

CoMapping: Efficient 3D-Map Sharing Methodology for Decentralized cases

Luis F. Contreras-Samamé, Salvador Domínguez-Quijada, Olivier Kermorgant and Philippe Martinet*
Laboratoire des Sciences du Numérique de Nantes (LS2N) - École Centrale de Nantes (ECN), 44300 Nantes, France
*INRIA Sophia Antipolis, 06902 Sophia Antipolis, France

Emails: *Luis.Contreras@ls2n.fr, Salvador.Dominguezquijada@ls2n.fr, Olivier.Kermorgant@ls2n.fr, Philippe.Martinet@inria.fr*

Abstract—CoMapping is a framework to efficient manage, share, and merge 3D map data between mobile robots. The main objective of this framework is to implement a Collaborative Mapping for outdoor environments where it can not use all the time GPS data. The framework structure is based on 2 stages. The first one, the Pre-Local Mapping Stage, each robot constructs in real-time a pre-local map of its environment using Laser Rangefinder data and low cost GPS information only in certain situations. Afterwards, in the Local Mapping Stage, the robots share their pre-local maps and merge them in a decentralized way in order to improve their new maps, renamed now as local maps. An experimental study for the case of decentralized cooperative 3D mapping is presented, where tests were conducted using 3 intelligent cars equipped with lidars and GPS receiver devices in urban outdoor scenarios. We also discuss the performance of all the cooperative system in terms of map alignments.

I. INTRODUCTION

Mapping challenge can be complex since in certain situations, e.g. for scenarios of large regions, it can require the usage of a group of robots that build the maps in a reasonable amount of time considering accuracy in the map construction [1]. So, a set of robots extends the capability of a single robot by merging measurements from group members, providing each robot with information beyond their individual sensors range. This allows a better usage of resources and executes tasks which are not feasible by a single robot. Multi-robot mapping is considered as a centralized approach when it requires all the data to be analysed and merged at a single computation unit. Otherwise, in a decentralized approach, each robot builds their local maps independent of one another and merge their maps upon rendezvous.



Fig. 1. Scheme of our CoMapping System considering a decentralized case

Figure 1 depicts the scheme of work proposed in this article for a group of robots where it was assumed that ZOE robot have direct exchange of data (as pose, size and limits of maps) with FLUENCE and GOLFCAR. And by contrast, FLUENCE

and GOLFCAR are in a scenario of non-direct communication, that is possible in cases where the robots have limited access conditions to a same environment, avoiding to define a meeting point for map sharing between these mobile units.

Following this scenario, this paper presents the development and validation of a new Cooperative Mapping framework (CoMapping) where:

- In the first stage named “Pre-Local Mapping”, each individual robot builds its map by processing range measurements from a 3D lidar moving in six degrees of freedom (6-DOF) and using low cost GPS data (GPS/GGA).
- For the second stage named “Local Mapping”, the robots send a certain part of their pre-local maps to the other robots based on our proposed Sharing algorithm. The registration process includes an intersecting technique of maps to accelerate processing

This decentralized system is deployed in an outdoor environment without continuous GPS service. Our proposal has been tested and validated in realistic situations. Results include maps developed with data acquired on the surroundings of the ECN (École Centrale Nantes) campus.

II. RELATED WORKS

In a scenario of cooperative mapping, robots first operate independently to generate individual maps. Here the registration method plays a fundamental role. Many registration applications use Lidar as a Rangefinder sensor for construction of maps [2]. However, a high lidar scan rate compared to its tracking can be harmful for this task, since it is possible the apparition of distortion in the map construction. For those cases, ICP [3] can be applied to match different scans. 2D and 3D lidar implementations with geometric structures matches of a generated local point group were presented in [4] [5]. Those methods use batch processing to build maps with accuracy and hence are not applicable to real-time map construction. In the first stage of our implementation we reconstruct maps as 3D pointclouds in real-time using 3-axis lidar by extraction and matching of geometric features in Cartesian space based in [6] initially. Then our system uses GPS position data to localize that cloud in a global frame.

Once all the maps have been placed in a global frame, they have to be merged together to form a global map. In this context, in [7] proposed a method for 3D merging of

occupancy grid maps based on octrees [8] for multi-robots. Simulation results were presented using Gazebo tool. Maps generated by each simulated robot are stored in files and finally merged offline. For the merging step, an accurate transformation between maps was assumed as known, nevertheless in real applications, that information (the transformation) is not accurate, since in many cases it is obtained by means of uncertain sensor observations that may not offer a reliable information. Contrary, we performed real experiments for a multi-robot application without supposed known the map transformation. Later, in [9] using a technique pre-merging, which consists in extract from of each map the subset of points included in the common region between maps bounding. Then, a centralized merging process refines the transformation estimate between maps by ICP registration [3] We use a variation of that method [9] but previously we include a efficient technique to exchange maps between robots in order to optimize bandwidth resources of multi-robot network.

On the other hand, other different solutions can be used in order to merge maps for a group of robots. For instance, cases with centralized approach, where the merging is computed on a unit or processing center once the entire environment has been explored by the vehicles, as is presented in [10], [9]. The other approach is the decentralized option, where map merging is executed in different units while traversing the environment, in which this approach considers a meeting point for the vehicles in order to exchange their maps and other data [11], [1], [12]. This last approach is experimentally studied in this paper.

III. METHODOLOGY

A. Pre-Local Mapping Stage

Each mobile robot executes a Pre-Local Mapping system using data provided by a LidarSLAM node. We just use GPS position to project the generated map on a global frame, in order to reduce project implementation costs, a beneficial cheap GPS service was used, specifically GPS/GGA(*Global Positioning System Fix Data*) at an accuracy of about 2 to 7 meters. Another advantage of our Pre-Local Mapping Stage is its versatile configuration, since it is not depend on a specific LidarSLAM method. A modified version of the LOAM technique¹ [6] was chosen as LidarSLAM method for this article because it currently ranks first in the KITTI evaluation table².

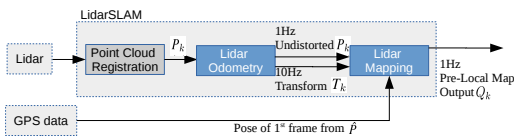


Fig. 2. Architecture of Pre-Local Mapping Stage

Figure 2 illustrates the block diagram of this stage, where \hat{P} is the raw point cloud data generated by a laser scan

¹LOAM: https://github.com/laboshin/loam_velodyne

²KITTI ranking: http://www.cvlibs.net/datasets/kitti/eval_odometry.php

in the beginning. For each sweep, \hat{P} is registered in the lidar coordinates $\{L\}$. The combined point cloud during each sweep k generates P_k . This P_k is processed by an algorithm named *Lidar Odometry*, which runs at a frequency around 10Hz and receives this point cloud and computes the lidar motion (transform T_k) between two consecutive sweeps. The distortion in P_k is corrected using the estimated lidar motion. The resulting undistorted P_k is processed at a frequency of 1Hz by an algorithm known as *Lidar Mapping*, which performs the matching and registration of the undistorted cloud onto a map. At last, using the GPS information of the vehicle pose during previous algorithm, it is possible to coarsely project the map of each robot into common coordinate frame for all the robots. This projected cloud is denoted as the Pre-Local Map.

1) *Lidar Odometry step*: The step begins with feature points extraction from the cloud P_k . The feature points are selected for sharp edges and planar surface patches. Let us define \bar{S} as the set of consecutive points i returned by the laser scanner in the same scan, where $i \in P_k$. A parameter proposed in [6] evaluates the smoothness of the local surface as following,

$$\bar{c} = \frac{1}{|\bar{S}| \cdot \|X_{(k,i)}^L\|} \left\| \sum_{j \in \bar{S}, j \neq i} (X_{(k,i)}^L - X_{(k,j)}^L) \right\|, \quad (1)$$

where $X_{(k,i)}^L$ and $X_{(k,j)}^L$ are the coordinates of two points from the set \bar{S} .

Moreover, a scan is split into four subregions to uniformly distribute the selected feature points within the environment. In each subregion is determined maximally 2 edge points and 4 planar points. The criteria to select the feature points as edge points is related to maximum \bar{c} values, and by contrast the planar points selection to minimum \bar{c} values. When a point is selected, it is thus mandatory that none of its surrounding points are already selected. Other conditions are: selected points on a surface patch can not be approximately parallel to the laser beam, or on boundary of an occluded region.

When the correspondences of the feature points are found based on the method proposed in [6], the distances from a feature point to its correspondence are calculated. Those distances are named as d_E and d_H for edge points and planar points respectively. The minimization of the overall distances of the feature points will allow to obtain the lidar odometry. That motion estimation is modelled with constant angular and linear velocities during a sweep.

Let us define E_{k+1} and H_{k+1} as the sets of edge points and planar points extracted from P_{k+1} , for a sweep $k+1$. The lidar motion relies on establishing a geometric relationship between an edge point in E_{k+1} and the corresponding edge line:

$$f_E(X_{(k+1,i)}^L, T_{k+1}^L) = d_E, i \in E_{k+1}, \quad (2)$$

where T_{k+1}^L is the lidar pose transform between the starting time of sweep $k+1$ and the current time t_i . T_{k+1}^L contains data about the sensor rigid motion in 6-DOF, $T_{k+1}^L =$

$[t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]^T$, wherein t_x , t_y , and t_z are translations along the axes x , y , and z from $\{L\}$, respectively, and θ_x , θ_y , and θ_z are rotation angles, following the right-hand rule.

Similarly, the relationship between an planar point in H_{k+1} and the corresponding planar patch is:

$$f_H(X_{(k+1,i)}^L, T_{k+1}^L) = d_H, i \in H_{k+1}, \quad (3)$$

Equations (2) and (3) can be reduced to a general case for each feature point in E_{k+1} and H_{k+1} , obtaining a nonlinear function, as:

$$f(T_{k+1}^L) = d, \quad (4)$$

in which each row of f is related to a feature point, and d possesses the corresponding distances. Levenberg-Marquardt method [13] is used to solve the Equation (4). Jacobian matrix (\mathbf{J}) of f with respect to T_{k+1}^L is computed. Then, the minimization of d through nonlinear iterations allows to solve the sensor motion estimation,

$$T_{k+1}^L \leftarrow T_{k+1}^L - (\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J}))^{-1} \mathbf{J}^T d, \quad (5)$$

where λ is the Levenberg-Marquardt gain.

Finally, the *Lidar Odometry* algorithm produces a pose transform T_{k+1}^L that contains the lidar tracking during the sweep between $[t_{k+1}, t_{k+2}]$ and simultaneously an undistorted point cloud \tilde{P}_{k+1} . Both outputs will be used by the Lidar Mapping step, explained in the next section.

2) *Lidar Mapping step*: This algorithm is used only once per sweep and runs at a lower frequency (1 Hz) than the Lidar Odometry step (10 Hz). The technique matches, registers and projects the cloud \tilde{P}_{k+1} provided by previous step (Lidar Odometry) as a map into the own coordinates system of a vehicle, defined as $\{V\}$. To understand the technique behaviour, let us defined Q_k as the point cloud accumulated until sweep k , and T_k^V as the sensor pose on the map at the end of sweep k , t_{k+1} . The algorithm extends T_k^V for one sweep from t_{k+1} to t_{k+2} , to get T_{k+1}^V , and projects \tilde{P}_{k+1} on the robot coordinates system $\{V\}$, denoted as \tilde{Q}_{k+1} . Then, by optimizing the lidar pose T_{k+1}^V , the matching of \tilde{Q}_{k+1} with Q_k is obtained.

In this step the feature points extraction and the finding feature points correspondences are calculated in the same way as in previous step (Lidar odometry), the difference just lies in that all points in \tilde{Q}_{k+1} share the time stamp, t_{k+2} .

In that context, nonlinear optimization is solved also by the Levenberg-Marquardt method [13], registering \tilde{Q}_{k+1} on the a new accumulated cloud map. To get a points uniform distribution, down-sampling process is performed to the cloud using a voxel grid filter [14] with a voxel size of 5 cm cubes.

Finally, since we have to work with multiple robots, we use a common coordinates system for their maps, $\{W\}$, coming from rough GPS position estimation of the 1st accumulated cloud frame Q_k .

B. Local Mapping Stage

In this section the Local Mapping is detailed, considering that the process is executed on the robot “i” with a shared map by robot “n” (see Figure 3).

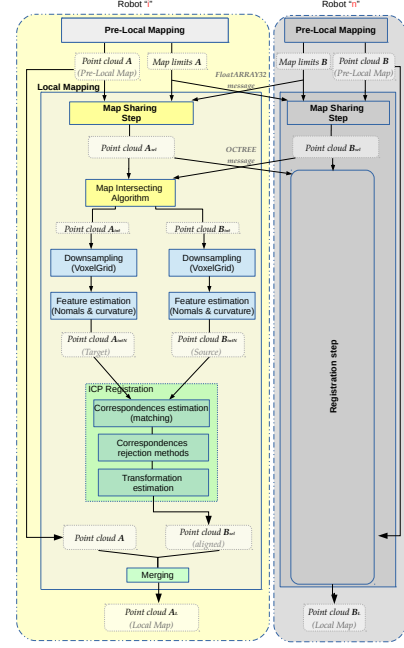


Fig. 3. Architecture of Local Mapping Stage for one robot “i”, receiving map data from another robot “n”.

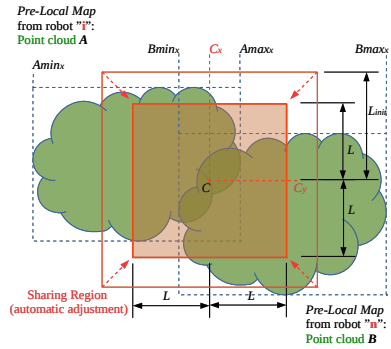


Fig. 4. Graphical representation of the Map Sharing technique (Top view of plane XY). $Amin_x$, $Amax_x$, $Bmin_x$ and $Bmax_x$ represent the point cloud limits along the x-axis.

1) *Map Sharing Step*: When the generation of Pre-Local Maps is done, the robots would have to exchange their maps to start the maps alignment process. In several cases the sharing and processing of maps of large dimensions can affect negatively the performance of the system with respect to runtime and memory usage. A sharing technique is presented in order to overcome this problem, in which each vehicle builds only sends a certain part of its map to the other robots. When the maps are ready for transferring, they are compressed in octree format using OctoMap library [8] in order to optimize the robot-communication.

The proposed sharing technique is based on the method developed in [15]. Figure 4 depicts the behaviour, wherein point clouds A and B represent the Pre-Local Maps from two robots “i” and “n” respectively. In each robot the algorithm first receives only information about the 3D limits of the maps (i.e. bounding cubic lattice of the point clouds) and then decides what part of its map will be shared to the other robot. These limits were determined previously using the function $GetBounds()$ that returns two vectors: in the first one $Amin$, their components represent the lowest displacement from the origin along each axis in the point cloud; and the other vector $Amax$ is related to the point of the highest displacement.

```

Data: Point Cloud  $A$ ; Limits: Vectors  $Amin$ ,  $Amax$ ,
         $Bmin$  and  $Bmax$ ; Parameters: Scalars  $L_{init}$ ,
         $L_{step}$ ,  $Np_{max}$ 
Result: Point Cloud  $A_{sel}$ 
begin
   $A_{sel} \leftarrow \emptyset$ ;
   $C_x = 0$ ;  $C_y = 0$ ;  $C_z = 0$ ;
   $(V2_x, V3_x) =$ 
   $GetValues(Amin_x, Amax_x, Bmin_x, Bmax_x)$ ;
   $(V2_y, V3_y) =$ 
   $GetValues(Amin_y, Amax_y, Bmin_y, Bmax_y)$ ;
   $(V2_z, V3_z) =$ 
   $GetValues(Amin_z, Amax_z, Bmin_z, Bmax_z)$ ;
   $C_x = (V2_x + V3_x)/2$ ;
   $C_y = (V2_y + V3_y)/2$ ;
   $C_z = (V2_z + V3_z)/2$ ;
   $Np = PointSize(A)$ ;
  for ( $L=L_{init}$  ;  $Np > Np_{max}$  ;  $L = L - L_{step}$ ) do
     $Smin_x = C_x - L$  ;  $Smax_x = C_x + L$ ;
     $Smin_y = C_y - L$  ;  $Smax_y = C_y + L$ ;
     $Smin_z = C_z - L$  ;  $Smax_z = C_z + L$ ;
    foreach  $a \in A$  do ;
    if  $Smin_x < a_x < Smax_x$  and  $Smin_y < a_y <$ 
     $Smax_y$  and  $Smin_z < a_z < Smax_z$  then
      |  $A_{sel} = A_{sel} + a$ ;
    end
     $Np = PointSize(A_{sel})$ ;
  end
end

```

Algorithm 1: Selection of Point Cloud to share with another robot.

Pseudo-code of the map sharing step is described in Algorithm 1. Inside the code, the function $GetValues()$ sorts in ascending order the array of components along each axis of the vectors $Amin$, $Amax$, $Bmin$, $Bmax$ and returns the 2nd and 3rd values from this sorted array, denoted $(V2)$ and $(V3)$ respectively. Next, for each axis, the average of the two values obtained by the function $GetValues()$ is used in order to determine the Cartesian coordinates (C_x, C_y, C_z) of the geometric center of the sharing region (S) . Actually, this map sharing region is a cube whose edge length $2L$ is determined iteratively. Points from A contained in this cube region are extracted to generate a new point cloud A_{sel} . In each iteration the cube region is reduced until the number of points from A_{sel} is smaller than the manual parameter

Np_{max} , which represents the number of points maximum that the user wants to exchange between robots. Once the loop ends, A_{sel} is sent to the other robot. Similarly on the other robotic platform “n”, the points from B included in this region are also extracted to obtain and share B_{sel} with the another robot “i”. Then, it is worth to remind, the clouds A_{sel} and B_{sel} are encoded and sent in octree format to reduce the usage of bandwidth resources of the multi-robot network. Then maps are decoded and reconverted in 3D point cloud format to be used in the next Registration step. Pointcloud-octree encoding and decoding were realized using ROS nodes supported on OctoMap library [8].

2) *Registration Step:* The intersecting volumes of the two maps A_{sel} and B_{sel} are computed and denoted as A_{int} and B_{int} , obtained from the exchanged map bounds [9]. In order to improve the computation speed, point clouds A_{int} to B_{int} first go through a down-sampling process to reduce the number of points in the alignment of our clouds. Feature descriptors as surface normals and curvature are used to improve the matching, which is the most expensive stage of the registration algorithm [16]. These generated normal-point clouds A_{intN} and B_{intN} are then used by Iterative Closest Point (ICP) algorithm [17]. This method refines an initial alignment between clouds, which basically consists in estimating the best transformation to align a source cloud B_{intN} to a target cloud A_{intN} by iterative minimization of an error metric function. At each iteration, the algorithm determines the corresponding pairs (b', a') , which are the points from A_{intN} and B_{intN} respectively, with the least Euclidean distance.

Then, least squares registration is computed and the mean squared distance E is minimized with regards to estimated translation t and rotation R :

$$E(R, t) = \frac{1}{N_{pb'}} \sum_{i=1}^{N_{pb'}} \| a'_i - (R b'_i + t) \|^2, \quad (6)$$

where $N_{pb'}$ is the number of points b' .

The resulting rotation matrix and translation vector can be express in a homogeneous coordinates representation (4×4 transformation matrix T_j) and are applied to B_{intN} . The algorithm then re-computes matches between points from A_{intN} and B_{intN} , until the variation of mean square error between iterations is less than an defined threshold. The final ICP refinement for n iterations can be obtained by multiplying the individual transformations: $T_{ICP} = \prod_{j=1}^n T_j$. Finally the transformation T_{ICP} is applied to the point cloud B_{sel} to align and merge with the original point cloud A , generating the Local Map A_L then. Each robot thus performed its own merging according to limited data shared from other agents within communication range.

IV. RESULTS

In this section we show results validating the presented concepts and the functionality of our system. As we consider ground vehicles, the ENU (East-North-Up) coordinate system is used as external reference of the world frame $\{W\}$, where



Fig. 5. Vehicles used in the tests: ZOE, FLUENCE and GOLFCAR.

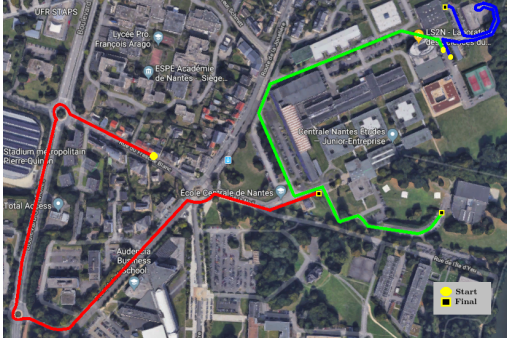


Fig. 6. Paths followed by ZOE (green one), FLUENCE (red one) and GOLFCAR robot (blue one) during experiments. Image source: Google Earth.

y -axis corresponds to North and x -axis corresponds to East, but coinciding its origin with the GPS coordinates [Longitude: -1.547963; Latitude: 47.250229].

In this article, our proposed framework was validated considering three vehicles for experiments, a *ZOE Renault*, a *FLUENCE Renault* and a *GOLFCAR* (see Figure 5) customized and equipped with a *Velodyne VLP-16* 3D lidar, with 360° horizontal field of view and a 30° vertical field of view. All data come from the campus outdoor environment in an area of approximately 1000m x 700m. The vehicles traversed that environment following different paths and collected sensor observations about the world, running pre-local mapping process in real-time.

For the validation, the vehicles build clouds from different paths (see Figure 6). Results of the Pre-Local Mapping of this experiment are shown in Figure 7.

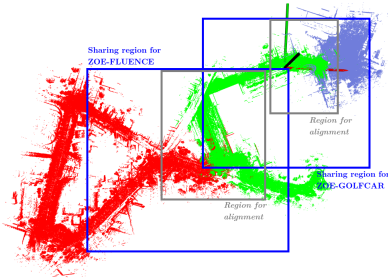


Fig. 7. Top view of unaligned Pre-Local Maps generated by ZOE (green one), FLUENCE (red one) and GOLFCAR robot (blue one) projected on common coordinate system

Figure 7 also depicts the “sharing region” determined during the map exchange process in each robot. It was assumed that all the vehicles have the constraint of exchanging the number of points maximum Np_{max} of 410000 to simulate restrictions

in resources of bandwidth network or memory usage in robots. The tests were divided in two. In the first one, test A, ZOE and FLUENCE car define a meeting point to transfer their maps. Once, ZOE car exchanges and updates its local map, a new point of rendezvous for map sharing is determined by ZOE and GOLFCAR in the following test B.

Since we study a decentralized case, then each robot performs a relative registration process considering its Pre-Local map as target cloud for alignment reference. The systems of each robot executes the intersecting algorithm and then an ICP refinement to obtain an improved transform between each map. Figures 8 and 9 depict the intersection between the shared point clouds during the alignment process in each robot. In the yellow box the alignment is more appreciated. Once the refined transformation is obtained, it is then applied to the shared map.

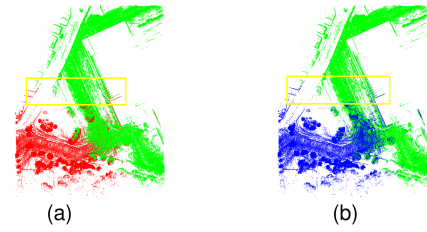


Fig. 8. Test A: Alignment of the intersecting regions with ICP refinement performed in ZOE robot, when it received the FLUENCE map (a) Green and red maps represent the target and source clouds pre ICP, top view (b) Green and blue maps represent the target and aligned source clouds post ICP, top view.

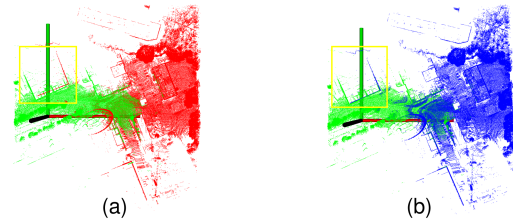


Fig. 9. Test B: Alignment of the intersecting regions with ICP refinement performed in ZOE robot, when it received the GOLFCAR map (a) Green and red maps represent the target and source clouds pre ICP, top view (b) Green and blue maps represent the target and aligned source clouds post ICP, top view.

Quantitative alignment results of the ICP transformation relatives to each robot are shown in Tables I and II. All the ICP transformations are expressed in Euler representation $(x, y, z, roll, pitch, yaw)$ in meters and radians. For instance, first row of Table I corresponds to the merging process in ZOE, when this robot received the map shared by FLUENCE and it aligned that map to its own pre-local map. The decentralized system demonstrated alignments in opposite directions for both robots, since we have to consider that each robot performs the merging process considering its Pre-Local map as target cloud for alignment reference. For instance, on Table II for ZOE vehicle the algorithm converged to the value of displacement of -0.1782 m and -3.2605 m along the x -axis and

y-axis respectively. On the other hand on the GOLFCAR robot, the algorithm converged to a value of displacement of 0.2213 m and 3.3857 m along the x-axis and y-axis respectively, reconfirming relative alignments in opposite directions.

TABLE I

TEST A: RELATIVE ICP TRANSFORMATIONS IN EULER FORMAT BETWEEN ZOE AND FLUENCE ROBOT

Robot	x	y	z	roll	pitch	yaw
ZOE	-1.6517	3.0966	-9.9729	0.0132	0.0730	0.0022
FLU.	4.5748	-4.4556	6.6061	-0.0054	-0.0624	-0.0084

TABLE II

TEST B: RELATIVE ICP TRANSFORMATIONS IN EULER FORMAT BETWEEN ZOE AND GOLFCAR ROBOT

Robot	x	y	z	roll	pitch	yaw
ZOE	-0.1782	-3.2605	1.7771	-0.0516	0.0115	0.0356
GOL.	0.2213	3.3857	-2.6070	0.0411	-0.0256	-0.0380

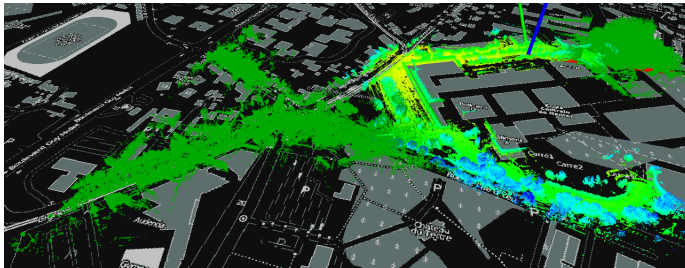


Fig. 10. Final 3D Local Map of ZOE robot

Figure 10 shows one of the merging results corresponding to the ZOE robot, in which the cloud represents the final 3D local map projected on a 2D map in order to make qualitative comparisons. Experiments showed the impact of working with intersecting regions, since it can accelerate the alignment process by decreasing the number of points to compute. In the same way, tests demonstrated that our proposed map sharing technique developed a transcendental position in the performance of the entire mapping collaborative system by reducing the map size to transmit. Finally, the sharing algorithm remains a suitable candidate to exchange efficiently maps between robots considering the use of clouds of large dimensions.

V. CONCLUSION AND FUTURE WORK

A framework, CoMapping, was presented for decentralized 3D mapping system for multiple robots. The work has showed that maps from different robots can be successfully merged, from a coarse initial registration and a suitable exchange of data volume. The system uses initially range measurements from a 3D lidar, generating a pre local maps for each robot. The complete system solves the mapping problem in an efficient and versatile way that can run in computers dedicated to three vehicles for experiments, leading to merged maps independently on each vehicle for GPS-denied environments all the time. Future work will focus on the analysis of maps

alignment in decentralized cases, studying the direct impacts on the consistence of maps generated by each robot.

ACKNOWLEDGMENT

This article is based upon work supported by the Erasmus Mundus Action 2 programme through the Sustain-T Project, as well as the institutions ECN (École Centrale de Nantes) and LS2N (Laboratoire des Sciences du Numérique de Nantes). Parts of the equipments used here were funded by the project ROBOTEX, reference ANR-10-EQPX-44-01.

REFERENCES

- [1] P. Dinnissen, S. N. Givigi, and H. M. Schwartz, "Map merging of multi-robot SLAM using reinforcement learning." in *SMC*. IEEE, 2012. [Online]. Available: <http://dblp.uni-trier.de/db/conf/smc/smc2012.html#DinnissenGS12>
- [2] S. Kohlbrecher, O. V. Stryk, T. U. Darmstadt, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *International Symposium on Safety, Security, and Rescue Robotics*. IEEE, 2011.
- [3] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets - open-source library and experimental protocol." *Auton. Robots*, 2013. [Online]. Available: <http://dblp.uni-trier.de/db/journals/arobots/arobots34.html#PomerleauCSM13>
- [4] R. Zlot and M. Bosse, "Efficient Large-Scale 3D Mobile Mapping and Surface Reconstruction of an Underground Mine." in *FSR*, ser. Springer Tracts in Advanced Robotics, K. Yoshida and S. Tadokoro, Eds. Springer, 2012. [Online]. Available: <http://dblp.uni-trier.de/db/conf/fsr/fsr2012.html#ZlotB12>
- [5] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a Spring-Mounted 3-D Range Sensor with Application to Mobile Mapping," *IEEE Transactions on Robotics*, Oct 2012.
- [6] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.
- [7] J. Jessup, S. N. Givigi, and A. Beaulieu, "Merging of octree based 3D occupancy grid maps," in *2014 IEEE International Systems Conference Proceedings*, March 2014.
- [8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Auton. Robots*, Apr. 2013. [Online]. Available: <http://dx.doi.org/10.1007/s10514-012-9321-0>
- [9] J. Jessup, S. N. Givigi, and A. Beaulieu, "Robust and Efficient Multi-robot 3-D Mapping Merging With Octree-Based Occupancy Grids," *IEEE Systems Journal*, 2015.
- [10] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE*, July 2006.
- [11] N. E. Özküçür and H. L. Akin, "Supervised feature type selection for topological mapping in indoor environments," in *21st Signal Processing and Communications Applications Conference, SIU 2013, Haspolat, Turkey, April 24-26, 2013*, 2013, pp. 1–4. [Online]. Available: <http://dx.doi.org/10.1109/SIU.2013.6531556>
- [12] J. Zhang and S. Singh, "Aerial and Ground-based Collaborative Mapping: An Experimental Study," in *The 11th Intl. Conf. on Field and Service Robotics (FSR)*, Sep 2017.
- [13] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.
- [14] R. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011.
- [15] L. Contreras, O. Kermorgant, and P. Martinet, "Efficient Decentralized Collaborative Mapping for Outdoor Environments," in *2018 IEEE International Conference on Robotic Computing (IRC)*, Laguna Hills, United States, Jan 2018 - In Press.
- [16] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, Jun. 2001.
- [17] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, Feb. 1992. [Online]. Available: <http://dx.doi.org/10.1109/34.121791>