topics may be subject to further filtering (see below).

~`robot_map_updates_topic` (string, default: `map_updates`)
Name of robot map updates topic of `map_msgs/OccupancyGridUpdate` without namespaces (last component of topic name). This topic will be always subscribed in the same namespace as `robot_map_topic`. You'll likely need to change this only when you changed `robot_map_topic`. These topics are never considered when searching for new robots.

~`robot_namespace` (string, default: `<empty string>`)
Fixed part of robot map topic. You can employ this parameter to further limit which topics will be considered during dynamic lookup for robots. Only topics which contain (anywhere) this string will be considered for lookup. Unlike `robot_map_topic` you are not limited by namespace logic. Topics will be filtered using text-based search. Therefore `robot_namespace` does not need to be ROS namespace, but can contain slashes etc. This must be common part of all robots map topics name (all robots for which you want to merge map).

~`known_init_poses` (bool, default: `true`)
Selects between merging modes. `true` if merging with known initial positions. See Section A.4

~`merged_map_topic` (string, default: `map`)
Topic name where merged map will be published.

~`world_frame` (string, default: `world`)
Frame id (in tf tree) which will be assigned to published merged map. This should be frame where you specified robot initial positions.

~`merging_rate` (double, default: `4.0`)
Rate in Hz. Basic frequency on which this node discovers merges robots maps and publish merged map. Increase this value if you want faster updates.

~`discovery_rate` (double, default: `0.05`)
Rate in Hz. Frequency on which this node discovers new robots. Increase this value if you need more agile behaviour when adding new robots. Robots will be discovered sooner.

~`estimation_rate` (double, default: `0.5`)
Rate in Hz. This parameter is relevant only when merging without known positions, see Section A.4. Frequency on which this node re-estimates transformation between grids. Estimation is cpu-intensive, so you may wish to lower this value.

~`estimation_confidence` (double, default: `1.0`)
Relevant only when merging without known positions, see Section A.4. Confidence according to probabilistic model for initial positions estimation. Default value 1.0 is suitable for most applications, increase this value for more confi-

dent estimations. Number of maps included in the merge may decrease with increasing confidence. Generally larger overlaps between maps will be required for map to be included in merge. Good range for tuning is $[1.0, 2.0]$.

A.6 Acknowledgements

This package was developed as part of my bachelor thesis at Charles University in Prague.

Idea for dynamic robot discovery is from map_merging package from Zhi Yan. Merging algorithm and configuration are different.

B. explore_lite

B.1 Package Summary

Lightweight frontier-based exploration.

- Maintainer status: developed
- Maintainer: Jiri Horner <laeqten AT gmail DOT com>
- Author: Jiri Horner <laeqten AT gmail DOT com>
- License: BSD
- Source: git `https://github.com/hrnr/m-explore.git` (branch: master)

B.2 Overview

This package provides greedy frontier-based exploration. When node is running, robot will greedily explore its enviroment until no frontiers could be found. Movement commands will be send to move_base.

Unlike similar packages, `explore_lite` does not create it's own costmap. Node subscribes to `nav_msgs/OccupancyGrid` messages. Commands for robot movement are send to move_base node.

B.3 Architecture

`explore_lite` uses move_base for navigation. You need to run properly configured move_base node.

`explore_lite` subscribes to a `nav_msgs/OccupancyGrid` and `map_msgs/OccupancyGridUpdate` messages to construct a map where it looks for frontiers. You can either use costmap published by move_base (ie. `<move_base>/global_costmap/costmap`) or you can use map constructed by mapping algorithm (SLAM). Better results were achieved on maps constructed by SLAM as they usually contain less noise.

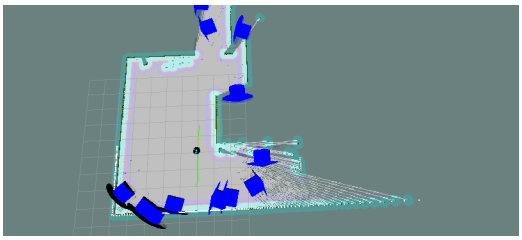


Figure B.1: Visualisation of robot during exploring. Frontiers found in the map are visualised as blue arrows.

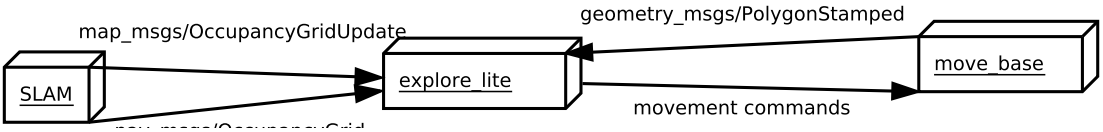


Figure B.2: Architecture of `explore_lite`, exploring node for ROS.

B.4 ROS API

B.4.1 explore

Provides exploration services offered by this package. Exploration will start immediately after node initialization.

Actions Called

`move_base` (move_base_msgs/MoveBaseAction)
: move_base actionlib API for posting goals. See move_base#Action API for details. This expects move_base node in the same namespace as `explore_lite`, you may want to remap this node if this is not true.

Subscribed Topics

`costmap` (nav_msgs/OccupancyGrid)
: Map which will be used for exploration planning. Can be either cost from move_base or map created by SLAM (see above). Occupancy grid must have got properly marked unknown space, mapping algorithms usually track unknown space by default. If you want to use costmap provided by move_base you need to enable unknown space tracking by setting `track_unknown_space: true`.

`costmap_updates` (map_msgs/OccupancyGridUpdate)
: Incremental updates on costmap. Not necessary if source of map is always publishing full updates, i.e. does not provide this topic.

Published Topics

`~frontiers` (visualization_msgs/MarkerArray)
: Visualization of frontiers considered by exploring algorithm. Each frontier is visualized as vector in the middle of frontier pointing towards unknown area.

Parameters

`~robot_base_frame` (string, default: `base_link`)
: The name of the base frame of the robot. This is used for determining robot position on map. Mandatory.

`~costmap_topic` (string, default: `costmap`)
: Specifies topic of source `nav_msgs/OccupancyGrid`. Mandatory.

`~costmap_updates_topic` (string, default: `costmap_updates`)

Specifies topic of source `map_msgs/OccupancyGridUpdate`. Not necessary if source of map is always publishing full updates, i.e. does not provide this topic.

~`visualize` (bool, default: `false`)
Specifies whether or not publish visualized frontiers.

~`planner_frequency` (double, default: `1.0`)
Rate in Hz at which new frontiers will computed and goal reconsidered.

~`progress_timeout` (double, default: `30.0`)
Time in seconds. When robot do not make any progress for `progress_timeout`, current goal will be abandoned.

~`potential_scale` (double, default: `1e-3`)
Used for weighting frontiers. This multiplicative parameter affects frontier potential component of the frontier weight.

~`orientation_scale` (double, default: `0`)
Used for weighting frontiers. This multiplicative parameter affects frontier orientation component of the frontier weight.

~`gain_scale` (double, default: `1.0`)
Used for weighting frontiers. This multiplicative parameter affects frontier gain component of the frontier weight.

~`transform_tolerance` (double, default: `0.3`)
Transform tolerance to use when transforming robot pose.

Required tf Transforms

`global_frame` → `robot_base_frame`
This transformation is usually provided by mapping algorithm. Those frames are usually called `map` and `base_link`. For adjusting `robot_base_frame` name see respective parameter. You don't need to set `global_frame`. The name for `global_frame` will be sourced from `costmap_topic` automatically.

B.5 Acknowledgements

This package was developed as part of my bachelor thesis at Charles University in Prague.

This project uses parts of frontier exploration algorithm from explore package by Charles DuHadway.