Chapter

Multi-Robot Mapping Based on 3D Maps Integration

Michał Drwiega and Elżbieta Roszkowska

Abstract

An unknown environment could be mapped more efficiently by a group of robots than a single robot. The time reduction due to parallelization is crucial in complex area mapping. There are two general solutions used in the multi-robot mapping. In the first one, robots exchange raw data from sensors. The second approach assumes that each robot creates a local map independently that is exchanged with other robots and integrated. In this chapter, we present a 3D maps integration algorithm that utilizes overlapping regions in the feature-based alignment process. The algorithm does not need any initial guess about the transformation between local maps. However, for successful integration, maps need to have a common area. We showed that the implemented method is effective in various environments. The approach has been verified in experiments with wheeled mobile robots and using public datasets with octree-based maps.

Keywords: multi-robot mapping, MR-SLAM, feature-matching, octomaps, ICP

1. Introduction

The development of autonomous mobile robots is accelerated by various applications like underground exploration [1, 2], planetary exploration, autonomous cars, search and rescue, reconnaissance, home cleaning, lawn mowing, or industrial applications. In many of these implementations, Multi-Robot Systems (MRS) have several advantages over a single robot. First of all, task execution time may be reduced due to parallelization. Moreover, the multi-robot system can provide a higher level of reliability. Even in the case of a single robot failure, other robots can complete the task. These features are crucial, for example, in search and rescue applications when robots find survivors in the extreme underground environment after accidents.

The multi-robot mapping of unknown environments can also be performed more efficiently than a single robot mapping. It is essential in the case of large and complex area mapping. Moreover, the robots can be equipped with different kinds of sensors to create more accurate world models. Nonetheless, several challenges specific to multi-robot systems arise, for instance, proper coordination of robots or handling the communication between them.

In general, there are two solutions used in multi-robot mapping. In the first one, robots exchange raw data from sensors. The second approach assumes that each robot creates a local map in the local coordinate system independently. Then local maps are exchanged between robots and integrated into one global map [3]. This chapter focuses on the second solution.

The key part of the map merging process is an estimation of the transformations between maps. The map alignment process is more challenging than consequent sensor data frames matching because of larger displacement between maps or measurements. There are several possible solutions on how to find this transformation. One of them assumes using the robots' initial poses and their local localization systems. However, it is not possible in all cases because of localization drift.

Another approach is to exchange and merge maps only during robot meetings. It assumes the use of additional sensors or methods to detect other robots when they see each other. Even partial information could be helpful during the transformation estimation, so in some systems, only the distance between robots is calculated based on the signal time-of-flight.

It is also possible to use a feature matching-based approach to get the relative poses of robots. In this case, the features are extracted from maps and matched. However, the transformation could be estimated successfully only if maps have an overlapping area.

This paper presents the design and implementation of the global 3D maps integration method with the alignment based on feature matching which does not require an initial transformation estimation.

1.1 Related work

One of the basic tasks of multi-robot mapping is a merging of local maps from individual robots into one global map [4–6]. As mentioned before, the most challenging part of the maps merging is finding the transformation between partial maps.

Some approaches use additional information about transformation between maps. For instance, it can be acquired by visual or range measurements during the robots meeting. In ref. [7], an approach has been presented that is based on the direct measurements between robots. Each direct detection creates a hypothesis that is validated during the next meetings of the same robots but in other locations. Maps are merged only if robots meet again and the hypothesis is accepted.

Map merging methods are also based on the idea of finding and matching the overlapping area in the maps. Such overlapping areas are not known before. In ref. [8], the system for detection of overlapping regions in maps created with ceiling-vision-based SLAM has been described. The algorithm can detect the overlapping regions and estimate transformations between partial maps.

Another approach has been included in ref. [9]. It uses an omnidirectional visual system and the initial version was intended for only one robot that creates partial maps. Nevertheless, the method can be applied also to multi-robot systems. The algorithm uses a vision system to generate coarse transformations between partial maps. The additional level of validation is based on the calculated bounding boxes with the Haar-based place recognition method.

The next group includes methods based on sensor measurements matching, for example, scan matching. In [10], the 2D local maps were merged with SIFT (*Scale-Invariant Feature Transform*) algorithm that allows to extract, describe and match features. Another optimization technique that was used is the ICP (*Iterative Closes Point*) [11, 12] which finds a rigid transformation between two point sets. It has been utilized to find a transformation between consecutive 2D scans during the map creation process.

In ref. [13], it has been presented as a spectral information-based method dedicated to 2D maps. It assumes that the map merging problem is a binary image matching problem. The algorithm allows using for instance Hough transforms to decompose the transformation into separate rotation and translation. Another method that uses geometric and topological similarities of vertices and edges to find a match between two maps has been presented in ref. [14].

A major part of the mentioned methods was supposed to be used with 2D maps. However, 3D maps have received much attention in recent years due to the growing demand for robotics services in complex environments to coexist with humans but also challenging extreme underground or underwater scenarios. The world representation in three dimensions allows robots to operate in multi-level buildings, in cluttered rooms but also in rough terrain. Moreover, such maps have better performance in heterogeneous multi-robot systems [15–17], especially when robots are equipped with different sensors. Several studies, for example [18, 19], have proposed solutions for the octomaps [20] integration problem with local alignment methods like ICP. It is worth mentioning that the octomap [21] is a tree-based representation built upon a multiple dividing of the world into eight cubic parts. Such representation is memory efficient and could be successfully used for large environments.

The mentioned map integration methods have a significant drawback. They do not optimize solutions in the background. It means that once maps are aligned the transformation is not corrected anymore. To solve this issue graph-based methods were developed [22] that benefit from backend graph optimization.

In ref. [23], local alignment method NDT (*Normal Distributions Transform*) has been presented. The comparison of NDT with ICP [24] shows that it is more efficient and in most cases, it converges from a broader range of initial poses. However, authors have noticed also that the NDT is less predictable than ICP in cases with smaller initial pose errors.

The ref. [25] presents the 3D point sets alignment algorithm that utilizes the transformation into the Radon/Hough domain.

As in opposition to the methods based on the local features description and matching, in ref. [26], an approach with higher-level descriptors based on lines or planes has been presented. Authors have noticed that higher-level descriptors improve performance in the case of small maps overlapping.

1.2 Contribution

This chapter presents a developed global 3D maps integration algorithm with the alignment based on feature matching. In contrast to many other approaches, it does not require an initial transformation estimation and is not sensitive to the local minima. The approach is based on a classic computer vision pipeline that has been modified and applied to the 3D maps integration. Moreover, the introduced model division into submodels procedure improves the feature matching process performance and increases the efficiency of data processing in many cases.

The method uses octree-based maps (octomaps) which allow to utilize the additional information like nodes' occupancy probability in the alignment process. The approach was verified in multiple test cases based on data from real robots. Furthermore, the algorithm has been implemented in C++ and released as open-source software (*3d_map_server* [27]).

1.3 Problem statement

Let us consider a system of N robots, in which each robot creates its partial map M_n in a local coordinate system T_n. Each map consists of a set of N_n nodes $M_n = \{m_1, m_2, \dots, m_{N_n}\}$. The maps integration problem can be defined as the creation of the consistent model of the world M based on a set of k separate models $M' = \{M_1, \ldots M_k\}$ (**Figure 1**).

In the next part of the chapter, the problem has been narrowed down to only two input models. However, in the case of more than two maps, they can be integrated sequentially.

2. Maps integration method

The presented maps integration algorithm consists of a few processing steps (**Figure 2**). On the input, there are two 3D maps (octomaps). To integrate them successfully, they must have an overlapping area. The map merging process consists of two major steps: finding the transformation between maps and data aggregation. On the output of the algorithm, there is an integrated map.

The most complex part of the algorithm is the process of the transformation finding. It consists of three steps: model extraction, global alignment, and local alignment. In the presented pipeline, map 2 is used entirely but map 1 is used to extract n models.

In the global alignment, there are common operations for both maps, like filtration, keypoints detection, and feature description. The feature descriptors from both maps are matched to each other with dedicated algorithms. Because one map has been divided into multiple models, the result of the initial alignment consists of n hypotheses $H = \{h_1, \ldots, h_n\}$ (each for a separate model).

Each hypothesis is validated with quality measures, for example, the fitness score. Finally, the best solution is selected from the set of accepted hypotheses H_A. If at least one hypothesis is accepted then the maps merging process is continued and the final result is the transformation that transforms one of the maps into the coordinate system of the other map. Otherwise, the processing is stopped at this point.

If the solution has been found in the previous step, then it is corrected with a local alignment algorithm. To generate the final map that consists of partial maps, it is necessary to transform maps to the common coordinate system and aggregate data from them into one model.

2.1 Models extraction

As mentioned before, one map is divided into multiple rectangular blocks that are used as models. Several cases of relations between maps have to be considered (**Figure 3**). The map integration can be finished successfully only in the first three cases when the area that corresponds to the model could be found in another map. In

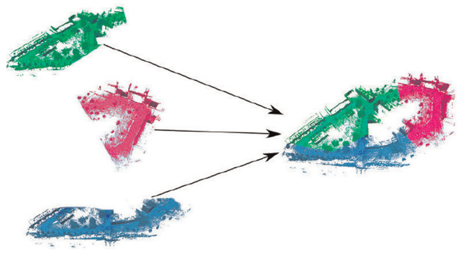

Figure 1.
Partial maps created by robots and merged into one model.

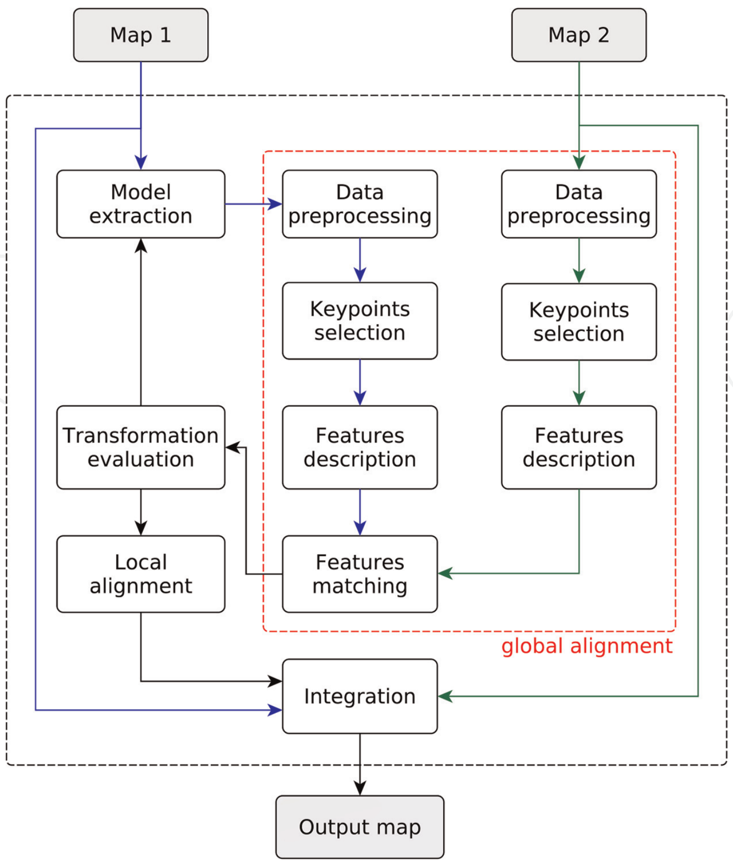

Figure 2.
The architecture of the developed maps integration algorithm.

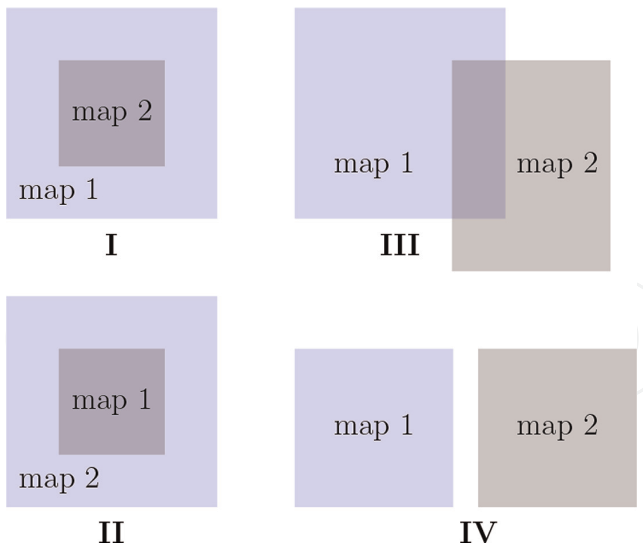

Figure 3.
Maps overlapping cases (I-III) and the case when there is no common part between maps (IV).

the last case, the integration method will be stopped because there is no common area that could be matched.

The maps integration process can be speeded up when robots start exploration from the same place and explore separate parts of the environment. Therefore, excluding the kidnapped robot problem, and a case when one of the maps is entirely included in the second one, the overlapping area begins on the borders of both maps.

Because of that, the processing of the map starts from the outside part and proceeds in a spiral toward the center of it. Concurrent models are processed in parallel until an acceptable solution is found.

If robots move on a flat surface, maps are limited on the Z axis and wider in x and y axes. In such a case, the map division in two dimensions is sufficient. Nonetheless, in specific cases like multi-level building maps, the model can be extracted from one of the maps based on the 3D grid (**Figure 4**).

2.2 Data preprocessing

The first step of data processing is pass-through filtration which rejects points that are not inside the specific area. For instance, the maps are reduced in the z-axis to get rid of the ground and reduce the number of points, and speed up further calculations.

Then, the point cloud is downsampled with a voxel grid filter. The idea behind it is to divide space into voxels with specified sizes and represent each voxel by a center point calculated as an approximation of all original points from this voxel.

The last step is outliers removal based on a statistical analysis of neighboring points. The points that do not meet the requirements are removed from the set.

2.3 Keypoints selection

The feature description is a computationally expensive operation. Therefore, the descriptors are computed only in carefully selected points - key points. The keypoints should be distinctive and repeatable to deal with noises and different points of view.

In this work, the ISS (*Intrinsic Shape Signatures*) [28] detector has been used. The ISS detector determines the relevance of the point based on eigenvalues of the matrix created for the support (local neighborhood) of a specific point. Let $X = \{x_1, x_2, \dots, x_N\}$ be a support of the keypoint k. Then,

$$\Sigma_k = \frac{1}{N}\sum_{i=1}^{N}(x_i - p_k)(x_i - p_k)^T, \tag{1}$$

be covariance matrix calculated for the support of k, where

$$p_k = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

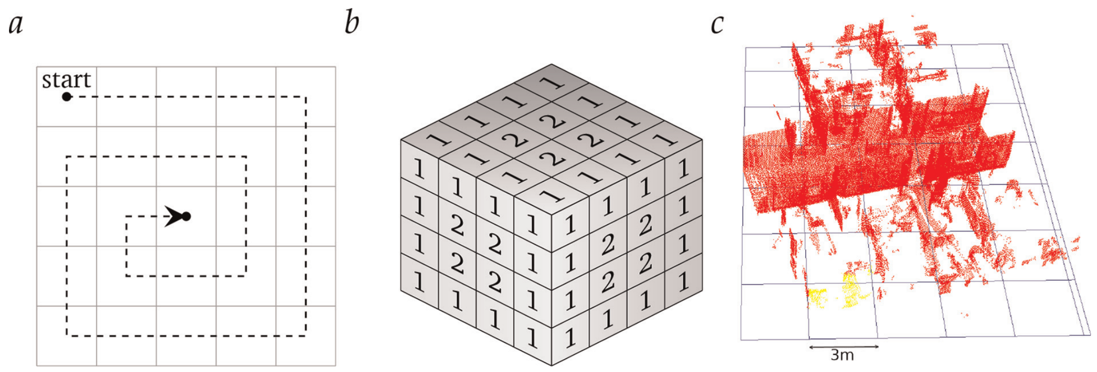

Figure 4.
The models extraction procedure with processing order - 2D (a), 3D case (b), and an example with real map (c).

is a mean point. The keypoint is relevant only if

$$\frac{\lambda_2}{\lambda_1} \leq \gamma_{12} \wedge \frac{\lambda_3}{\lambda_2} \leq \gamma_{23}, \tag{3}$$

where γ_{12} i γ_{23} are arbitrary selected parameters and $\lambda_1 < \lambda_2 < \lambda_3$ are eigenvalues of Σ_k.

2.4 Features description

Features description creates a local surface representation that could be easily compared what is crucial in the process of similar features finding. Therefore, the local descriptor is computed in each keypoint (**Figure 5**). Local descriptors describe a local neighborhood of a query point.

A local descriptor that was used is a SHOT (*Signature of Histograms of Orientations*) descriptor [29]. It uses the spherical support that is divided into spatial segments (**Figure 6**). For each segment, it is calculated a histogram representing the distribution of the $\cos\theta_i$, 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. Finally, all local histograms are combined into a vector that is the descriptor. It is also possible to utilize texture information like a point color received from the RGB-D sensor.

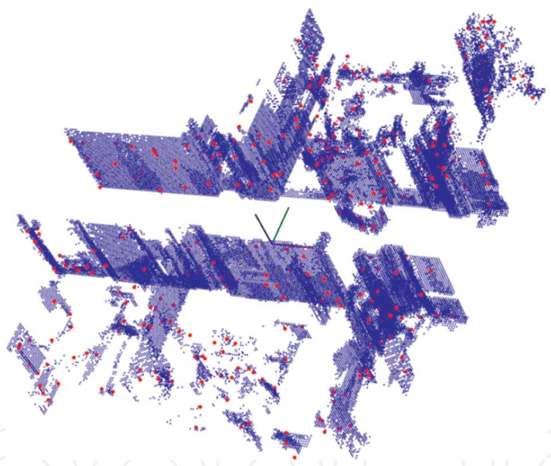

Figure 5.
Points on the example map, for which descriptors were calculated.

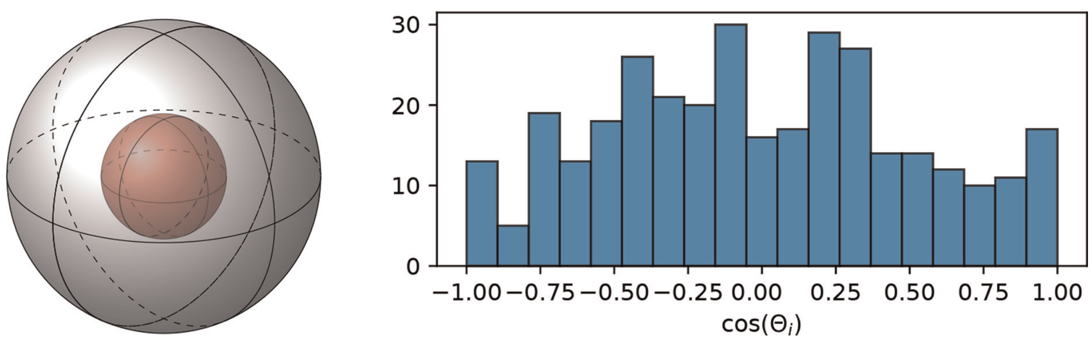

Figure 6.
The spherical support of SHOT descriptor divided into 32 spatial parts and histogram of $\cos\theta_i$.

2.5 Feature matching

To estimate the transformation between maps, it's necessary to find similar features in two maps and matched them to each other. The first feature descriptors set D_M is created for the keypoints of the model $M' = \{m_i | m_i \in \mathbb{R}^3, i = 1, \dots, N_M\}$ and the second set D_s for the scene $S' = \{s_i | s_i \in \mathbb{R}^3, i = 1, \dots, N_S\}$. During the matching process, for each keypoint from the set M', it should be found the point in S' with a similar feature descriptor. Finally, it is calculated a transformation that minimizes distances between pairs of descriptors.

Two matching methods were used. The first is a randomized algorithm SAC (Sample Consensus) [30]. The idea behind it is to use a random search of corresponding features in two sets. The algorithm in each iteration consists of three steps:

- Randomly select k points from the set M' and add them to set $P = \{p_i | p_i \in \mathbb{R}^3, i = 1, \dots, k\}$,
- 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 corresponding points, compute the transformation between points and check the error metric.

The second global alignment method is a GCC (Geometry Consistency Clustering) [31]. It uses the correspondences grouping based on the geometrical distances between them (**Figure 7**).

2.6 Local alignment

Finally, to improve the previous solution, the local alignment is applied. For this purpose, the scene is cropped to the size of the model inflated by a distance d_m (**Figure 8**). Then, the ICP-based algorithm is used to correct a transformation between the scene $S = \{s_i | s_i \in \mathbb{R}^3, i = 1, \dots, N_S\}$ and the model $M = \{m_i | m_i \in \mathbb{R}^3, i = 1, \dots, N_M\}$. The method matches one map (scene) S to the second one M called model in a way that minimizes distances between pairs of points

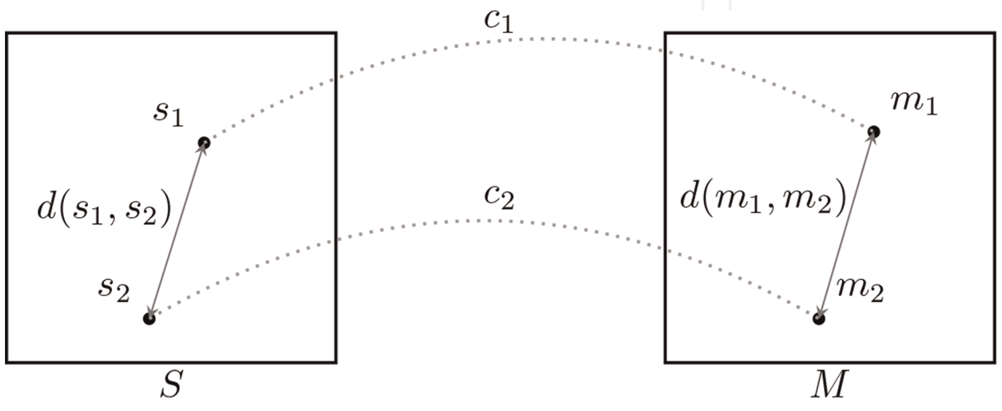

Figure 7.
Correspondences $c_1, \dots, c_n$ grouping in GCC method based on distances between point features in two sets.

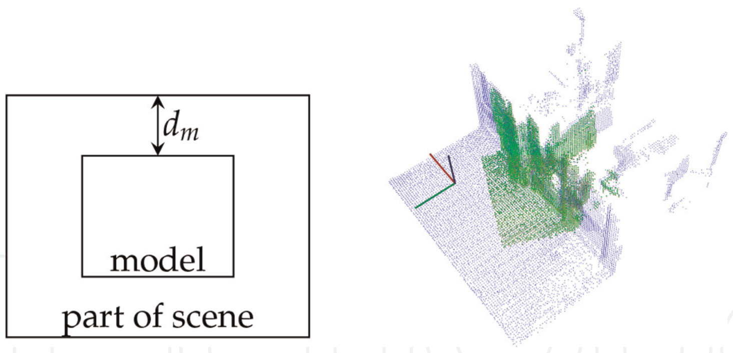

Figure 8.
The scene cropped to the size of the inflated model. On the right side, there is a model matched to the scene.

(nearest neighbors) from both sets. It is done in multiple iterations and a single k-th iteration consists of the following steps:

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

$$E(R,t) = \frac{1}{N_M} \sum_{i=1}^{N_M} \left\| R m_i + t - s_i^k \right\|^2, \tag{4}$$

- Transform model according to estimated rotation R and translation t

$$M^{k+1} = RM^k + t^k, \tag{5}$$

- Terminate if the error value is below the threshold τ.

Also, other methods were used for the local corrections. The first is the OICP (*Occupancy Iterative Closest Point*) which is a variant of ICP that utilizes additional information - occupancy in minimized goal function. Such information is included in occupancy-based maps like octomaps. The second method was the NDT [23] which divides space into voxels and estimates normal distribution parameters in each voxel. Parameters of the normal distributions are calculated with Eqs. (1 and 2).

2.7 Transformation evaluation

The solutions have been evaluated with a computed fitness score

$$f_s = \begin{cases} \sum_{i=1}^{n} \frac{\left\| p_i - q_i \right\|_2}{n_w} w_i, & \text{if } n_w \geq n_{th} \\ +\infty, & \text{otherwise} \end{cases}. \tag{6}$$

It represents the mean error between n pairs of corresponding points (p_i, q_i) from two data sets $P = \{p_i | p_i \in \mathbb{R}^3, i = 1, \ldots, n\}$ and $Q = \{q_i | q_i \in \mathbb{R}^3, i = 1, \ldots, n\}$. Nevertheless, the basic fitness score that is a mean distance is not very robust, for instance, it can be calculated on the basis of a small number of pairs. Because of that, binary weights

$$w_i = \begin{cases} 1, & \text{if} \|p_i - q_i\| \le d_{th} \\ 0, & \text{otherwise} \end{cases}, \tag{7}$$

were calculated for each pair depending on the maximum distance between corresponding points d_{th}. Due to that, only pairs with smaller distances are considered in calculations. Also, if the number of pairs with a positive weight

$$n_w = \sum_{i=1}^{n} w_i \tag{8}$$

is below the threshold value n_{th}, the fitness score is set to infinity. The presented metric avoids 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 presented maps integration method has been validated in numerous experiments. The first set of experiments was based on data from public datasets from Freiburg University. Another experiment contains data from two Tuerlebot robot runs. Finally, different map alignment methods were compared.

3.1 Experiments with public datasets

The dataset prepared at Freiburg University has been released under the *Creative Commons Attribution License CC 3.0* [32]. It contains maps created with a 2D SICK LMS laser scanner that was placed on a pan-tilt unit to get the additional dimension. As a result, the accuracy of maps is quite better than the accuracy of maps created with the RGB-D sensor but they do not provide information about the surface color.

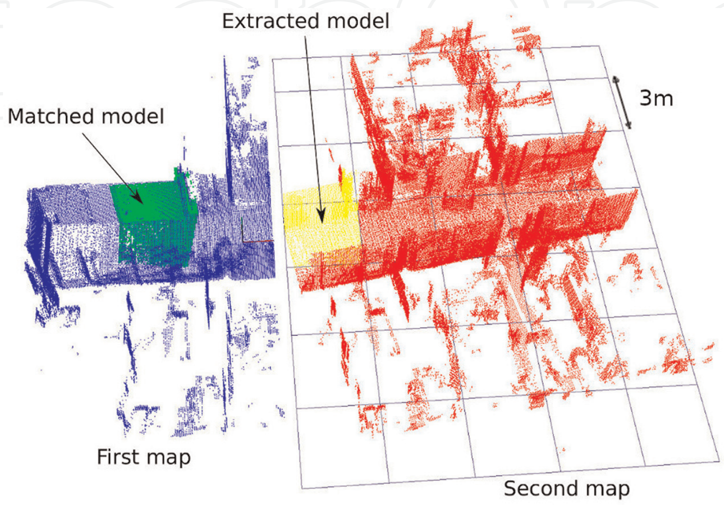

Figure 9.
Global alignment example. A model (yellow) has been extracted from the one map (red) and it was matched to the second map (blue).

Figure 9 shows the example of the global alignment process. The selected results of the maps integration have been placed in **Figures 10** and **11**. Other results are shown in **Table 1**, where:

- n_1 and n_2 are sizes (a number of nodes) of 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,

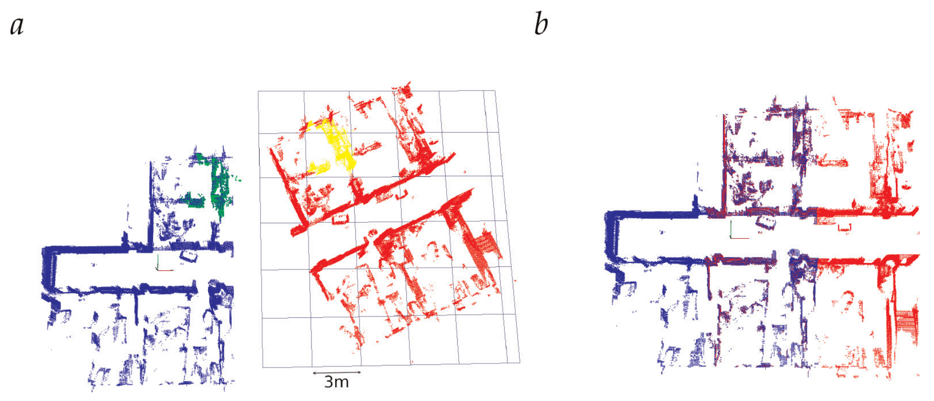

Figure 10.
Indoor maps integration (a) and an integrated map (b).

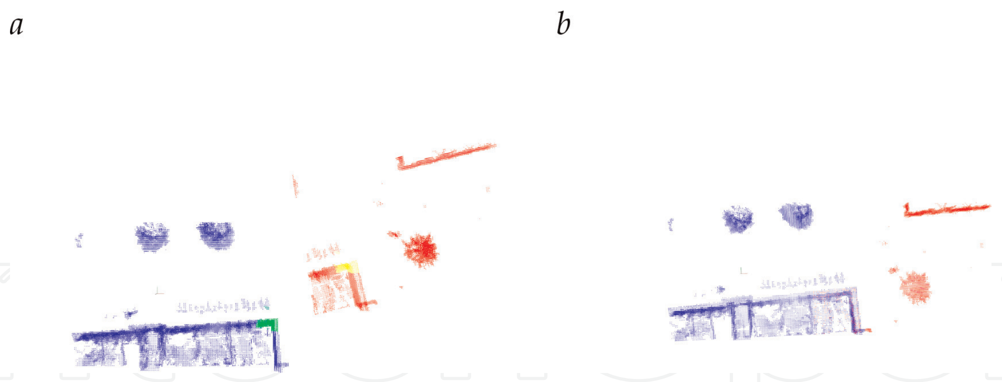

Figure 11.
Outdoor maps integration (a) and the integrated map (b).

n_1	n_2	$T_R\ (x, y, z, R, P, Y)$	r	f_s [m]	T_{err}	t [s]
51k	67k	(10, 1.5, 0.1, 3°, 2°, 25°)	12%	0.02	0.6	4.3
		(10, 1.5, 0.1, 3°, 2°, 25°)	24%	0.002	0.16	2.1
		(10, 1.5, 0.1, 0,0,0°)	36%	0.0006	0.10	1.9
		(12, 6, 0.5, 5°, 5°, 60°)	24%	0.0009	0.21	2.4

Table 1.
Results of transformations estimation.

- T_{err} is an error value between real and estimated transformation and is calculated as matrix norm $T_{err} = \|T_{est} \cdot T_R^{-1} - I_4\|_F$,
- t is a processing time in seconds.

3.2 Experiment with turtlebots robots

To validate the integration algorithm with data captured with RGB-D sensor, the experiments with two mobile robots Turlebots (**Figure 12**) were carried out. The Turtlebots were equipped with an odometry system, lidar Hokuyo UST-10LX and RGB-D sensor Intel RealSense D435.

The system used for the octomaps creation (**Figure 13**) was built upon the ROS framework and the GMapping SLAM. During the online session, the 2D SLAM

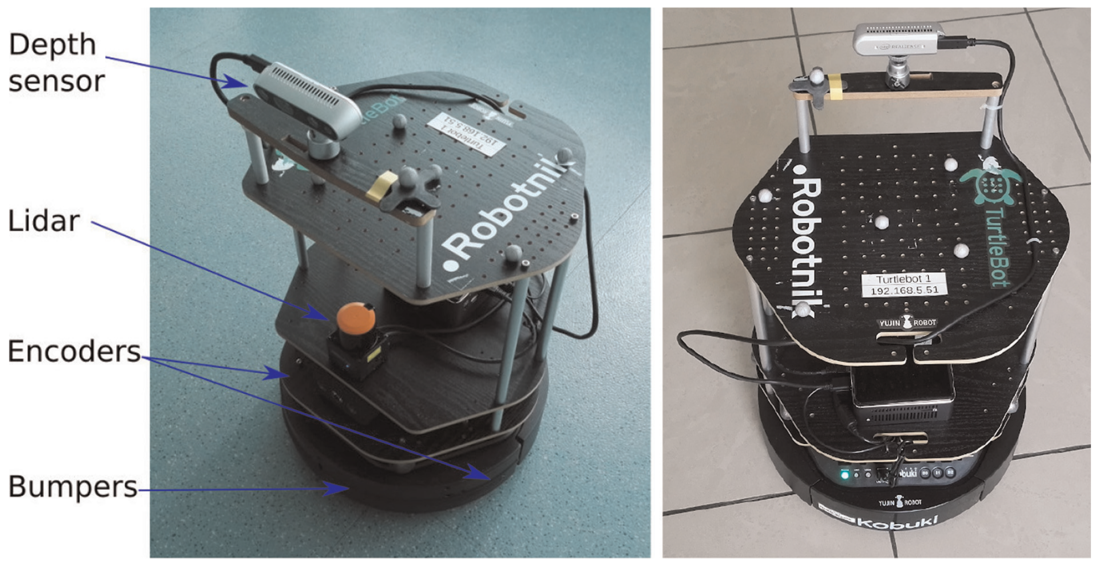

Figure 12.
Turtlebot robot equipped with multiple sensors used for localization and mapping.

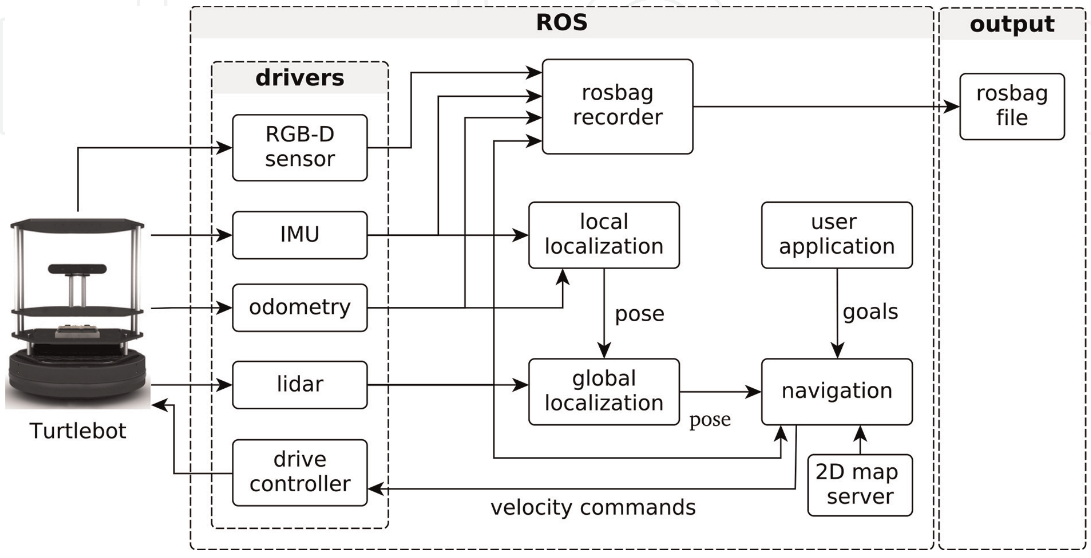

Figure 13.
The high-level control system of the mobile robot used to create the octomap.

algorithm estimated robot pose based on data from IMU, lidar, and odometry. On the other hand, the offline 3D map creation (octomap) used data from RGB-D sensor and pose estimated by SLAM.

The experiments with mobile robots were carried out in a robotics laboratory and corridor localized on the campus of Wrocław University of Science and Technology (**Figure 14**). Experiments results have been shown in **Figures 15 and 16**.

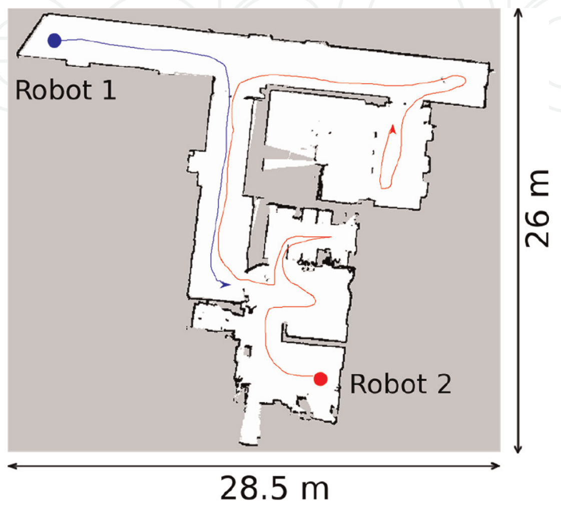

Figure 14.
The paths of two robots followed during the experiment. The paths are presented on the 2D map created for the same localization with lidar and GMapping SLAM.

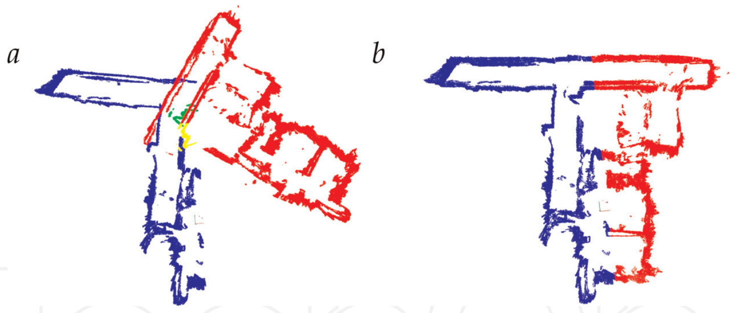

Figure 15.
An experiment with a mapping of the laboratory in WrocÅ‚aw University of Science and Technology. The figure shows maps before merging (a) with extracted (yellow) and matched model (green). On the other side, there is a merged map (b).

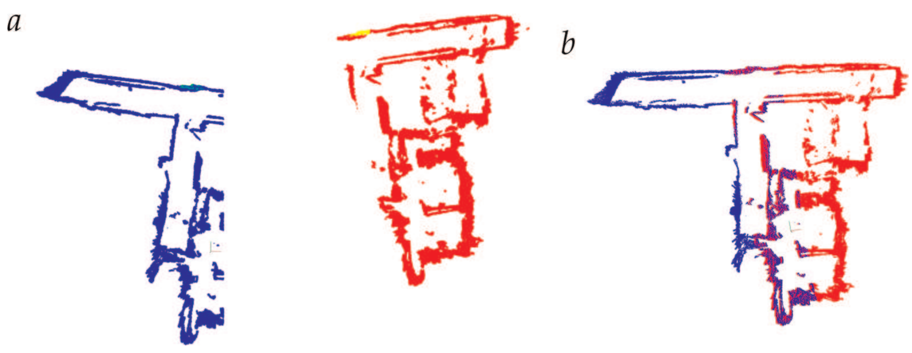

Figure 16.
Integration of maps from the same location but with different initial poses of robots.

3.3 Comparison of the alignment methods

The performance of alignment algorithm variants has been evaluated in multiple testing cases which contain distinctive maps. The mean value T_{err} has been calculated for all testing cases. Additionally, it was verified if a division of one map into models (model extraction process) could speed up the alignment procedure. Therefore, there are the comparison of two kinds of methods, with *DIV* postfix and without it. Added postfix means that method uses a model extraction procedure.

The diagram (**Figure 17**) shows mean errors for different local alignment and global alignment methods checked separately. On the other hand, the diagram (**Figure 18**) contains mean errors for combinations of local and global alignment methods. Additionally, **Figure 19** shows how map overlapping percentage affects the alignment error.

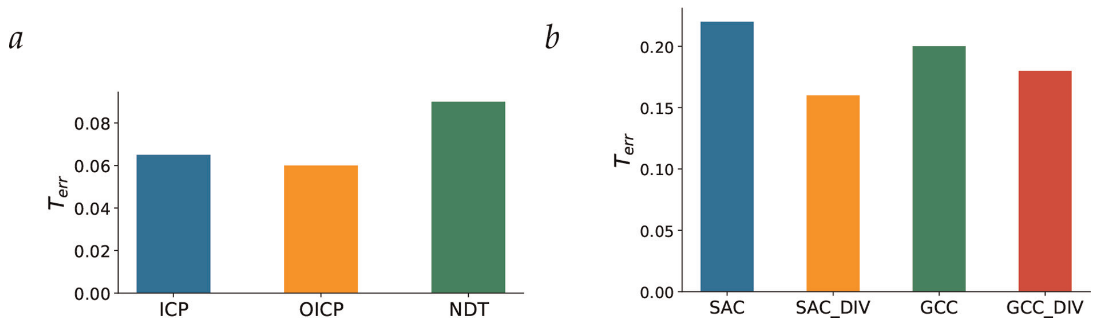

Figure 17.
Mean error between real and estimated transformations for (a) local alignment methods, and global alignment methods.

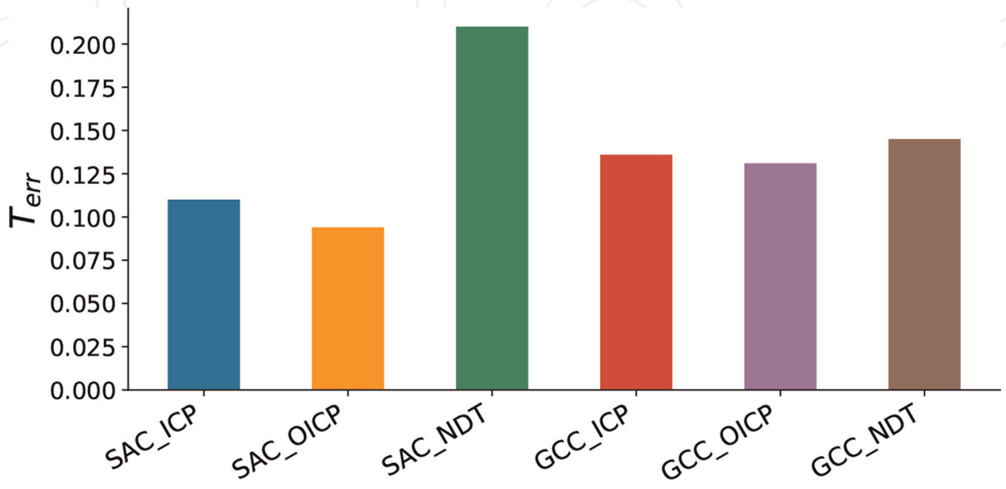

Figure 18.
Mean error between real and estimated transformations for combination of global, and local alignment methods.

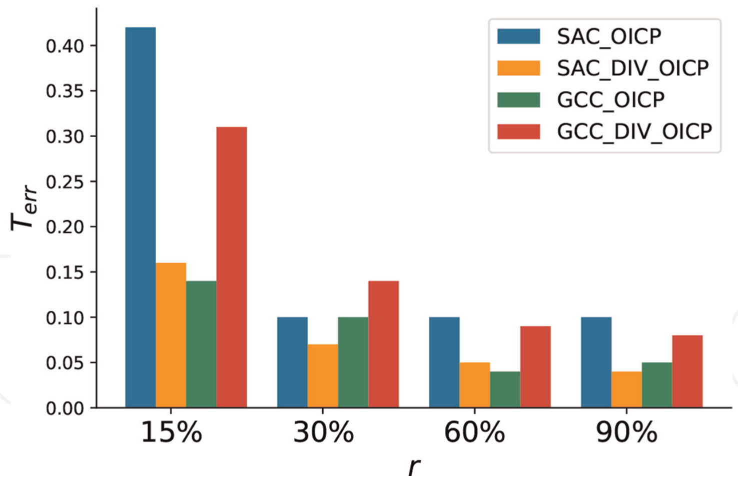

Figure 19.
Comparison of mean error depending on maps overlapping percentage.

4. Conclusions

The chapter presents a 3D maps integration algorithm that does not need initial knowledge about the relative poses of robots. Instead, it uses the feature extraction and matching idea. Therefore, the algorithm needs an overlapping area between maps to integrate them successfully.

The algorithm validation is based on public datasets with octomaps and experiments with two mobile robots. Multiple variants of the method have been validated and compared, especially different combinations of local and global alignment methods, but also additional model extraction procedure.

The results show that the approach is effective in various environments, especially if maps have at least 15% of common area. The best results have been noticed for probabilistic SAC method with local alignment OICP and enabled models extraction procedure.

On the other hand, the maps integration method has some limitations. The computational cost of the data processing is significant, especially the cost of the global alignment step that is necessary always during the first maps exchange between specific pair of robots. The whole integration process takes about 5 seconds in the case of two maps with sizes about $20m \times 20m \times 2m$ and resolution equal to 5 cm. It has been also noticed that it is hard to find a transformation between maps that contain a ground plane because the ground plane does not contain enough distinctive features. Nonetheless, it can be easily improved, by removing the ground plane from maps. The scaling also needs some attention, as the method was already tested only with two robots.

A significant drawback of the approach that is still not addressed is the loop closure problem. In the current version, it was assumed that local alignment errors are small enough and their influence is negligible. However, it is not true and multiple maps integrations lead to increased errors, what has an impact on the quality of the output map. This issue may be solved by providing a high-level graph-based approach with graph optimization to manage multiple partial maps from robots.

Acknowledgements

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

Author details

Michał Drwiega* and Elżbieta Roszkowska
Department of Cybernetics and Robotics, Wrocław University of Science and Technology, Wrocław, Poland

*Address all correspondence to: drwiega.michal@gmail.com

References

[1] Chang Y, et al. LAMP 2.0: A Robust Multi-Robot SLAM System for Operation in Challenging Large-Scale Underground Environments

[2] Ferrein A, Scholl I, Neumann T, Krückel K, Schiffer S. A system for continuous underground site mapping and exploration. In Mahmut Reyhanoglu and Geert De Cubber, Unmanned Robotic Systems and Applications. London, United Kingdom: IntechOpen; 2019. Available from: https://www.intechopen.com/chapters/67435

[3] Yu S, Fu C, Gostar AK, Hu M. A review on map-merging methods for typical map types in multiple-ground-robot SLAM solutions. Sensors. 2020; **20**(23):6988

[4] Andersone I. The characteristics of the map merging methods: A survey. Scientific Journal of Riga Technical University. Computer Sciences. 2010; **41**(1):113-121

[5] Lee H-C, Lee S-H, Lee T-S, Kim D-J, Lee B-H. A survey of map merging techniques for cooperative-SLAM. In: 2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). Vol. 1. Daejeon, Korea (South): IEEE; 2012. pp. 285-287

[6] Magnusson M, Vaskevicius N, Stoyanov T, Pathak K, Birk A. Beyond points: Evaluating recent 3D scan-matching algorithms. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). Vol. 1. Seattle, WA, USA: IEEE; 2015. pp. 3631-3637

[7] Konolige K, Fox D, Limketkai B, Ko J, Stewart B. Map merging for distributed robot navigation. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453). Vol. 1. Las Vegas, Nevada, USA: IEEE; 2003. pp. 212-217

[8] Lee HS, Lee KM. Multi-robot SLAM using ceiling vision. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 1. 2009. p. 6

[9] Tungadi F, Lui WLD, Kleeman L, Jarvis R. Robust online map merging system using laser scan matching and omnidirectional vision. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 1. Taipei: IEEE; 2010. pp. 7-14

[10] Wang K, Jia S, Li Y, Li X, Guo B. Research on map merging for multi-robotic system based on RTM. In: 2012 IEEE International Conference on Information and Automation. Vol. 1. Shenyang, China: IEEE; 2012. pp. 156-161

[11] Besl P, McKay N. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1992;**14**(2): 239-256

[12] Han J, Yin P, He Y, Gu F. Enhanced ICP for the registration of large-scale 3D environment models: An experimental study. Sensors. 2016;**16**(2):228

[13] Carpin S. Fast and accurate map merging for multi-robot systems. Autonomous Robots. 2008;**25**(3): 305-316

[14] Saeed Gholami S, Magnusson M. 2D map alignment with region decomposition. Autonomous Robots. 2019;**43**(5):1117-1136

[15] Saeedi S, Trentini M, Seto M, Li H. Multiple-robot simultaneous localization and mapping: A review: Multiple-robot

simultaneous localization and mapping. Journal of Field Robotics. 2016;**33**(1):3-46

[16] Surmann H, Berninger N, Worst R. 3D mapping for multi hybrid robot cooperation. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2019;**1**: 626-633

[17] Yue Y, Yang C, Wang Y, Senarathne PGCN, Zhang J, Wen M, et al. A Multilevel fusion system for multirobot 3-D mapping using heterogeneous sensors. IEEE Systems Journal. 2019;**1**:1-12

[18] Drwięga M. 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). IEEE. pp. 663-668

[19] Jessup J, Givigi SN, Beaulieu A. 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: IEEE; 2014. pp. 3996-4001

[20] Hornung A, Wurm KM, Bennewitz M, Stachniss C, Burgard W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots. 2013;**34**(3):189-206

[21] Wurm KM, Hornung A, Bennewitz M, Stachniss C, Burgard W. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. ICRA; 2010. p. 8

[22] Bonanni TM, Corte BD, Grisetti G, et al. IEEE robotics and automation letters. 2017;**2**(2):1031-1038

[23] Magnusson M, Lilienthal A, Duckett T. Scan registration for autonomous mining vehicles using 3D-NDT. Journal of Field Robotics. 2007; **24**(10):803-827

[24] Magnusson M, Nuchter A, Lorken C, Lilienthal AJ, Hertzberg J. Evaluation of 3d registration reliability and speed - A comparison of ICP and NDT. In: 2009 IEEE International Conference on Robotics and Automation. Kobe, Japan: IEEE; 2009. pp. 3907-3912. Available from: https://ieeexplore.ieee.org/document/5152538

[25] Censi A, Carpin S. HSM3D: Feature-less global 6DOF scan-matching in the Hough/Radon domain. In: 2009 IEEE International Conference on Robotics and Automation. Kobe: IEEE; 2009. pp. 3899-3906

[26] Chen S, Nan L, Xia R, Zhao K, Wonka P. Plade: A plane-based descriptor for point cloud registration with small overlap. IEEE Transactions on Geoscience and Remote Sensing. 2020; **58**(4):2530-2540

[27] Drwięga M. 3D Map Server Software. Available from: https://github.com/mdrwiega/3d_map_server

[28] Zhong Y. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops. Kyoto, Japan: IEEE; 2009. pp. 689-696

[29] Tombari F, Salti S, Di Stefano L. A combined texture-shape descriptor for enhanced 3D feature matching. In: 2011 18th IEEE International Conference on Image Processing. Brussels, Belgium: IEEE; 2011. pp. 809-812

[30] Rusu RB, Blodow N, Beetz M. Fast point feature histograms (FPFH) for 3D registration. In: 2009 IEEE International

Conference on Robotics and Automation. Kobe: IEEE; 2009. pp. 3212-3217

[31] Chen H, Bhanu B. 3D Free-Form Object Recognition in Range Images Using Local Surface Patches. p. 4

[32] A. Hornung. OctoMap 3D Scan Dataset. Available from: http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/