Efficient Decentralized Collaborative Mapping for Outdoor Environments

Luis Contreras
Laboratoire des Sciences du Numérique de Nantes (LS2N)
École Centrale de Nantes (ECN)
44300 Nantes, France
Email: *Luis.Contreras@ls2n.fr*

Olivier Kermorgant
Laboratoire des Sciences du Numérique de Nantes (LS2N)
École Centrale de Nantes (ECN)
44300 Nantes, France
Email: *Olivier.Kermorgant@ls2n.fr*

Philippe Martinet
INRIA Sophia Antipolis
06902 Sophia Antipolis, France
Laboratoire des Sciences du Numérique de Nantes (LS2N)
École Centrale de Nantes (ECN)
44300 Nantes, France
Email: *Philippe.Martinet@inria.fr*

***Abstract*—An efficient mapping in mobile robotics may involve the participation of several agents. In this context, this article presents a framework for collaborative mapping applied to outdoor environments considering a decentralized approach. The mapping approach uses range measurements from a 3D lidar moving in six degrees of freedom. For that case, each robot performs a local SLAM. The maps are then merged when communication is available between the mobile units. This allows building a global map and to improve the state estimation of each agent. Experimental results are presented, where partial maps of the same environment are aligned and merged coherently in spite of the noise from the lidar measurement.**

I. Introduction

In the area of mobile robots, many tasks are related with exploration and re-construction of the environments. These tasks involve problems as Simultaneous Localization and Mapping (SLAM), in which the mobile robot is placed in an unknown environment and must estimate the map of its surroundings as well as its pose (position and orientation) relative to the map. This is a complex task since determining the robot's localization typically requires prior knowledge of its surroundings, and on the other hand, map building of its surroundings requires knowledge of the robot's localization. In order to build a map, the robot has to traverse the unknown environment and incrementally collect measurements of its surroundings and new pose. Furthermore, the presence of uncertainty and noise in the measurements from the sensors, produces accumulate errors over time that consequently distort the robots estimate of its position and map.

The mapping of a large area may require the use 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, each robot builds their local maps independent of one another and merge their maps upon rendezvous.

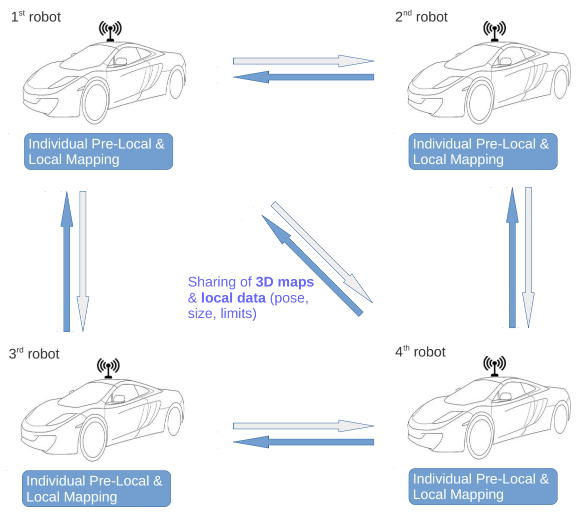

1st robot
2nd robot
3rd robot
4th robot
Individual Pre-Local & Local Mapping
Individual Pre-Local & Local Mapping
Individual Pre-Local & Local Mapping
Individual Pre-Local & Local Mapping
Sharing of 3D maps & local data (pose, size, limits)

Fig. 1. Scheme our Collaborative Decentralized Mapping System.

Figure 1 shows the scheme of work proposed in this article for a group of robots where most of them have a direct exchange of data (as pose, size and limits of maps, and other kind of information); and by contrast another cluster of them (2nd and 3rd robot) can be present in a scenario of indirect communication due for instance to access conditions to a same environment that avoid defining a meeting point between these mobile units. For that case, we propose a collaborative decentralized mapping framework for efficient merging of 3D maps where:

- In the first stage named "Pre-Local Mapping", each individual robot builds its map and sends a certain part of it to the other robots based on our proposed Sharing algorithm. Low-cost GPS data is used for the first alignment between maps represented in a global frame.

- For the second stage named “Local Mapping”, the registration process considers the first alignment previously mentioned and includes an intersecting technique of maps to reduce runtime.

This article is organized as follows: Section II presents the current works developed in this field. We then detail our approach in Section III. Finally, experimental results are presented and analysed in Section IV, from data acquired on the campus of École Centrale de Nantes.

II. Related works

One of the topic to consider in mapping is the type of sensor used in the robot navigation. In this context, lidar has become a useful range sensor for these applications [2] [3]. Nonetheless, when the lidar scan rate is higher than its tracking, distortion may occur in the map building. In this case, standard registration methods as ICP [4] can be applied to match laser returns for different scans.

Bosse and Zlot [5] [6] [7] use a 2-axis lidar and matches geometric structures of a set local point generated by the lidar in order to build a point cloud [7]. The mapping system includes an IMU and uses loop closure to reconstruct large maps. This approach is based on batch processing to build maps with accuracy and hence is not applicable to real-time mapping.

In the proposed approach, we consider a Pre-Local Mapping stage, which extracts and matches geometric features in Cartesian space based on [8]. Our system uses these pre-local clouds inside a more global (at least 2 maps) registration process, ensuring both computation speed and accuracy.

The map representation is crucial when merging and communication are considered. Occupancy grids and feature maps are popular approaches in this domain [9]. In this context, in [10], the authors presented method for 3D merging of occupancy grid maps based on octrees [11], [12], [13], [14], [15], [16]. The corresponding experiments consisted in using two simulated robots that traverse an simulated environment and take sensor observations about the scenario for the maps construction. These maps are stored in files and finally merged offline. Furthermore, an accurate transformation between maps was supposed being known, however in real applications, the transform matrix between two robots map is usually only coarsely estimation from sensor observations. This work was extended in [17], where the subset of points included in the common region is extracted prior to the merge. Then, the merging process refines the transformation estimate between maps by ICP registration [4]

In this paper, we assume provisionally an environment representation of 3D point clouds format during the process in order to validate the initial proposed framework.

We also consider a technique to exchange maps between robots while optimizing the registration step. The overall method is part of the decentralized paradigm (used for instance in [18]), where map merging is executed in different units while traversing the environment. In order to ensure communication range, we consider a rendezvous point for the vehicles for exchanging their maps and other data.

We now present an overview of our contribution.

III. Methodology

In this section the general overview of the approach is first described. We then focus on each of the stages leading to merged maps for each vehicles according to their communication capabilities.

A. Overview

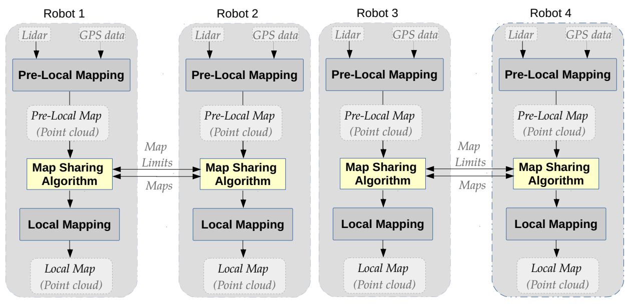


Fig. 2. Final structure for a team of robots.

Figure 2 depicts the final idea of our collaborative mapping system applied on a group of robots. Each mobile robot executes the mapping task using its on-board lidar sensor and computer. In the beginning, each vehicle independently estimates the 3D-map of the environment using a variant of LOAM (*Lidar Odometry And Mapping*) technique [8], [19]. This method constructs an environment map represented in the respective local coordinate systems of each vehicle using range measurements from a 3D lidar moving in 6-DOF. Then all the robots use a common reference frame for their maps, where the absolute pose information for each robot is provided by GPS prior to the map construction. All this first stage is denoted as Pre-Local Mapping and those initial maps generated in point cloud format are indicated as Pre-Local maps.

Afterwards all the vehicles send information about their Pre-Local maps (as bounding cubic lattice of the point cloud) to a Sharing Algorithm, which processes that data for then deciding what part of the Pre-Local map is sent to another robot.

For the next stage, indicated as the Local Mapping, it is necessary for robots to merge the resultant Pre-Local maps into one large and new Local map. During this registration process, a Map Intersecting Algorithm is executed to extract the intersecting volumes from each map. Finally, a refinement for maps alignment is performed by an Iterative Closest Point (ICP) algorithm, which leads to a consistent local map.

We now detail each mentioned stages.

B. Pre-Local Mapping Stage

As previously mentioned, each mobile robot executes a Pre-Local Mapping system based on the LOAM method [8] using as devices: a computer, a lidar sensor and a GPS receiver (specifically GGA type: *Global Positioning System Fix Data*).

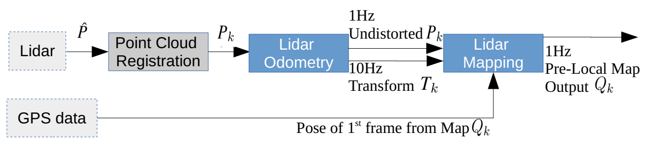


Fig. 3. Architecture of Pre-Local Mapping Stage based on [8].

Figure 3 illustrates the block diagram of this stage, where $\hat{P}$ is the generated point cloud. 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 10 Hz and receives this point cloud and computes the lidar motion (pose transform T_k) between two consecutive sweeps.The distortion in P_k is corrected using the estimated lidar motion. The resultant undistorted P_k is processed at a frequency of 1 Hz by performing the matching and registration of the undistorted cloud onto a map. Finally, using the GPS information of the vehicle pose during, it is possible to coarsely project the map of each robot into a common coordinate frame for all the robots. This projected cloud is denoted as the Pre-Local Map.

We now detail the lidar odometry and mapping scheme that is used to build the Pre-Local map of each robot.

1) Lidar Odometry: Lidar odometry is based on feature points extraction from the point cloud. The feature points are selected for sharp edges and planar surface patches. For that, we consider that $\overline{S}$ is the set of consecutive points i returned by the laser scanner in the same scan, where $i \in P_k$. An indicator to evaluate the smoothness of the local surface is defined as:

$$\bar{c} = \frac{1}{\mid \overline{S} \mid . \parallel X^L_{(k,i)} \parallel} \parallel \sum_{j \in S, j \neq i} (X^L_{(k,i)} - X^L_{(k,j)}) \parallel, \qquad (1)$$

where $X^L_{(k,i)}$ and $X^L_{(k,j)}$ are the coordinates of two points from the set $\overline{S}$.

Moreover, a scan is split into four subregions to uniformly distribute the selected feature points within the environment. 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 point are already selected, the other conditions being that 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 [8], then the distances from a feature point to its correspondence are computed. The minimization of the overall distances of the feature points allows to obtain the so-called 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 odometry relies on establishing a geometric relationship between an edge point in E_{k+1} and the corresponding edge line:

$$f_E(X^L_{(k+1,i)}, T^L_{k+1}) = d_E, i \in E_{k+1}, \qquad (2)$$

where T^L_{k+1} is the lidar pose transform between the starting time of sweep $k+1$ and the current time t_i. T^L_{k+1} contains information about the sensor rigid motion in 6-DOF, $T^L_{k+1} = [t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]^T$, in which 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:

$$f_H(X^L_{(k+1,i)}, T^L_{k+1}) = d_H, i \in H_{k+1}, \qquad (3)$$

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:

$$\boldsymbol{f}(T^L_{k+1}) = \boldsymbol{d}, \qquad (4)$$

in which each row of $\boldsymbol{f}$ is related to a feature point, and $\boldsymbol{d}$ possesses the corresponding distances. The Levenberg-Marquardt method [20] is used to solve Equation (4). For that case, the Jacobian matrix ($\mathbf{J}$) of $\boldsymbol{f}$ with respect to T^L_{k+1} is computed. Then, the minimization of $\boldsymbol{d}$ through nonlinear iterations allows to solve the sensor motion estimation,

$$T^L_{k+1} \longleftarrow T^L_{k+1} - (\mathbf{J}^T\mathbf{J} + \lambda \mathtt{diag}(\mathbf{J}^T\mathbf{J}))^{-1}\mathbf{J}^T\boldsymbol{d}, \qquad (5)$$

where λ is the Levenberg-Marquardt gain.

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

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

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

In that case, the nonlinear optimization is solved also by the Levenberg-Marquardt method, registering $\bar{Q}_{k+1}$ on the a new accumulated map. To get a points uniform distribution, down-sampling process is performed to the map cloud using a voxel grid filter [21] with a voxel size of 5cm 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.

We now present the map sharing step, happening when two robots are in communication range.

C. Map Sharing

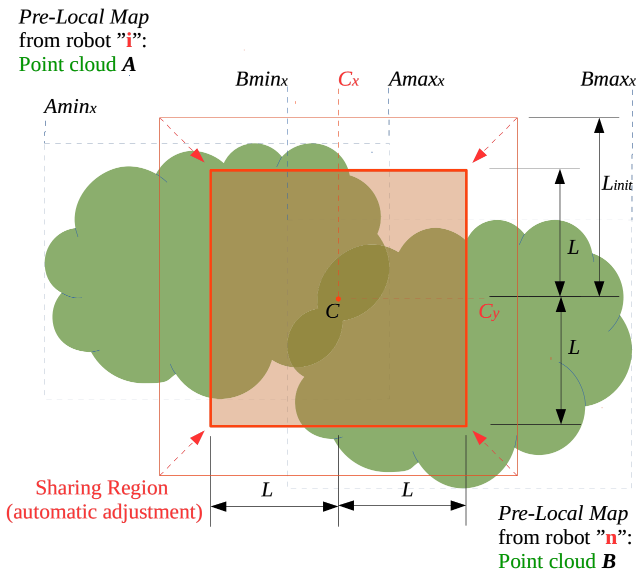


Fig. 4. Graphical representation of the Map Sharing technique (Top view of plane XY). $Amin_x$, $Amax_x$, $Bmin_x$ and $Bmax_x$ represent the point cloud limits along the x-axis.

After the generation of Pre-Local Maps, 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 limit 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.

Figure 4 depicts the behaviour of the proposed method, in which point clouds A and B represent the Pre-Local Maps from two different 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, their components represent the lowest displacement from the origin along each axis in the point cloud; and the other vector is related to the point of the highest displacement.

Algorithm 1 presents the pseudo-code of the map sharing step. 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

```
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.

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).

This map sharing region is a cube whose edge length $2L$ is determined iteratively. Actually, the 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".

We now focus on the Local Mapping stage, that is the map merging on a given mobile unit.

D. 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 coming from robot "n" (see Figure 5).

1) Preparation for Registration step: The computed intersecting volumes of the two maps A_{sel} and B_{sel} are denoted as A_{int} and B_{int} and can be obtained from the exchanged map bounds [17] (see Algorithm 2). In order to improve the computation speed, point clouds A_{int} to B_{int} first go through a down-sampling process in order to reduce the number of points to align of our clouds. The chosen voxel size is 3.5 m cubes in our experimentation. The next step is the feature descriptors estimation, where are computed the surface normals and curvature of the input clouds. This information highly improves the feature points matching, which is the most expensive stage of the ICP algorithm [22]. The normal-point clouds generated after this step are denominated as A_{intN} and B_{intN}. Those normal point clouds are then used and aligned in the next step.

```
Data: Point Clouds $B_{sel}$ and $A_{sel}$
Result: Point Clouds $B_{int}$ and $A_{int}$
begin
    $A_{int} \leftarrow \emptyset$ ; $B_{int} \leftarrow \emptyset$;
    $(Amin, Amax) = GetBounds(A_{sel})$;
    foreach $\boldsymbol{b} \in B_{sel}$ do ;
    if $Amin_x < b_x < Amax_x$ and $Amin_y < b_y < Amax_y$ and $Amin_z < b_z < Amax_z$ then
        $B_{int} = B_{int} + \boldsymbol{b}$;
    end
    $(Bmin, Bmax) = GetBounds(B_{sel})$;
    foreach $\boldsymbol{a} \in A_{sel}$ do ;
    if $Bmin_x < a_x < Bmax_x$ and $Bmin_y < a_y < Bmax_y$ and $Bmin_z < a_z < Bmax_z$ then
        $A_{int} = A_{int} + \boldsymbol{a}$;
    end
end
```

Algorithm 2: Map Intersecting Algorithm based on [17].

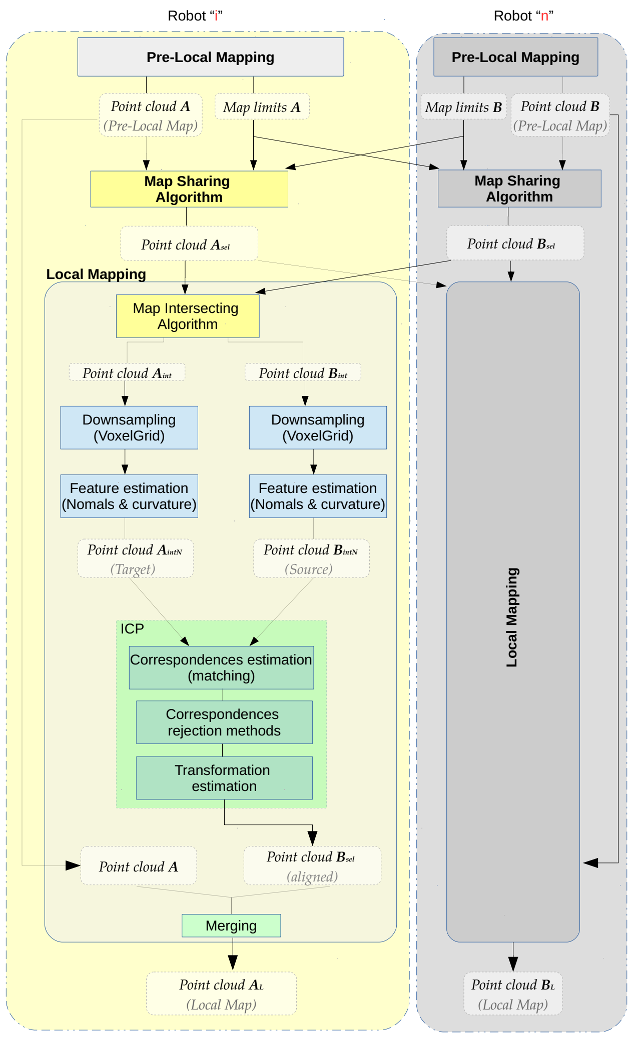
2) Registration step with ICP: Environment data used in this work are aligned with Iterative Closest Point (ICP) algorithm [23]. The algorithm refines an initial alignment between clouds, which basically consists in estimating the best transformation to align a source cloud B_{intN} to a target cloud 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 $\boldsymbol{t}$ and rotation R:

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

where $Np_{b'}$ is the number of points ***b'***.

The resultant 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.



Robot "i"
Robot "n"
Pre-Local Mapping
Pre-Local Mapping
Point cloud A (Pre-Local Map)
Map limits A
Map limits B
Point cloud B (Pre-Local Map)
Map Sharing Algorithm
Map Sharing Algorithm
Point cloud A_sel
Point cloud B_sel
Local Mapping
Map Intersecting Algorithm
Point cloud A_int
Point cloud B_int
Downsampling (VoxelGrid)
Downsampling (VoxelGrid)
Feature estimation (Nomals & curvature)
Feature estimation (Nomals & curvature)
Point cloud A_intN (Target)
Point cloud B_intN (Source)
ICP
Correspondences estimation (matching)
Correspondences rejection methods
Transformation estimation
Local Mapping
Point cloud A
Point cloud B_sel (aligned)
Merging
Point cloud A_L (Local Map)
Point cloud B_L (Local Map)

Fig. 5. Architecture of Local Mapping Stage for one robot.

We now present the experimental results to illustrate the proposed method.

IV. Results

In this section we present 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. 6. Vehicles used during the tests.

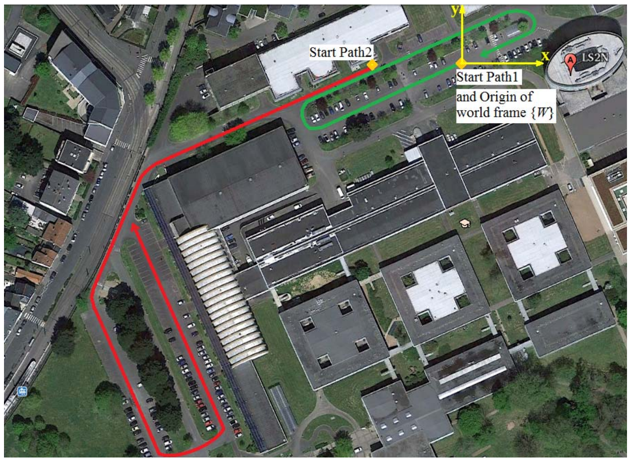


Fig. 7. Paths followed by 1st robot (green one) and 2nd robot (red one) during the test. Image source: Google Earth.

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, by the moment, our proposed framework was validated considering initially two vehicles for experiments, a Renault Fluence and a Renault Zoe (see Figure 6) 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 290m x 170m. The vehicles traversed that environment following different paths and collected sensor observations about the world, running the real-time mapping process on two laptops Core-i5 independently for each vehicle.

In this experiment the vehicles build clouds from different paths (see Figure 7). The results of the Pre-Local Mapping of this experiment are shown in Figure 8. The map of the first robot is shown in green, and in red for the second robot. Figure 8 also depicts the "sharing region" and the "intersecting region" determined during the alignment process in each robot. For that test, the number of points maximum to exchange between robots, Np_{max}, was set to 156000 points.

Once the Sharing step ends, the systems of each robot performs the intersecting algorithm and then an ICP refinement

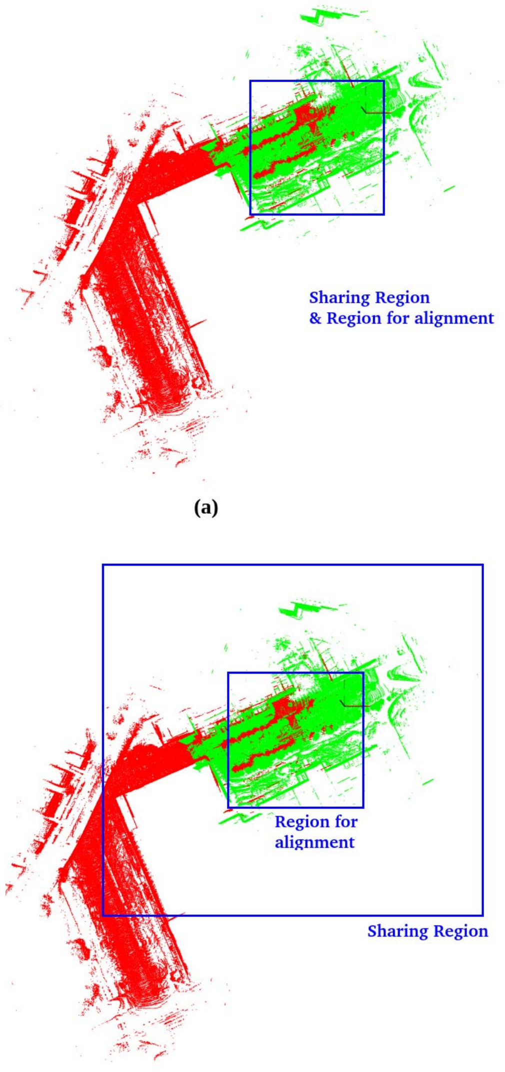
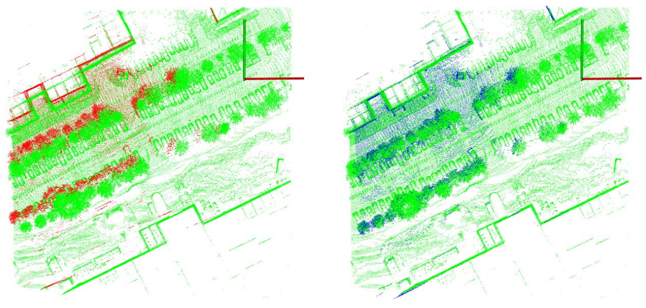


Fig. 8. Top view of Pre-Local Maps for 1st robot (green one) and 2nd robot (red one) projected in common coordinates system prior to ICP refinement. (a) Sharing and alignment region for the 1st robot; (b) for the second one.

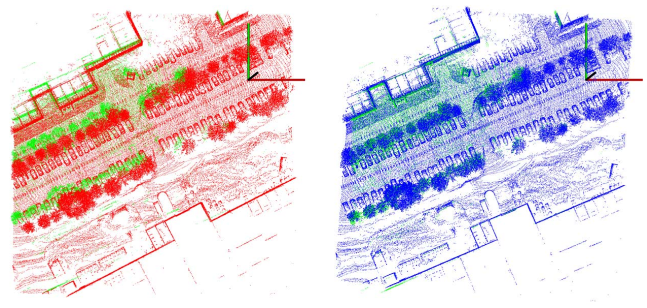
to obtain an improved transform between each map. Figures 9 and 10 depict the intersection of the point clouds during the alignment process in each robot. Once the refined transformation is obtained, it is then applied to the shared map.

Quantitative alignment results of the ICP are shown in Table I. On the 1st robot the algorithm converged to the value of displacement of 0.693 m and 1.572 m along the x-axis and y-axis respectively. On the other hand on the 2nd robot, the algorithm converged to a value of displacement of -1.110 m and -1.557 m along the x-axis and y-axis respectively. Those results reconfirm the alignments on opposite directions for both robots, since we must consider that each robot performs a relative registration process considering its Pre-Local map as target cloud for alignment reference.



(a) (b)

Fig. 9. Alignment of maps with ICP refinement in 1st robot (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.



(a) (b)

Fig. 10. Alignment of maps with ICP refinement in 2nd robot (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.

Figure 11 shows the results of the final merging in the robots, in which the green cloud represents the target Pre-Local map generated by each robot and the red cloud corresponds the Pre-Local map shared by the other mobile unit but finally aligned to the target cloud. The fusion of these maps generates the Local Maps in both platforms.

The experiments showed also the importance of selection of maps intersecting region for the alignment, avoiding the divergence of the registration algorithm. In the same direction, 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 vehicles considering the use of maps of the large dimensions.

TABLE I
REFINEMENT TRANSFORMATION MATRICES

1st Robot	2nd Robot
Rotation-matrix (T_{ICP}) : $\begin{bmatrix} 0.997 & -0.065 & 0.036 \\ 0.065 & 0.998 & 0.007 \\ -0.037 & -0.005 & 0.999 \end{bmatrix}$	Rotation-matrix (T_{ICP}) : $\begin{bmatrix} 0.996 & 0.068 & -0.049 \\ -0.068 & 0.998 & 0.005 \\ 0.049 & -0.002 & 0.999 \end{bmatrix}$
Translation-vector (T_{ICP}) : $[0.693 \quad 1.572 \quad -2.149]$	Translation-vector (T_{ICP}) : $[-1.110 \quad -1.557 \quad 2.510]$

V. CONCLUSION

We presented a framework for decentralized mapping system for mobile robots. The work has demonstrated that maps from different robots can be successfully merged, from a coarse initial registration and a suitable exchange of data volume. The system uses range measurements from a 3D lidar, generating a local map for each robot. The complete system solves the mapping problem in an efficient way that runs individually online on computers dedicated to two vehicles for preliminary experiments, leading to merged maps on each vehicle.

Future works will consider heterogeneous vehicles, such as a fleet of ground and aerial robots where the view points are less similar and where both communication and computing capabilities are a crucial aspect of the whole process.

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).

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] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM - 3d mapping outdoor environments: Research articles," *J. Field Robot.*, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1002/rob.v24:8/9

[3] 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 *in International Symposium on Safety, Security, and Rescue Robotics. IEEE*, 2011.

[4] 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

[5] 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

[6] 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.

[7] M. Bosse and R. Zlot, "Continuous 3D scan-matching with a spinning 2D laser," in *2009 IEEE International Conference on Robotics and Automation*, May 2009.

[8] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.

[9] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *In Proc. of the Int. Conf. on Artificial Intelligence (IJCAI*, 2003.

[10] 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.

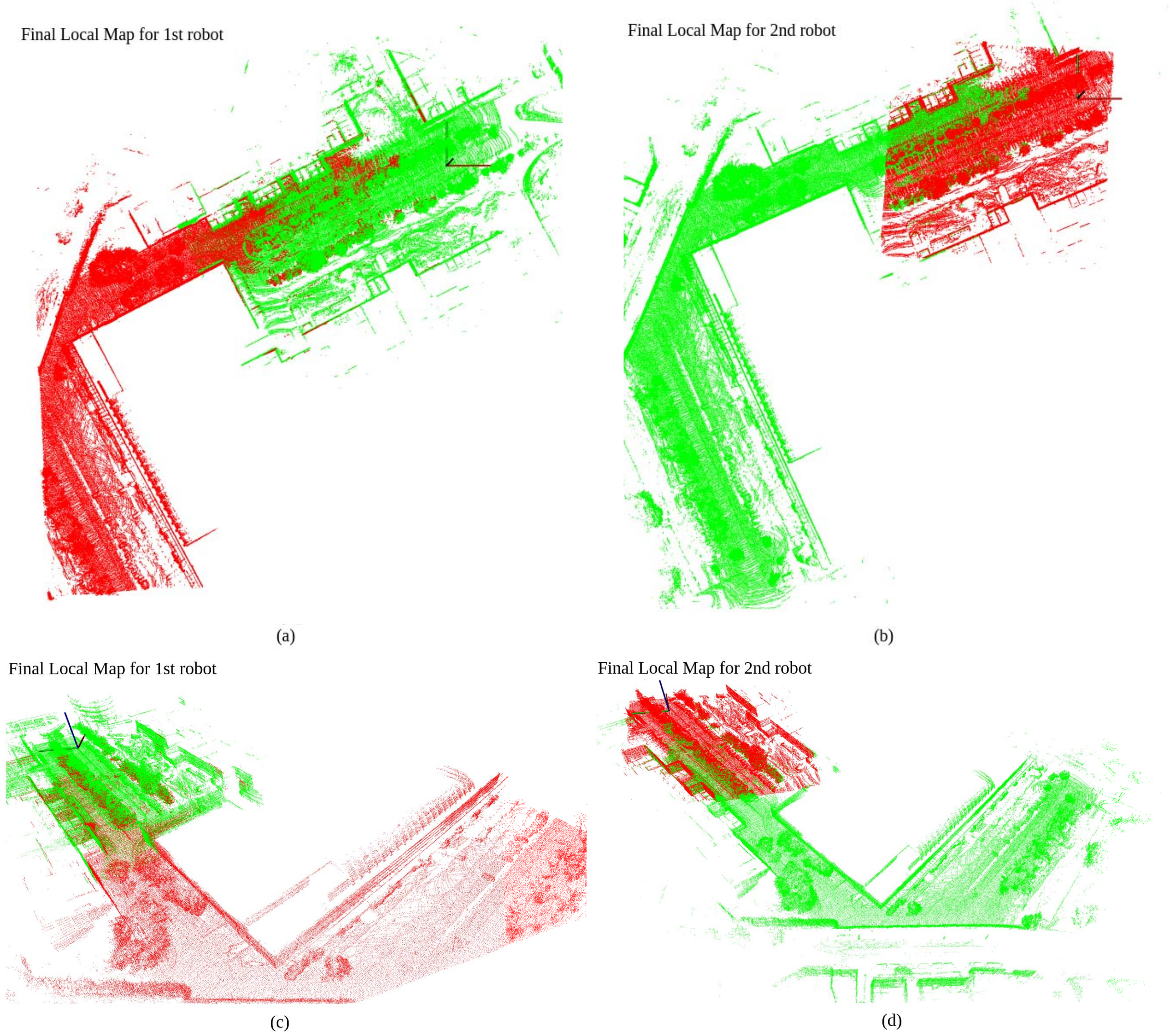

Final Local Map for 1st robot
Final Local Map for 2nd robot
(a)
(b)
Final Local Map for 1st robot
Final Local Map for 2nd robot
(c)
(d)

Fig. 11. Top view of 3D-Map merging result for (a) 1st robot and (b) 2nd robot. (c)(d) Different view of maps.

[11] P. Payeur, P. Hebert, D. Laurendeau, and C. M. Gosselin, “Probabilistic octree modeling of a 3D dynamic environment,” in *Proceedings of International Conference on Robotics and Automation*, Apr 1997.

[12] J. Fournier, B. Ricard, and D. Laurendeau, “Mapping and Exploration of Complex Environments Using Persistent 3D Model,” *Fourth Canadian Conference on Computer and Robot Vision (CRV ’07)*, 2007.

[13] K. Pathak, A. Birk, S. Schwertfeger, and J. Poppinga, “3D Forward Sensor Modeling and Application to Occupancy Grid Based Sensor Fusion,” in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE Press. IEEE Press, 2007.

[14] N. Fairfield, G. Kantor, and D. Wettergreen, “Real-time SLAM with octree evidence grids for exploration in underwater tunnels,” *J. Field Robotics*, 2007.

[15] 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

[16] A. Hornung, K. M. Wurm, and M. Bennewitz, “Humanoid robot localization in complex indoor environments,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010.

[17] J. Jessup, S. N. Givigi, and A. Beaulieu, “Robust and Efficient Multirobot 3-D Mapping Merging With Octree-Based Occupancy Grids,” *IEEE Systems Journal*, 2015.

[18] N. E. Özkucur and H. L. Akın, *Cooperative Multi-robot Map Merging Using Fast-SLAM*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 449–460. [Online]. Available: https://doi.org/10.1007/978-3-642-11876-0_39

[19] J. Zhang and S. Singh, “Visual-lidar odometry and mapping: low-drift, robust, and fast,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

[20] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[21] R. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011.

[22] S. Rusinkiewicz and M. Levoy, “Efficient variants of the ICP algorithm,” in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, Jun. 2001.

[23] 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