3D Maps Integration Based on Overlapping Regions Matching

Submitted: 23rd September 2021; accepted: 9th February 2022

Michał Drwięga

DOI: 10.14313/JAMRIS/3-2021/20

Abstract:

This paper presents a developed method of 3D maps integration based on overlapping regions detection and matching that works without an initial guess about transformation between maps. The presented solution is based on a classic pipeline approach from computer vision that has been applied to the 3D maps integration with multiple improvements related to model extraction and the descriptors matching. The process of finding transformation between maps consists of three steps. The first one is the extraction of the model from one of the maps. Then the initial transformation is estimated between extracted model and another map based on feature extraction, description, and matching. The assumption is that the maps have an overlapping area that can be used during the feature-based alignment. In the last step, the initial solution is corrected using local alignment approaches, for example, ICP or NDT. The maps are stored in the octree-based representation (octomaps) but during transformation estimation, a point cloud representation is used as well. In addition, the presented method was verified in various experiments: in a simulation, with wheeled robots, and with publicly available datasets. Eventually, the solution can be applied to many robotic applications related to the exploration of unknown environments. Nevertheless, so far it was validated with a group of wheeled robots. Furthermore, the developed method has been implemented and released as a part of the open-source ROS package 3d_map_server.

Keywords: *multi-robot mapping, map merging, feature matching, ICP, NDT, octomaps*

1. Introduction

The development of autonomous mobile robots has received much attention in recent years. Rapid progress in this area is stimulated by numerous possible applications like mine exploration [11], planetary exploration, scout robots, search and rescue, reconnaissance, home vacuum cleaning, lawn mowing or industrial applications, for instance, transport in warehouses [3]. However, it turns out that Multi-Robot Systems (MRS) have several advantages over Single Robot Systems in many of these applications. Especially, they are more time-efficient because tasks execution can be parallelized. Also, the multi-robot configuration can provide a higher level of reliability, for example, in case of malfunction of one of the robots.

One of the requirements for the creation of the autonomous robot is the ability to create a map of the unknown environment and localize the robot on it. The multi-robot mapping of unknown environments can also be performed more efficiently than a single robot mapping. First of all, it can be done faster when it is executed in parallel, which is especially important during mapping larger areas. Moreover, the robots can be equipped with different types of sensors what makes it possible to create more accurate and complete world models. Nonetheless, several new problems specific to multi-robot systems arise, like coordination of robots or communication between them.

Typically of multi-robot mapping, each robot creates its local map in the local coordinate frame. One of the intensively researched topics of mapping by multiple robots is merging all of these local maps into one global map [36]. However, to create such a globally consistent world model, a few problems have to be solved. The first one is finding transformation between maps. There are a few ideas on how to use the robots' initial poses and internal localization system to estimate transformation. But it may be not possible especially in the case of a big drift in the pose estimated by the local localization method.

Another idea is to integrate maps only during robots meetings. However, it needs in many cases an additional sensory system to detect other robots or a dedicated detection method. Also, this idea is limited because sending data between robots is not enough to estimate the orientation of robots and they have to see each other. Nevertheless, it is worth mentioning that systems that use partial information are developed as well. Such partial information could be the only distance between robots, calculated based on signal time-of-flight.

Another group of approaches depends on sending measurements from one robot to other robots and the assumption that maps have an overlapping area. With measurements from the other robot, it is possible to locate it on the map, for example, with a particle filter algorithm.

The last group of methods is based on the feature matching idea. The features are extracted from maps, identified, and matched. It is assumed like in the previous group that maps have an overlapping area that can be used during the matching process.

Nonetheless, the map alignment process is more challenging than, for example, 2D laser scan matching or depth sensors measurements matching because of displacement between maps or measurements. The displacement can reach bigger values especially when robots start mapping from totally different parts of en-

vironments. In the case of small displacement, algorithms that are based on the local optimization can be used. However, the maps alignment process needs a global optimization that is resistant to local minima.

1.1. Contribution

This paper presents a developed global 3D maps integration method with the alignment based on the feature matching which does not require an initial transformation estimation. Most of the other approaches for 3D maps integration are based on some kind of initial information or they are sensitive to the local minima. In contrast, this method does not need any initial cues and it deals with the local minima problem. The presented method is based on a classic pipeline approach from computer vision that has been applied to the 3D maps integration with multiple improvements related to model extraction and the descriptors matching. Moreover, the introduced model division into submodels improves the feature matching process and decreases the number of necessary calculations in the optimistic case. To my knowledge, there is no other application of feature matching with the model subdivision-based method for solving the 3D maps integration problem without an initial transformation guess.

The presented approach is based on the assumption that maps have the same scale and that they have an overlapping area. The intersection area of the maps can be extracted and used in the initial alignment step. In the initial alignment process, one of the maps is divided into regions, then each region is aligned to the second map. Finally, the best solution that consists of the transformation is selected according to the proposed quality measure, for example, a fitness score.

The method has been developed to work mainly with octree-based maps (octomaps) but during the merging process, the point clouds representation is used as well. Dual representation makes it possible to use the advantages of two representations and efficiently perform specific operations. But of course, the cost behind that is the increased memory usage. Moreover, the method was verified in multiple test cases based on data from real robots. It was confirmed that with some assumptions it is possible to merge large 3D maps from multiple robots.

Furthermore, the method of maps integration described in this paper has been implemented in C++ and released as the open-source software. The software is a part of the ROS (Robot Operating System) [1] package *3d_map_server* [10].

1.2. Problem Statement

This paper deals with the three-dimensional (3D) feature maps integration problem. Briefly, the problem can be defined as a data association between multiple representations of the same part of the environment.

Let's consider a system of N robots $R = \{r_1, r_2, \ldots, r_n, \ldots, r_N\}$ in $\mathbb{R}^3$ space, where the each robot creates its partial map M_n in a local coordinate system T_n. The map can be defined as a set of nodes $M_n = \{m_1, m_2, \ldots, m_{N_n}\}$ where $|M_n| = N_n$. In general, the octomaps integration can be defined as a creation of one, consistent model of the world M based on the k separate models which represents regions of the environment $M' = \{M_1, \ldots M_k\}$ (fig. 1). The problem can be narrowed down to the only two input models without loss of generality because of the assumption that more than two maps can be merged, for instance, recursively.

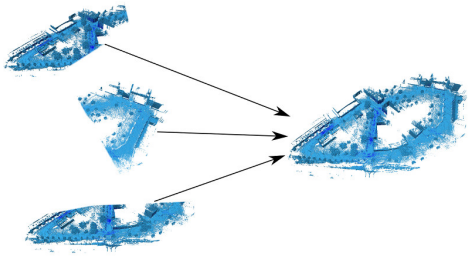

Fig. 1. Local maps created by different robots merged into a one consistent world model

Let's assume that exists a transformation T_2^1 between a pair of models that transforms M_1 and M_2 to the common coordinate system. The output map is a composition of the partial maps transformed to the coordinate system of the first map.

$$M = M_1 \cup T_2^1 M_2 \tag{1}$$

In the presented problem the solution consists of estimated transformations between the coordinate frames of the input maps. More precisely, the goal is to find the transformations under which the distances between corresponding nodes in input maps are minimized. In the real world, it is difficult to find the optimal transformation T_2^1 because of multiple inaccuracies sources. Errors are introduced by similarities on the maps and not enough distinctive descriptions of features. Also, the input maps have different scales, and sensors used for maps creation are not ideal.

1.3. Related Work

One of the basic and intensively researched topics of multi-robot mapping is a merging of local maps from robots into one global map [3]. Following the classification proposed in [21] approaches can be classified into two categories: a direct map merging and an indirect map merging.

In the direct map merging the system has additional information about transformation between maps. For example, this information can be acquired by visual or range measurements during the meeting of robots. Such an approach has been presented in [18]. It generates hypotheses by direct measurements between robots. Then robots move to a specific location and meet again. If a meeting happens and robots again detect each other correctly, then the hypothesis is accepted and maps are merged. In [40], also a solution based on robot-to-robot visual observations has been proposed. Based on the estimated transformation, maps are transformed and in the next step, it

is checked if maps overlap. In the case of maps overlapping the accuracy of robots relative poses is increased by landmarks detection on overlapped parts of maps by matching them. Another approach has been presented in paper [20]. This probabilistic map merging method depends on Rao-Blackwellized particle filters with unknown initial poses and models inter-robot measurements as Gaussian processes.

The other branch of direct map merging methods includes these based on the pose of overlapping area regions or objects in multiple maps. A probabilistic online map merging approach working with an omnidirectional visual system has been described in [32]. It has been used for one robot that collects partial maps but can be adapted to a multi-robot system as well. It uses a vision system to generate coarse transformations between maps based on the place recognition method. Then depending on that the bounding boxes are calculated for Haar-based place recognition that is able to discriminate new and previously visited locations.

On the other hand, indirect map merging is based on finding and matching the overlapping area of the maps which are not known a priori. These methods can be classified into three groups. In the first group, there are mostly approaches based on point feature detection and matching. In [19] the system for detection of the overlapping regions of maps has been described. Maps were created with ceiling-vision-based SLAM. The algorithm robustly detects the overlapping regions and estimates transformations for map alignment.

The next group includes methods based on the scan matching. In [34] the 2D local maps were merged by integration of Scale-Invariant Feature Transform (SIFT) algorithm to extract, describe and match features. Also, they used a well-known optimization technique in robotic mapping, the ICP(Iterative Closes Point) [4, 13] that finds a rigid transformation between two points sets. They used that data from the 2D laser scanner and consecutive scans were matched during the map creation. In [2] the virtual robot approach has been provided. It treats laser scans from multiple robots as range measurements to the virtual robot and generates its odometry data by detection of similar structures in local maps.

The last group includes spectral information-based methods. In [6, 22], some approaches have been presented that utilize the spectral information on 2D maps. They consider the maps merging as a binary image matching problem, so they use the Hough transform to structure and decompose the transformation into separate operations of rotation and translation. Paper [12] describes a method that uses geometric and topological similarities of vertices and edges to find a match between two maps.

With the growing demand for robotics services in complex, human environments, the need for 3D maps storing and processing methods will be essential. This kind of world representation allows robots to operate in the interior of multi-level buildings, inside cluttered rooms, or in rough terrain. Also, 3D maps are better suited to heterogeneous multi-robots systems [28, 30, 38], especially when robots use different sensors. Papers [9, 17] deal with the octomaps [16] merging problem with the ICP based methods. The alternative to the ICP approach for the 3D local alignment has been presented in [23,33]. It is based on NDT (*Normal Distribution Transform*) and is more efficient than the ICP algorithm because it doesn't require nearest neighbors search. Graph-based merging methods are also developed [5], and they have the advantage in inconsistencies reduction because of a backend graph optimization. The paper [7] presents a solution for the 3D point clouds alignment based on the transformation into the Radon/Hough domain. In [24] it was presented the comparison of different 3D maps matching approaches.

1.4. World Representations

An octomap [35,37] is a tree-based representation, which is memory efficient and well suited to large environments. It is built upon a recursive dividing of the world into eight cubic parts. One of the key features of octomap is the possibility to postpone the initialization of nodes until a robot visits a specific part of the environment. Naturally, it is not possible with fixed representations like voxel grids. However, the computational complexity of a node random access is $\mathcal{O}(\log d)$ and it depends on the depth of the octree. Another advantage of the octomap is the possibility of optimization. Blocks can be divided into smaller parts as long as output blocks are distinct. If the obtained blocks are similar enough, the branch is cut and the model is consistent.

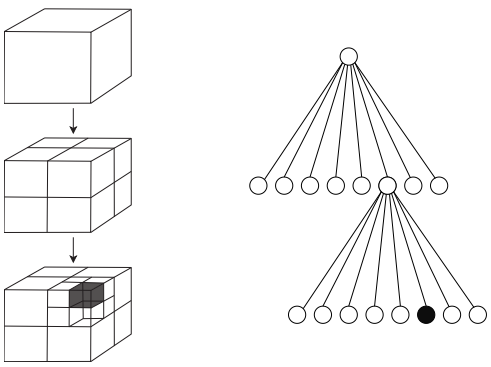

Fig. 2. Recursive space division which is the idea behind octomaps

As mentioned in section 1.2, the map can be defined as a set of nodes which determines the probability of occupancy of represented segments of the environment. Such probabilities are updated according to the following formula [16]:

$$p(n \mid z_{1:t}) = \left[1 + \frac{1 - p(n \mid z_t)}{p(n \mid z_t)} \frac{1 - p(n \mid z_{1:t-1})}{p(n \mid z_{1:t-1})} \frac{p(n)}{1 - p(n)}\right]^{-1} \tag{2}$$

where:
- z_t – denotes measurement at time t,

- $p(n)$ – a priori occupancy probability,
- $p(n \mid z_{1:t-1})$ – previous probability estimation,
- $p(n \mid z_t)$ – occupancy probability at measurement z_t calculated based on sensor model.

2. Maps Integration Method

In the proposed approach a few processing steps are necessary for each of the maps. The data pipelines have been presented in the fig. 3. On the input of the

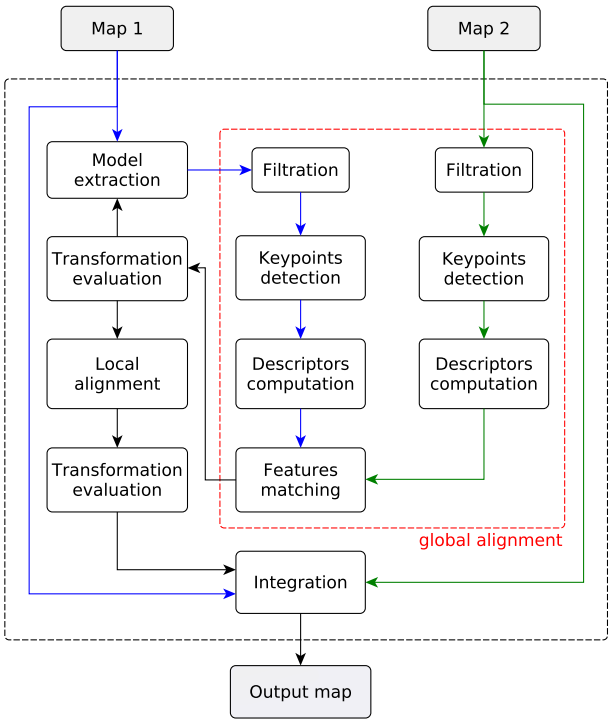

Fig. 3. The data pipelines in the presented maps merging approach

integration method, there are two 3D maps. Nevertheless, the important assumption is that maps have an overlapping area, which is necessary for correct operation. The whole maps merging process consists of two main parts: finding of the transformation between maps and data integration. The output of the algorithm is an integrated map.

The finding of the transformation could be divided into three steps:

- model extraction,
- global alignment,
- and local alignment.

One of the maps (map 2 in the fig. 3) is used directly and the other one is used for n models extraction. The process of models extraction is described in the following section.

In the global alignment, there are some common operations for both maps, like filtration, keypoints detection, and descriptors computation. Based on the computed descriptors and the assumption that maps have an overlapping area, the descriptors from maps are matched to each other with a randomized algorithm. As a consequence of one map division into multiple models, the result of initial alignment is a set of n hypotheses $H = \{h_1, \ldots, h_n\}$. Then, each hypothesis is evaluated based on the selected quality measure, for example, the fitness score. As a result, the best solution is selected from the set of accepted hypotheses H_A. If at least one hypothesis is accepted then the processing is continued and the final result is the transformation that transforms one of the maps to the coordinate system of the other map. Otherwise, the processing is stopped at this point.

In the next step, the transformation from the previous step is corrected in the process called a local alignment. For this purpose, variants of ICP and NDT algorithms were used.

However, to generate the integrated map that consists of both parts, it is necessary to include some additional steps that are parts of the data integration process. This process consists of a map conversion to the octomap, the transformation of one map to another coordinate system, and finally the combination of data of two maps into one consistent model.

2.1. Model Extraction From One of Maps

The first step is to divide the first map into rectangular blocks that are used as models. As shown in the fig. 4, a few cases of relations between two maps have to be considered. The maps integration can be done only in the first three cases when the overlapping region exists.

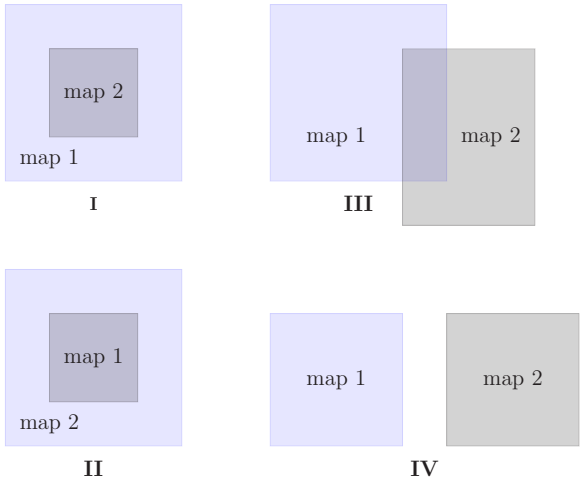

Fig. 4. Different cases of maps overlapping (I-III) or when there is no common part between maps (IV)

Moreover, it has been noticed that the process of maps integration can be speeded up in the most common case (fig. 4), when robots start exploration from the same place and explore different parts of the environment. Therefore, excluding the multi-floor mapping, kidnapped robot problem, and a case when one of the maps is entirely included in the second one, in most cases the overlapping area begins on the borders of both maps. Because of that, processing of the map begins from the outside and proceed in a spiral towards the center (fig. 5). To speed up the method and decrease the number of calculations, not all rectangular blocks are processed.

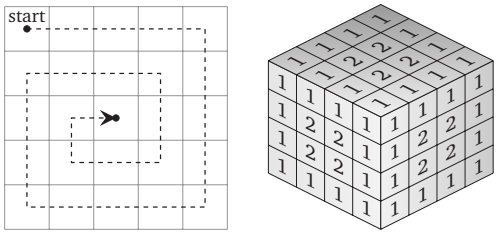

Fig. 5. A division of one map and extraction of the model - 2D and 3D cases

Also, assuming that robots move mostly parallel to the ground, then maps are rather limited in Z axis and wider in x and y axes. Therefore, map heights are significantly smaller than sizes in x and y axes. Then the division can be performed in two dimensions. In specific cases, for example, maps of multi-level buildings, the model can be extracted from one of the maps based on the 3D grid (fig. 5).

One of the stop criteria is a fitness score combined with a number of correspondences. So, if the criteria presented in the following part are satisfied, the processing of the remaining blocks is stopped. If a map has a lot of nodes near the border that is in the overlapping area with another map, it should stop fast - just after having processed a few blocks. However, in some cases, maps near the borders are not dense and then processing should be continued until the center of the map is reached.

2.2. Input Data Filtration

In the process of maps merging it is necessary to correctly prepare the data. Therefore, the preprocessing was done in a few steps.

The first one is a pass-through filtration that allows the rejection of points that are not inside the useful area. For example, the maps are cut in the z-axis to remove the ground and reduce the number of points to speed up further calculations.

In the next step, the point cloud is downsampled with a voxel grid filter. The idea is to divide space into voxels with specified sizes and approximate points in each voxel.

The last filtration step is the removal of outliers that is based on a statistical analysis of neighboring points, and points that do not meet the requirements are removed. In this work, an approach [27] that computes the distribution of the point to neighbors distances has been applied. Assuming that resulted distribution is Gaussian, points which are outside a specified range are filtered out.

2.3. Keypoint Detection

After filtration, surface normals are estimated (fig. 6). The algorithm for each point finds neighboring points in a specified search radius and then estimates a fitting plane with the least-square algorithm [25]. Based on the estimated plane equation $ax + by + cz + d = 0$, the surface normal is calculated $n = [a, b, c]$.

The descriptors calculations is a computationally expensive step. Therefore, the descriptors are computed only in selected points - keypoints. The properly selected keypoints should be distinctive and repeatable to deal with noises or different points of view. Commonly used approaches that deals with keypoints extraction in 3D data are NARF detector [29], ISS [39] (fig. 7) or modified Harris detector [14]. Another, less computationally demanding solution that has been applied to this work is uniform sampling. It creates a 3D voxel grid over the input point cloud and then in each voxel the present points are approximated with their centroid.

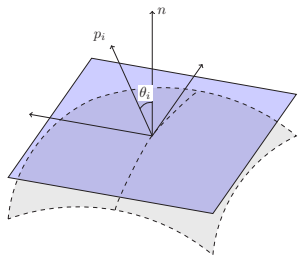

Fig. 6. Normal vector in specified point and the θ_i angle between the normal vector n and point p_i

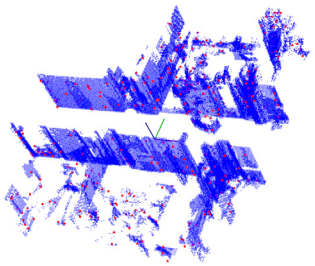

Fig. 7. An example of keypoints (red points) detection with the ISS method

2.4. Local Features Description

In order to find similar areas on the maps, all detected features should be described in the most concise possible way that makes it easy to compare. Therefore, in each keypoint selected in the previous step, the local descriptor is calculated. Local descriptors describe a local neighborhood around a query point on the surface. The widely used is FPFH method that is a local version of PFH [26] around a given keypoint. It is based on pairing the query point with neighbors and calculations of orientation differences for each pair. After that, a weighted sum of the orientations differences is computed and the output vector is created by concatenation of histograms.

However, for the purpose of the maps integration method, a SHOT (Signature of Histograms of Orientations) descriptor [31] has been used. It is based on the spherical support (neighbors points) that is divided into spatial segments (fig. 8). For each segment, the descriptor calculates a histogram representing the distribution of the cos θ_i, where θ_i is the angle between the surface normal in each point from the support and the surface normal vector n in the query point (fig. 6).

Then all local histograms are combined into one vector and the descriptor is created. One of the advantages of the SHOT descriptor is the possibility to utilize texture information like a point color received from the RGB-D sensor.

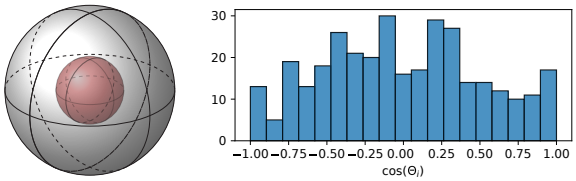

Fig. 8. A spatial division of the spherical support and histogram of cos θ_i value range

2.5. Features Matching

The features matching uses the descriptors extracted and computed in the previous steps. The first descriptors set D_M is calculated for keypoints of extracted model $M' = \{m_i \mid m_i \in \mathbb{R}^3, i = 1, \ldots, N_M\}$ and the second set D_S for the scene $S' = \{s_i \mid s_i \in \mathbb{R}^3, i = 1, \ldots, N_S\}$. Those two sets of descriptors are matched to each other. It means that for each point from the set M' it should be found the point in S' with a similar descriptor that relates to the same region in the second map. After that, one should calculate a transformation that minimizes distances between pairs of descriptors.

To match the descriptors sets, the SAC (Sample Consensus) alignment approach was used [26]. The algorithm idea is similar to the RANSAC (Random Sample Consensus) [8] method and random matching of k-nearest neighbors. The method consists of three steps:

- Randomly select k points from the set M' and them do set $P = \{p_i \mid p_i \in \mathbb{R}^3, i = 1, \ldots, k\}$,
- For each $p_i \in P$, find points with similar descriptors in S' and randomly select from them the one that will make a pair with the point from P,
- For each pair of points, called correspondence, compute the transformation between points and the error metric.

Repeating the above steps allows to avoid local minima and find transformation which minimizes the error metric. It is not an final solution but rather an initial guess for the next step which is a local alignment. The example of pairs of features matching between two maps has been shown in the fig. 9.

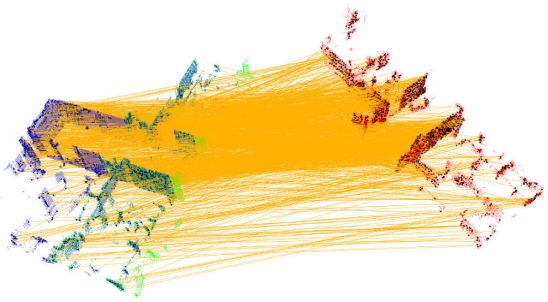

Fig. 9. Matching of feature pairs from two maps

2.6. Local Alignment

As a final step of transformation estimation, the local correction is applied. For this purpose, the scene is cropped to the size of the model inflated by a specified distance d_m, as shown in the fig. 10.

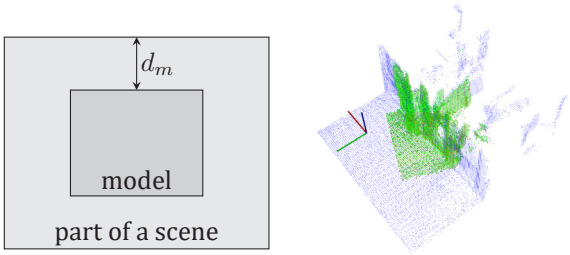

Fig. 10. The scene cropped to the size of the model with the inflation distance. On the right side, there is a model matched to the cropped scene

After that, the ICP based approach is used to correct locally a transformation between the scene $S = \{s_i \mid s_i \in \mathbb{R}^3, i = 1, \ldots, N_S\}$ and the model $M = \{m_i \mid m_i \in \mathbb{R}^3, i = 1, \ldots, N_M\}$. The idea behind this method depends on matching one map (scene) D to the second one M called model in such a way as to minimize distances between pairs of points (nearest neighbors) from both sets. The steps in k-th iteration of algorithm are as follows:

- $\forall m_i^k \in M$ find the closest point (nearest neighbor) $s_i^k \in S$,
- Minimize distances between corresponding points pairs with the least squares method

$$E(R,t) = \frac{1}{N_M} \sum_{i=1}^{N_M} ||Rm_i + t - s_i^k||^2, \qquad (3)$$

- Transform model according to estimated rotation R and translation t

$$M^{k+1} = RM^k + t^k, \qquad (4)$$

- Terminate if error value is below the threshold τ .

The final transformation can be estimated by repeating the above steps.

2.7. Evaluation of Estimated Transformation

To evaluate the solution, a fitness score f_s has been computed. It computes the mean error between pairs of corresponding points (p_i, q_i) based on the nearest neighbors calculation. The pairs of points are placed in two points sets. The first one is $P = \{p_i \mid p_i \in \mathbb{R}^3, i = 1, \ldots, n\}$ and the second one is $Q = \{q_i \mid q_i \in \mathbb{R}^3, i = 1, \ldots, n\}$. Nevertheless, the standard version of the fitness score is not very robust, for instance, it can be calculated on the basis of a small number of pairs, which means it is not very reliable. So, there were applied a few modifications. One of them is the calculation of binary weights for each pair depending on the maximum distance between corresponding points d_{th}, so only pairs with smaller distances are considered in calculations. Also, if the number of pairs with positive weight

n_w is below the threshold value, the fitness score is marked as worthless.

$$w_i = \begin{cases} 1, & \text{if } \|p_i - q_i\| \leq d_{th} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$n_w = \sum_{i=1}^{n} w_i \tag{6}$$

$$f_s = \begin{cases} \sum_{i=1}^{n} \frac{\|p_i - q_i\|_2}{n_w} w_i, & \text{if } n_w \geq n_{th} \\ +\infty, & \text{otherwise} \end{cases} \tag{7}$$

where:

- n is a number of corresponding pairs,
- n_{th} is a threshold value for the number of pairs.

The presented formula allows avoidance of false good nodes matching because of the use of a threshold. Without it, it is possible to get a low fitness score only based on a small number of points pairs.

3. Validation

The maps integration algorithm has been validated in numerous experiments. The performed experiments could be divided into three parts. The first part is based on data from publicly available datasets from Freiburg University, released under *Creative Commons Attribution License CC 3.0* [15]. The second part consists of experiments with two Turtlebot robots. The last part contains the performance evaluation for different maps alignment methods.

3.1. Experiments Based on Datasets

Maps placed in dataset [15] has been created based on data from a SICK LMS laser scanner was placed on a pan-tilt unit. As a result, the accuracy of maps is better in comparison to maps created with the RGB-D sensor but they do not provide information about the color of the surface.

The selected results of the maps merging based on the datasets have been placed in the figures 11 - 13.

Other results are shown in table 1, where:

- n_1 and n_2 denote sizes (a number of nodes) of two input maps,
- T_R is a real transformation between maps in format $(x, y, z, roll, pitch, yaw)$,
- r is an approximated size of an overlapping area of both maps as a percentage of the full map,
- f_s is a fitness score of the best solution from all extracted models,
- T_{err} is an error value between real and estimated transformation and is calculated as $T_{err} = \|T_{est} \cdot T_R^{-1} - I_4\|_F$,
- t denotes processing time in seconds.

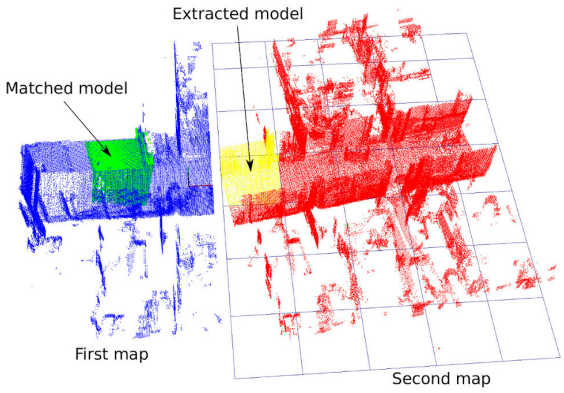

Fig. 11. Model matching and transformation estimation between two maps. The first map (blue) was used as a scene. From the other map (red), a model (yellow) has been extracted and matched to the first map. The matched and transformed model was marked as a green region. The real transformation between maps was $T_R = (7.5, 0.2, 0.1, 1^o, 1^o, 5^o)$

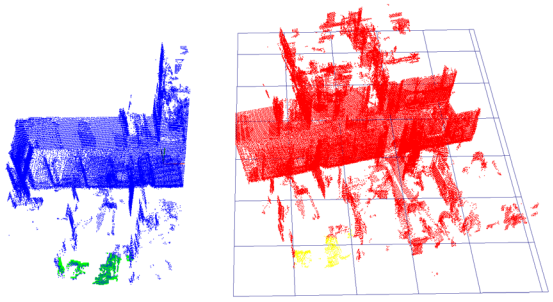

Fig. 12. Another example of finding transformation between maps, with different real, initial transformation $T_R = (9, 1.0, 0.1, 1^o, 2^o, 15^o)$

Tab. 1. Results of transformations estimation

n_1	n_2	T_R	r	f_s	T_{err}	$t[s]$
$4.0 \cdot 10^5$	$8.3 \cdot 10^5$	$(12, 6, 0.5, 5^o, 5^o, 60^o)$	16%	0.028	0.25	15
$6.5 \cdot 10^5$	$8.3 \cdot 10^5$	$(12, 6, 0.5, 5^o, 5^o, 30^o)$	32%	0.023	0.007	26
$8.9 \cdot 10^5$	$9.9 \cdot 10^5$	$(14, 4, 0.5, 5^o, 3^o, 45^o)$	64%	0.023	0.006	40
$1.0 \cdot 10^6$	$1.0 \cdot 10^6$	$(15, 3, 0.5, 5^o, 5^o, 45^o)$	84%	0.023	0.009	44

3.2. Experiment With Turtlebot Robots

To validate the integration algorithm on noisy data representing scenes encountered in mobile robotics applications, the experiments with two mobile robots Turlebots (fig. 14) were performed. The robots were equipped with an odometry system, laser scanner Hokuyo UST-10LX and RGB-D sensor Intel RealSense D435. The system used for the octomaps creation was presented in the fig. 15. Similarly like in the simulation, it was built upon the ROS framework, and it used the GMapping SLAM.

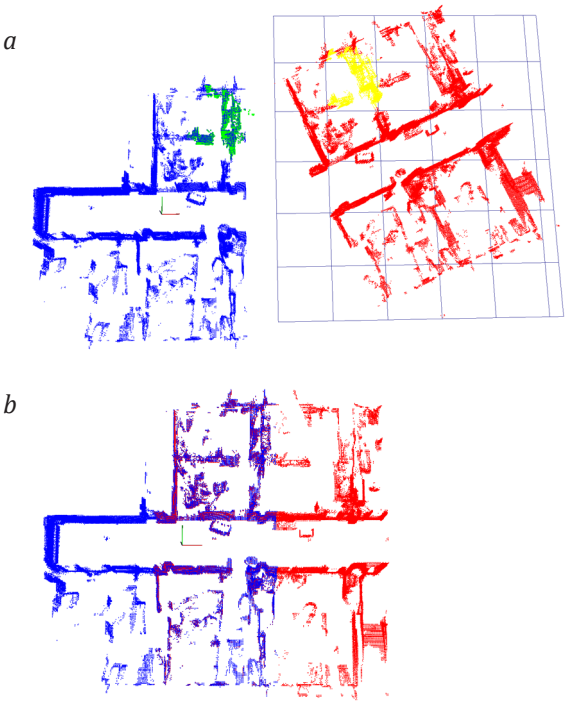

Fig. 13. Example of two maps integration (a) with real transformation between maps $T_R = (10, 1.5, 0.1, 3^o, 2^o, 25^o)$ and an integrated map (b) as a result

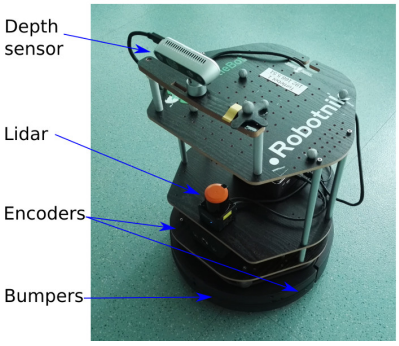

Fig. 14. The Turtlebot mobile robot with the sensors

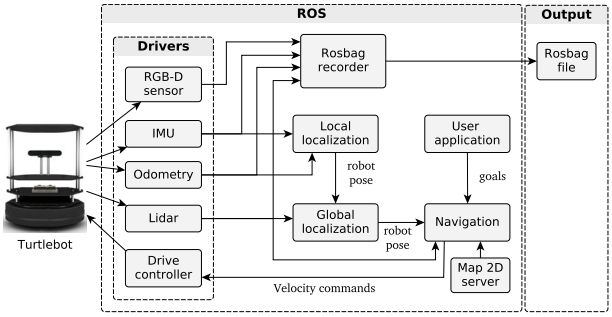

Fig. 15. The high-level control system used to create the octomap by the Turtlebot robot

During the experiment with Turtlebots in a robotics laboratory and corridor localized in the campus of Wrocław University of Science and Technology, robots moved following the paths shown in fig. 16. Results from experiments have been shown in figures 17-19.

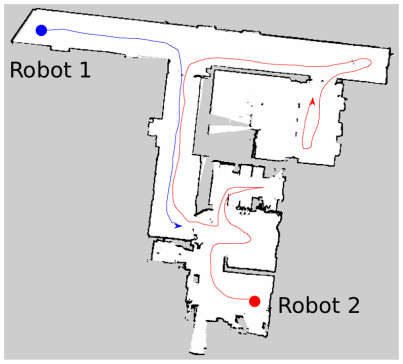

Fig. 16. Paths of two robots during the experiment

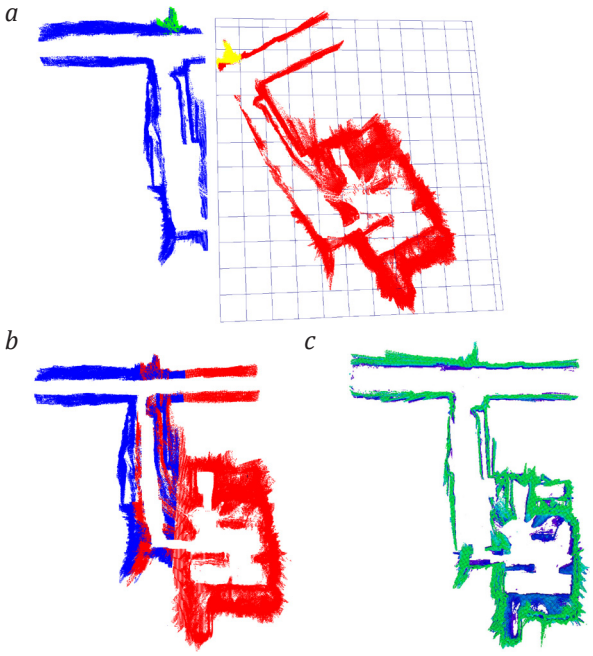

Fig. 17. An experiment with a mapping of the laboratory in Wrocław University of Science and Technology. The figure shows maps before merge (a) with extracted (yellow) and matched model (green). Also it contains an output map (b) and a map of the same area created by one robot (c)

3.3. Performance Evaluation of Alignment Algorithms

The performance of different maps integration method modifications has been evaluated. The mean value of error has been calculated for multiple testing cases which contain distinctive maps of the environment.

The diagram (fig. 20) contains mean errors (T_{err}) for different global alignment methods. The following global alignment methods were evaluated: SAC (*Sample Consensus*) and GCC (*Geometry Consistency Clustering*). Additionally, it was checked if a division of one map into models (model extraction process) could speed up the alignment procedure. The cases with *div* postfix in name, have been using model division.

The diagram (fig. 21) contains mean errors (T_{err}) for combinations of local and global alignment methods. As a local methods, ICP [4] and NDT [23] have been compared.

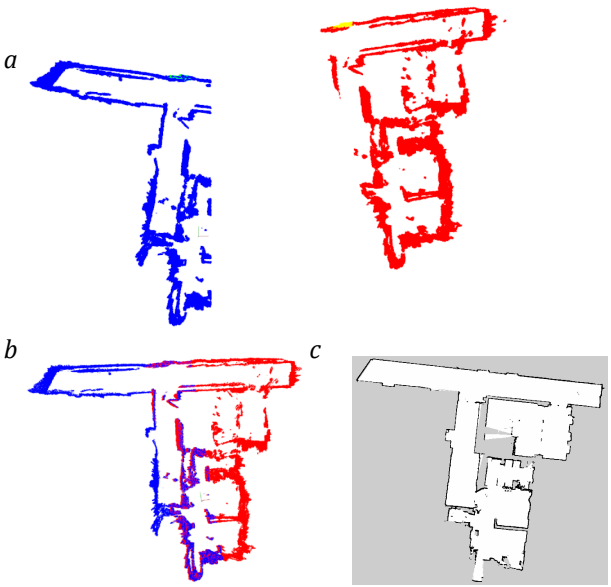

Fig. 18. Integration of maps from another robots run, from the same location (a,b) but with extended area. It has been shown also the 2D map of the same area (c)

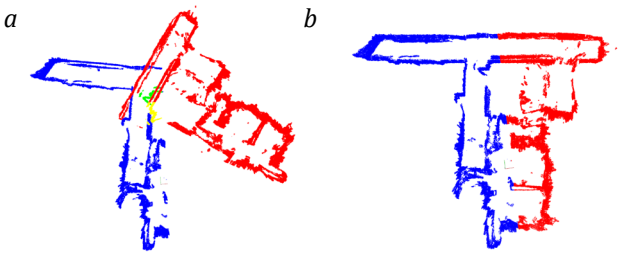

Fig. 19. The last case of two maps integration from Turtlebots robots

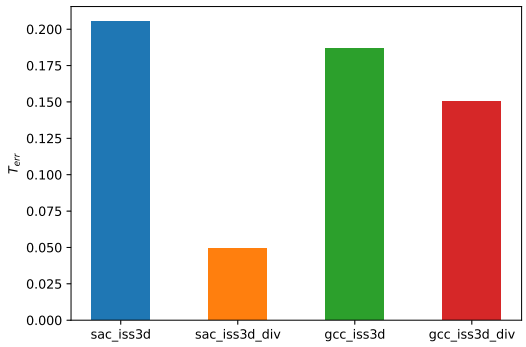

Fig. 20. Mean error between real and estimated transformations for global alignment methods

4. Conclusion

The paper presents the approach to 3D maps integration problem without the initial knowledge about the relative poses of robots. It is based on feature detection and matching techniques. To speed up processing, randomized algorithms were used. Also, some optimization steps for the most common cases were introduced like processing maps from the outside part to the center.

The evaluation uses publicly available data sets

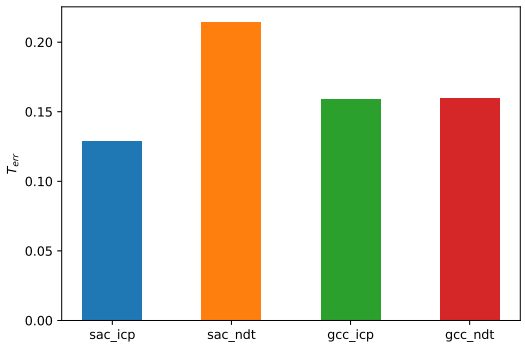

Fig. 21. Mean error between real and estimated transformations for combination of global and local alignment methods

and data from experiments with Turtlebots. The results show that the approach is effective in various environments. So, when properly tuned and maps overlap enough (above 15% of common space) the method is quite robust.

However, the merging method has some weaknesses. One of them is the computational cost of the data processing pipeline, especially during the first robots met when initial alignment is necessary. Then processing can take hundreds of seconds for two $20m \times 20m \times 2m$ maps. Also, it turns out to be hard to find a transformation between octomaps that contain a ground plane as the ground plane is wrongly matched between maps. Of course, it can be fixed relatively easily, by removing the ground plane from maps or by introducing another keypoints detection method. Besides of that the open topic is a scaling of the method as it was tested only on two robots so far.

Another issue that is still not addressed is the loop closure problem. Currently, it was assumed that a local merging error is low enough and its influence is negligible. However, it is not true and after multiple merging processes it can increase to a significant value and have an impact on the quality of the map. One of possible improvements can be providing a high-level graph-based approach to manage multiple partial maps from robots together with some kind of backend SLAM method.

Further research will be also directed to the optimization of algorithms, especially that many of the operations can be processed in parallel.

AUTHOR

Michał Drwięga – Department of Cybernetics and Robotics, Wrocław University of Science and Technology, ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland, e-mail: michal.drwiega@pwr.edu.pl.

ACKNOWLEDGEMENTS

This research was supported by the National Science Centre, Poland, under the project number 2016/23/B/ST7/01441.

REFERENCES

[1] "ROS: robot Operating System". https://www.ros.org/. Accessed on: 2022-04-21.

[2] N. Adluru, L. J. Latecki, M. Sobel, and R. Lakaemper, "Merging maps of multiple robots". In: *2008 19th International Conference on Pattern Recognition*, Tampa, FL, USA, 2008, 1–4, 10.1109/ICPR.2008.4761036.

[3] I. Andersone, "The Characteristics of the Map Merging Methods: A Survey", *Scientific Journal of Riga Technical University. Computer Sciences*, vol. 41, no. 1, 2010, 113–121, 10.2478/v10143-010-0032-8.

[4] P. Besl and N. D. McKay, "A method for registration of 3-D shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, 1992, 239–256, 10.1109/34.121791.

[5] T. M. Bonanni, B. Della Corte, and G. Grisetti, "3-D Map Merging on Pose Graphs", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, 2017, 1031–1038, 10.1109/LRA.2017.2655139.

[6] S. Carpin, "Fast and accurate map merging for multi-robot systems", *Autonomous Robots*, vol. 25, no. 3, 2008, 305–316, 10.1007/s10514-008-9097-4.

[7] A. Censi and S. Carpin, "HSM3D: Feature-less global 6DOF scan-matching in the Hough/Radon domain". In: *2009 IEEE International Conference on Robotics and Automation*, Kobe, 2009, 3899–3906, 10.1109/ROBOT.2009.5152431.

[8] K. G. Derpanis. "Overview of the RANSAC Algorithm". http://www.cs.yorku.ca/~kosta/CompVis_Notes/ransac.pdf, 2010. Accessed on: 2022-04-21.

[9] M. Drwiega, "Features Matching based Merging of 3D Maps in Multi-Robot Systems". In: *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, Międzyzdroje, Poland, 2019, 663–668, 10.1109/MMAR.2019.8864711.

[10] M. Drwięga. "3d_map_server". https://github.com/mdrwiega/3d_map_server, March 2022. Accessed on: 2022-04-21.

[11] A. Ferrein, I. Scholl, T. Neumann, K. Krückel, and S. Schiffer. "A System for Continuous Underground Site Mapping and Exploration". In: M. Reyhanoglu and G. De Cubber, eds., *Unmanned Robotic Systems and Applications*. IntechOpen, April 2020.

[12] S. Gholami Shahbandi and M. Magnusson, "2D map alignment with region decomposition", *Autonomous Robots*, vol. 43, no. 5, 2019, 1117–1136, 10.1007/s10514-018-9785-7.

[13] J. Han, P. Yin, Y. He, and F. Gu, "Enhanced ICP for the Registration of Large-Scale 3D Environment Models: An Experimental Study", *Sensors*, vol. 16, no. 2, 2016, 228, 10.3390/s16020228.

[14] C. Harris and M. Stephens, "A Combined Corner and Edge Detector". In: *Procedings of the Alvey Vision Conference 1988*, Manchester, 1988, 23.1–23.6, 10.5244/C.2.23.

[15] A. Hornung. "OctoMap 3D scan dataset - Arbeitsgruppe: Autonome Intelligente Systeme". http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/. Accessed on: 2022-04-21.

[16] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees", *Autonomous Robots*, vol. 34, no. 3, 2013, 189–206, 10.1007/s10514-012-9321-0.

[17] J. Jessup, S. N. Givigi, and A. Beaulieu, "Robust and efficient multi-robot 3D mapping with octree based occupancy grids". In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, San Diego, CA, USA, 2014, 3996–4001, 10.1109/SMC.2014.6974556.

[18] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, "Map merging for distributed robot navigation". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, vol. 1, Las Vegas, Nevada, USA, 2003, 212–217, 10.1109/IROS.2003.1250630.

[19] H. S. Lee and K. M. Lee, "Multi-robot SLAM using ceiling vision". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, 2009, 912–917, 10.1109/IROS.2009.5354435.

[20] H.-C. Lee, S.-H. Lee, M. H. Choi, and B.-H. Lee, "Probabilistic map merging for multi-robot RBPF-SLAM with unknown initial poses", *Robotica*, vol. 30, no. 2, 2012, 205–220, 10.1017/S026357471100049X.

[21] 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: *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Daejeon, Korea (South), 2012, 285–287, 10.1109/URAI.2012.6462995.

[22] K. Lee, C. Jung, and W. Chung, "Accurate calibration of kinematic parameters for two wheel differential mobile robots", *Journal of Mechanical Science and Technology*, vol. 25, no. 6, 2011, 1603–1611, 10.1007/s12206-011-0334-y.

[23] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT", *Journal of Field Robotics*, vol. 24, no. 10, 2007, 803–827, 10.1002/rob.20204.

[24] M. Magnusson, N. Vaskevicius, T. Stoyanov, K. Pathak, and A. Birk, "Beyond points: Evaluating recent 3D scan-matching algorithms". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, 2015, 3631–3637, 10.1109/ICRA.2015.7139703.

[25] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments", *KI - Künstliche Intelligenz*, vol. 24, no. 4, 2010, 345–348, 10.1007/s13218-010-0059-6.

[26] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration". In: *2009 IEEE International Conference on Robotics and Automation*, Kobe, 2009, 3212–3217, 10.1109/ROBOT.2009.5152473.

[27] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, 1–4, 10.1109/ICRA.2011.5980567.

[28] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review: Multiple-Robot Simultaneous Localization and Mapping", *Journal of Field Robotics*, vol. 33, no. 1, 2016, 3–46, 10.1002/rob.21620.

[29] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries". In: *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, 2601–2608, 10.1109/ICRA.2011.5980187.

[30] H. Surmann, N. Berninger, and R. Worst, "3D mapping for multi hybrid robot cooperation". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, 2017, 626–633, 10.1109/IROS.2017.8202217.

[31] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3D feature matching". In: *2011 18th IEEE International Conference on Image Processing*, Brussels, Belgium, 2011, 809–812, 10.1109/ICIP.2011.6116679.

[32] F. Tungadi, W. L. D. Lui, L. Kleeman, and R. Jarvis, "Robust online map merging system using laser scan matching and omnidirectional vision". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, 2010, 7–14, 10.1109/IROS.2010.5654446.

[33] C. Ulas and H. Temeltas, "A 3D Scan Matching Method Based On Multi-Layered Normal Distribution Transform", *IFAC Proceedings Volumes*, vol. 44, no. 1, 2011, 11602–11607, 10.3182/20110828-6-IT-1002.02865.

[34] K. Wang, S. Jia, Y. Li, X. Li, and B. Guo, "Research on map merging for multi-robotic system based on RTM". In: *2012 IEEE International Conference on Information and Automation*, Shenyang, China, 2012, 156–161, 10.1109/ICInfA.2012.6246800.

[35] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems". In: *Proc. of the ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, vol. 2, 2010.

[36] S. Yu, C. Fu, A. K. Gostar, and M. Hu, "A Review on Map-Merging Methods for Typical Map Types in Multiple-Ground-Robot SLAM Solutions", *Sensors*, vol. 20, no. 23, 2020, 6988, 10.3390/s20236988.

[37] Y. Yue, D. Wang, P. Senarathne, and D. Moratuwage, "A hybrid probabilistic and point set registration approach for fusion of 3D occupancy grid maps". In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Budapest, Hungary, 2016, 001975–001980, 10.1109/SMC.2016.7844529.

[38] Y. Yue, C. Yang, Y. Wang, P. G. C. N. Senarathne, J. Zhang, M. Wen, and D. Wang, "A Multilevel Fusion System for Multirobot 3-D Mapping Using Heterogeneous Sensors", *IEEE Systems Journal*, vol. 14, no. 1, 2020, 1341–1352, 10.1109/JSYST.2019.2927042.

[39] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition". In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, Kyoto, Japan, 2009, 689–696, 10.1109/ICCVW.2009.5457637.

[40] X. Zhou and S. Roumeliotis, "Multi-robot SLAM with Unknown Initial Correspondence: The Robot Rendezvous Case". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 2006, 1785–1792, 10.1109/IROS.2006.282219.