2012 International Conference on Indoor Positioning and Indoor Navigation, 13-15th November 2012

Real-time Laser Based SLAM for Multiple Heterogeneous Robots in Indoor Environments

Youssef Ktiri
Department of Mechano-Informatics
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, 113-8656 Tokyo, Japan
Email: youssef@jsk.t.u-tokyo.ac.jp

Masayuki Inaba
Department of Mechano-Informatics
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, 113-8656 Tokyo, Japan
Email: inaba@jsk.t.u-tokyo.ac.jp

***Abstract*—We present in this paper a fast and accurate 2D laser based SLAM for indoor environments. The approach does not presuppose the availability of odometry data and hence is suitable for a large number of robots like UGVs, UAVs or humanoid robots. We use a Rao-Blackwellized particle filter to track the robot position with the number of particle used kept very low. This number increases in case of ambiguity like when traversing corridors. 2D robot pose can be accurately retrieved using a fast scan matching method. We also present an implementation of our algorithm in case of cooperative exploration using UAV and ground robots. Altitude invariant features like walls are extracted and given more weight during the scan matching process. Using this approach we show how fused maps even at different altitudes remain reliable enough to be used by either ground or aerial robots.**

I. Introduction

Autonomous exploration lies at the heart of truly autonomous systems. Systems which show such autonomy can merge easily with human's daily environment, safely navigating and executing higher level orders. Exploration of sparse environments with a single robot can be a time prohibitive task. Using a team of robots allows the use of efficient coordination strategies and achieve the exploration task in substantially shorter time and in more robust way [1][2][3]. Of course, this comes at the expense of a more complex system to build and coordinate. In such cooperative framework, a team can be made of multiple heterogeneous robots, asking for the underlying Simultaneous Localization and mapping (SLAM) approach to handle most of types of robots like humanoids, wheeled robots or UAVs with each of these types having inherent constraints and characteristics. For example, in most of cases, basing on a 2D range sensor, SLAM with wheeled robot can be formulated as a 2D SLAM problem on a plane ground [4][5], while UAVs on the other hand can tilt or roll which asks for building a 6D robust SLAM approach [6][7][8]. Humanoids can tilt their head too looking for obstacles on the ground to avoid and walk away from, asking for a 6D robust handling [9] and a correct handling of such floor obstacles during the mapping process.

In this paper, we propose a fast SLAM implementation which takes into account all three types of robots specificities and can operate in relatively dense indoor environments. We then extend our approach to the case of mapping with a team of heterogeneous robots.

II. Problem Statement

Simultaneous localization and mapping has been one of the most active fields of mobile robotics research during the past decade. The proposed approaches in the literature range from filtering based [4][10][11] to smoothing based approaches [12][13][14] and can build on vision [15] or laser [16]. Most of these algorithms yield very good results in practice, and a combination of two or more of these methods can achieve very good reliability for indoor SLAM problems. Our primary focus in the present work is to build a SLAM algorithm that handles different types of robots, mainly UAVs, humanoid robots and wheeled robots and can operate in dense environments.

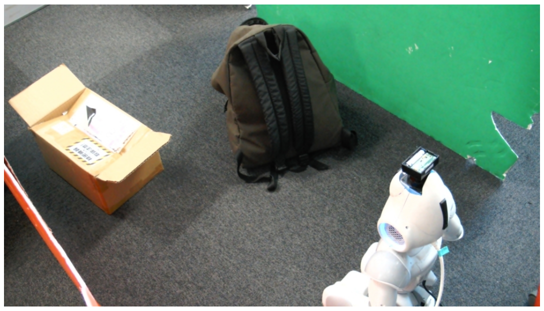
Fig. 1. Mapping with a Humanoid robot

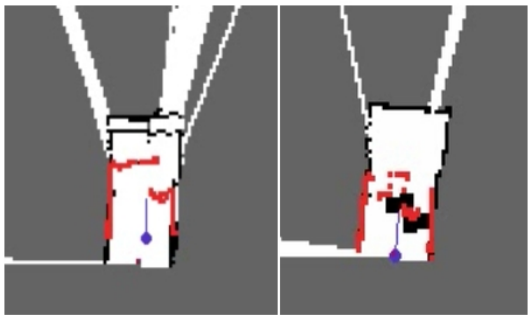
Fig. 2. Mapping with a Humanoid robot

Let's consider the example Fig. 1. A standing Humanoid robot tilts its head, observing the wall, the floor and then floor obstacles. Fig. 2 shows at low resolution the result of mapping in such scenario. A 2D SLAM approach yields the result shown on the left on Fig. 2. As can be seen, the floor and the obstacles on the ground create corruption in the map. The robot seems to be moving forward while it's standing static and tilting its head only. The right part of Fig. 2 shows a proper handling where the map is correct and floor obstacles are detected as different components of the map.

This shows the necessity of a SLAM approach general enough to handle different kind of robots. Navigation with wheeled robot (with no laser tilt) can be expressed as a 2D SLAM problem. The laser sensor on such robots being in general at relatively low level, floor obstacles can be easily avoided. UAV's navigation on the other hand can hardly be approximated as a 2D slam problem. UAV can show non negligible roll and pitch when moving and their altitude is an additional factor to estimate. This asks for a proper knowledge of the attitude and altitude (6D robustness) of the robot to create reliable 2D maps. When mapping with a 2D laser sensor, a classic approach is to fuse the result of a 2D SLAM with the data coming from an IMU or a MARG (Magnetic, Angular rate and Gravity) sensor [6][7][17]. On the top of that, an assumption to fully constrain our model is the 2.5D assumption where all object in the world are supposed to fall perpendicularly on the floor. If this is the case for walls, this is certainly not the case for most of objects in our environments like bookshelves or chairs. These objects when mapped create corruption in the map. Given that most of corrupting objects are at relatively low altitude level, during the flight, the UAV sees generally good proportions of the walls and does not bother about the floor plane or objects lying on the ground. Finally, UAVs typically don't possess odometry data. Displacements have to be extracted by other methods such as scan-matching or visual odometry. Things gets more complex when mapping with humanoids. Humanoids typically don't possess laser sensors in their legs, and can fall easily in contact with ground obstacles. This asks for a proper detection and avoidance of obstacles on the floor. Moreover, when they tilt their head in densely populated environments, humanoid mapping goes through 4 different stages :

- first stage where the tilt is nearly 0 and in which case the SLAM problem is a straightforward 2D problem
- second stage where the tilt angle is small and big proportion of walls can be seen which yield robust result when mapping with a 2D SLAM and associated IMU
- third stage where the tilt angle is big but the floor is still not seen and in which mapping becomes harder given that laser beam reflect on floor objects that do not respect the 2.5 world assumption
- finally a fourth stage where the laser hits the floor and floor objects as well creating an ambiguous environment like Fig. 2. In such case, mapping augments the probability of creating highly corrupted maps and later ending up in a localization failure.

Based on the previous remarks, our approach bases on the following ideas to solve the presented issues. First, the 2D front-end approach used has to be as fast as possible. As said before, when the robot faces the ground, scan-matching or mapping is likely to produce erroneous results especially in dense environments. The 2D front-end SLAM has to handle this issue by recovering the robot position as soon as the floor is not detected anymore. This is to cope with the case of Humanoid robots. Then, the general approach has to be 6D robust to account for possible tilts and rolls such as when using an UAV, Humanoid or a wheeled robot on a slope. Finally, our approach has to find low semantics in the map such as floor, walls and obstacles on the floor and handle each separately when registering scans in the overall map.

III. Laser based SLAM for heterogeneous robots

A. 2D front-end slam

1) General approach: We take a grid based approach where each grid models an occupancy state (unknown, obstacle, free). In practice, each grid stores a probability of occupancy. The grid size was taken as 5x5 cm which we believe is a good trade-off between speed and accuracy.

In all the following the 3D navigation coordinate system is taken as a right-handed system with the z axis pointing up. We call navigation frame the robot centered frame described by the three parameters (x, y, θ).

We "locally" rely on a Rao-Blackwellized particle filter where each particle represents $< x_t^i, m_{t-1}^i, w >$ and samples the joint probability on position and maps. The sensor model $P(z_t \mid x_t^i, m_{t-1}^i)$ is taken gaussian with low variance and we used the optimal proposal distribution $P(x_t \mid m_{t-1}^i, z_{l,t}, u_t, x_{t-1}^i)$ for Rao-Blackwellized particle filter update. Since we do not rely on odometry to perform the prediction step, a scan-matching step is performed to calculate robot displacement between laser updates.

For most of non dense and non ambiguous environments, we noticed however that one particle would suffice to yield accurate results and using multiple particles would be a waste of processing time compared to the gain in accuracy. We hence restrict the use of multiple particles only to two scenarios :

- when the laser faces the ground (such as in the case of humanoid robots), we stop scan-matching. Within such period of time however, the robot can move beyond the reach of a the window size used for position update. To account for this phenomenon one could enlarge the search area. A faster and more elegant approach is to take into consideration the previous state and commands of the robot. If we know the previous robot state (position and speed) jointly with the commands sent by the controller to the robot then we can formulate a guess on the actual robot position even when the robot's head faces the floor. This is the role of an Extended Kalman Filter (EKF) based state updater presented in a further section in this paper. Having such prediction, we generate particles

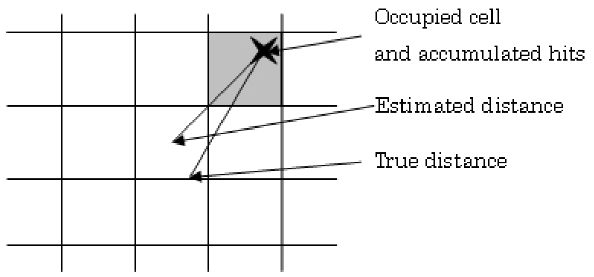

Fig. 3. Storing likelihoods of a hit in lookup tables

around the predicted position and update when the floor is not detected anymore and scan-matching can restart again.

- since we don't use any visual based approaches in our current system implementation, loop closure for single robots are recovered by scan-matching. To accelerate the search for loop closure candidates we restrict the search for loop closure candidates to a limited search area (typically 45 degrees in rotation and within 2m radius). The typical limitation to such approach is in the case of mapping in a featureless corridor much longer than the laser range maximum detection range. In such a case we can only base on controller commands as a source of knowledge on translational displacement. To be sure to detect a loop in such conditions we generate more particles when the displacement is likely to exceed the maximum detection range of our loop closure algorithm.

when ambiguity vanishes then we resample and keep only the best particle to continue navigation with.

B. Lookup table construction

Let's consider the map $\mathbf{m}$ computed during previous steps. Each cell stores the likelihood $p_{hit} \sim exp(-\frac{d}{\sigma})$ where d is taken as the distance to the closest cell hit by a laser beam so far. Since these values are likely to be needed a high number of times, these have to be precomputed and stored. Doing so is one of the keys to great speed enhancement and to online feasibility. Of course this yields approximations in the computed probabilities (since we don't know where exactly inside a cell is the laser going to hit but are formulating an estimate). Fig.3 shows the difference that can occur in the computed probabilities.

Free cells in the vicinity of occupied cell will have non null likelihood of a hit, this is to model the fact that a previous robot guess may have been wrong and that the laser sensor data can never be perfect. Each freshly hit cell updates its occupancy state as well as the accumulated hit point. This is done to have maximum precision in our scoring process and to account for the fact that some of the cells may be only hit in the corner. This phenomenon has to be reflected when calculating the lookup table values. Hit cells then update other free cells in its vicinity according to an exponential model. Inconsistency can arise from the fact that only a guess can be expressed by calculating the likelihood from the distance to the center of the neighboring cell while the true laser hit may occur far from the center of the cell. Such a case is illustrated in Fig.3. In practice however, such a model even being approximate is consistent enough to yield a correct convergence of the optimization process. In further steps, if a previously updated cell comes to be set as free again (being traversed too many times by laser beams) the lookup table has to be revised again to account for such a change.

C. Scan-Matching

Scan-matching has been extensively used for this purpose and can be implemented to overcome scenarios where no odometry data is available (navigating with UAV in indoor environments for instance). It aims at aligning currently available laser scans with previous scans or with a given map. To do so many approaches have been taken. Some aim at finding the best point to point correspondences like ICP based methods, other aim at feature to feature correspondences. A third paradigm aims at computing the rigid body transform which yields a maximum of likelihood given the map computed so far like [18]. We take a similar approach in the scope of this work. For our case, scan-matching aims at finding the best rigid body transform which aligns actual laser scan with the previously registered map. Accounting for all laser hit independently, our goal can hence be expressed as:

$$\mathbf{T}(\mathbf{t_x}, \mathbf{t_y}, \phi)^* = argmaxP_{hit}(\mathbf{T} \otimes \mathbf{E} \mid \mathbf{m}) \tag{1}$$

Where $\mathbf{E}$ is the laser measurement and $\mathbf{m}$ our map. The previous equation describes a planar problem since the transformation matrix T accounts only for planar displacements. As it has been shown in previous section such a model fails to capture tilting and rolling noise or voluntary head attitude change and leads to potential failure in most of the cases. For such a reason it has to be changed to account for possible roll and pitch.

D. 6D Robustness

6D robustness can be achieved if we know the roll and pitch of the robot, or in the case of humanoid robots, the roll and pitch of the robot's head and using the 2.5D assumption about the world. The information on tilting and rolling movement is given by a separate MARG sensor. The sensor also provides heading information. Such information if not corrupted can be used as a start to the scan-matching process. For UAVs, the altitude z can change. By changing, the map can also change. However, for most of the case and since UAVs stand relatively far from the floor, a big proportion of walls can still be seen yielding changes at very local points on the map in most of the environment. Moreover, if we account for rolling and tilting as parameters to optimize in our state vector, the scan-matching step takes much more time to finish which is not desirable. We derived our sensor fusion algorithm for MARG attitude estimate very carefully, and given our map resolution, the error on the provided roll and pitch estimate are negligible. Hence, we take the roll and pitch estimates as granted and do not seek further optimization on these parameters. In the sensor frame the beam end point has coordinates :

$$\mathbf{e^i_{sensor}} = (x^i_{sensor}, y^i_{sensor}, z^i_{sensor}, \theta^i_{sensor}) \tag{2}$$

The problem can be restated as :

$$\begin{aligned}&\mathbf{T}(\mathbf{t_x}, \mathbf{t_y}, \phi)^* = \\ &\quad argmaxP_{hit}(\mathbf{T}_{\mathbf{tx},\mathbf{ty},\phi} \otimes \mathbf{T}^{global}_{navigation} \\ &\quad \otimes \mathbf{O}(\mathbf{T}_{\beta\gamma}\mathbf{E_{sensor}}) \mid \mathbf{m})\end{aligned}$$

where the matrix $\mathbf{O}$ designates an orthography transform to the initial robot plane, $\beta\,\gamma$ account for the pitch and roll and $\mathbf{E}$ is our laser sensor measurement. As we said before, since we use an orthography transform of 2D laser slice, we are only writing an "approximate description" of the world supposing that all objects fall orthogonally on the ground.

E. Extracting semantics

The scan-matcher takes into consideration every laser end-point as a separate piece of the total probability estimate. If we could separate walls from other objects giving more weight to walls, the total scan-matching result would be drastically more robust to roll and pitch changes. Detecting the walls from one laser scan is not easy. However detecting walls from multiple and successive laser scans is relatively straightforward following a simple definition: the walls are the invariants under the 2.5D assumption. Thus, when the robot tilts, the projection on the plane ground that hits the expected cells under the 2.5D corresponds in our case to "walls". An additional score is added to such grid cells giving them more weight during the scan-matching step. The second element to estimate in our semantics is the floor. Detecting the floor should stop registration and position update and generate more particles to keep consistency in the map. The floor position is calculated very easily knowing the current altitude z by transforming from the sensor frame to the global frame and extracting point corresponding to altitude 0 in the global coordinate frame. Since we know the expected position of the floor, we know which laser points correspond to obstacles on the floor. The points are assembled and marked as floor obstacles. Cells corresponding to floor obstacles have a null score so not to corrupt the scan-matching process yielding a substantially more robust approach. These are however marked as occupied space and taken into consideration by our planner when generating safe trajectories for robots.

F. Robot State Update

The full state $X = [x\,y\,z\,\dot{x}\,\dot{y}\,\dot{z}\,\psi\,\theta\,\phi\,\dot{\psi}\,\dot{\theta}\,\dot{\phi}]^T$ is updated with varying delay depending on communication delays, on the current scenario (interruption of scan-matching when seeing the floor), ambiguity in the environment. For more guarantee on the reactivity and the predictability of the system, and also to keep track of robot positions during SLAM interruptions, we maintain a state estimate at constant rate using a separate Extended Kalman Filter. In practice, we do not include the roll and pitch data in the following since these are directly given by the MARG sensor. Moreover, at the current stage of our system, z is not estimated. UAVs are dealt with just like wheeled robots supposing that UAVs will never see the floor. For Humanoids z is estimated based on the current laser tilt and laser position within the robot body and is used to estimate the floor and floor obstacles position.

The filter predicts its state given a simple motion model then is updated with MARG sensor and scan-matching results. For instance the prediction for $x\,y\,\dot{x}\,\dot{y}$ is given by :

$$\begin{aligned}x_{k+1} &= x_k + \delta t(cos(\phi)v_x - sin(\phi)v_y) + w_x \\ y_{k+1} &= y_k + \delta t(sin(\phi)v_x + cos(\phi)v_y) + w_y \\ \dot{x}_{k+1} &= \dot{x}_k + w_{\dot{x}} \\ \dot{y}_{k+1} &= \dot{y}_k + w_{\dot{y}}\end{aligned} \tag{3}$$

with w_X being the white noise associated with parameter X. Measurement updates come essentially from the scan-matching result. The scanmatcher essentially gives the displacement between two different timestamps. From such information we can also extract observations of velocity. The grid resolution imposes however a bound on the precision we can extract following such approach. For 5cm grid cell the robot can appear not moving between two laser scans. We hence maintain a history of laser scan results an update velocities based on the difference of sufficiently time spaced samples. Such approach, if it gives more accurate estimates during constant speed movements, does not handle very well fast velocity changes due to the smoothing we operate. In practice, giving an appropriate averaging time (around 1s) can yield results good enough to perform tasks relying on estimates of the robot velocities.

Since the scan-matching result comes with high delays (20 to 10 Hz as update rate) than the higher frequency MARG data (100Hz), such out of sequence measurement has to be correctly dealt with. In our case, the state dimension is low and the jacobians quite simple to derive. We hence first keep a history of previous state updates. When an out of sequence measurement reaches we drop all measurements made after the measurement data. As this stage the problem becomes a pure 1-step lag problem and solved consequently. Then, we re-update the state the appropriate number of times until the most recent timestamp. Doing so we are sure to find an optimal estimate.

IV. Experimentation

We first show an application of our approach in the case of a humanoid Robot. Fig. 4 shows our experimental environment. A Nao robot [19] has to explore autonomously an indoor environment with randomly placed ground obstacles until it finds a given target landmark as shown in Fig. 5. The Robot tilts its head up and down to detect obstacles on the floor. Doing so it also detects the floor itself. Fig. 6 shows an intermediate result of the SLAM process. As shown in the figure, obstacles on the ground are correctly detected as separate components of the map and handled correctly by the planner to yield a safe trajectory to the next best target.

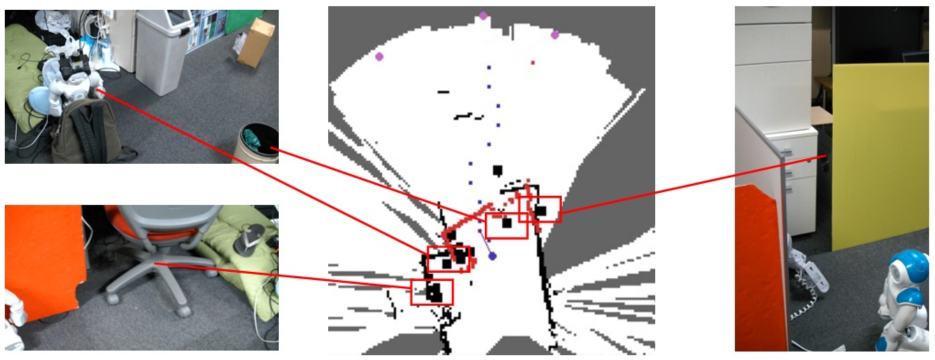
Fig. 6. Example of our SLAM approach for a Humanoid Robot in dense environment

Fig. 4. Test environment

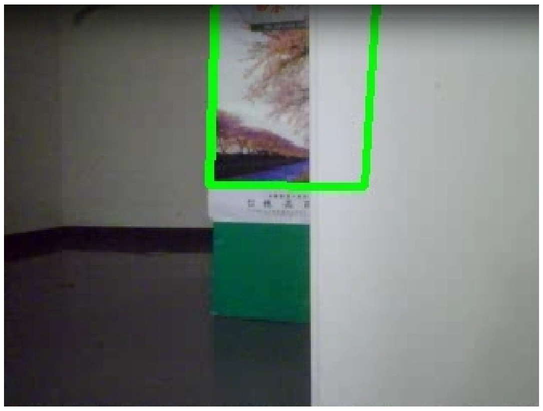
Fig. 5. Found landmark

The front-end SLAM algorithm has an update rate of 20Hz. As it has been said before, the SLAM stops when the robot laser faces the ground. The red line in Fig. 6 is hitting the floor, and mapping in such case would yield big inaccuracies. In such case the EKF state updater provides with the necessary prediction information to generate more particles. When the robot sees the walls again the SLAM can be updated once again.

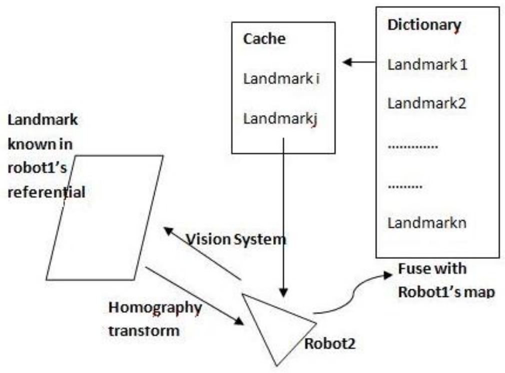

Fig. 7. Relative pose determination

V. SLAM WITH MULTIPLE ROBOTS

Growing the number of robots involved in the exploration task is a solution to explore larger surfaces with a maximum time efficiency [20][21][2][22][3][1][23][24]. Early approaches dealt with the multi-robot problem as a straightforward extension to the single robot case, principally making the assumption of a common frame of reference and unlimited communication range. A simple solution to the second assumption is to make every robot of the cooperating team highly independent, running all components of the SLAM individually and communicating whenever communication constraints allow it. The first assumption's validity highly depends on the kind of cooperative system being in use. If the robots are known to start at a known point with known configurations then one can assume that all the robots share a common frame of reference. In such case doing SLAM in a cooperative way can be pretty straightforwardly dealt with. However in practice, this is almost never the case. Robots can start at two different points without any knowledge of their relative positions. In such a case, the robots should be able to recover the transform between their respective frame of reference either in a synchronous way such as direct observation of a robot by another robot, or asynchronously by recognizing a part of map previously seen by a teammate robot. Robot to robot direct observation is not an easy task unless one uses special markers easy to detect. For the most general case where we use heterogeneous robots without any prior knowledge of the robot involved in the cooperative task, synchronous observation can hardly occur. Moreover in very sparse environments direct observation of a robot by a teammate robot becomes a rare event, making any cooperation between robots hard. for such reasons, in the most general case, in sparse environments and without any markers placed on the robots, every single robot should be able to recover the relative position of a pair team robot by inference from its own knowledge and other teammate's knowledge. As an example of such strategy robots can record several different landmarks extracted randomly and sparsely from the environment and broadcast such knowledge through the network. Each teammate robot then checks if one of these landmarks appears in the current map being explored. If such is the case, the teammate robot's position can be recovered easily, and by transitivity, every cooperating robot's position can also be received. We take such approach in the scope of the present work as shown in Fig 7.

A. Relative Pose Determination

We place ourselves in the general case where no initial relative poses are given. In such case, recovering such information is an essential step during the multi-robots SLAM process and a prerequisite to any coordination strategy. We use 3D landmark recognition to recover a good estimate of the transformation existing between the frames of reference if two robots based on the homography matrix returned by the vision algorithm. In practice, any feature+associated classifier vision algorithm can be suitable for this task. 3D detection capabilities under severe affine transforms is necessary to accommodate for the case where the robots look at a landmark from very different angle of views or altitudes. Without such ability, the robots may miss a common landmark even when its position is close enough which increases the need to store even more landmarks in our database to be sure an asynchronous observation can be made as soon as possible.

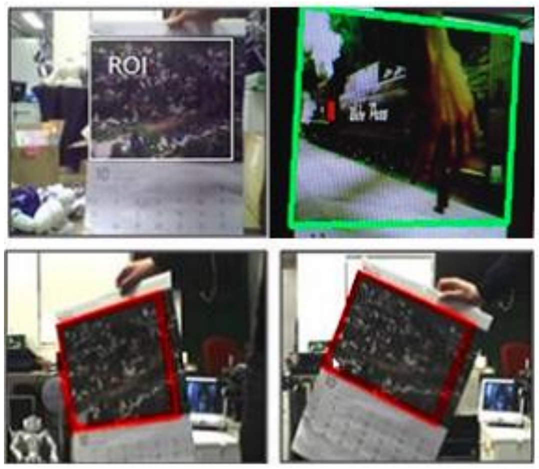

Fig. 8. 3D detection algorithm results

Fig.8 shows an example of our 3D landmark detection system based on the work presented in in [25]. The result of the algorithm is a homography matrix from which we can extract planar parameters $(x_l\, y_l\, \theta_l)$. The transforms between two robots can then be easily retrieved following the equation :

$$T_{robot2}^{robot1} = T_L^{robot1} T_{robot2}^L \tag{4}$$

The transform T_{robot2}^{robot1} given by the vision algorithm constitutes in most of the cases a good estimate of the relative position. As our most important goal is consistency with the environment map we add a local scan-matching step that refines the translational and rotational parameters. The matrix T_{robot2}^{robot1} as well as T_{robot1}^{robot2} are the result to be stored and used by cooperating robots so as to recover at every time step other robots position.

B. Map merging

In the previous section we computed an estimate of the current transform between two robots. One important point remaining is how to combine multiple measurements/constraints made between the two same robots. Each of these constraints represent a measurement associated with noise. Given this noise/covariance matrices associated with the set of different measurements we can derive an optimized solution mapping robot1's position and associated map to robot2's frame of reference. We use the same framework presented by [23] that uses anchor nodes and bases on factor graph solving by isam library[12]. In this context every robot updates its own graph separately. If more than one inter-robot constraint is added to problem becomes an optimization problem yielding an optimized transformation between the two robots frames of reference. If only one constraint measurement is made, the transformation is generally very noisy. The local merged map is generally consistent, but further points of the map may appear off of the track causing mapping problems. From a practical point of view, when a teammate map is added to a robot's map, this actually corresponds to one observation added to the map. Even in case of noisy data this is often not enough to erase good parts of the initial map that have very high mapping score such as walls. The effect of merging is such that only previously unseen spots are going to appear affected, previously mapped cell retain their value which originates from a progressive and longer mapping.

Inter-robot measurements in a fully autonomous system are rare events in the case of asynchronous maps especially when the environment to explore is sparse. On the other hand, the consequences of a bad transform estimate originating from too few available constraints can be non negligible in merged maps far from the current merging point and previously unseen by the current robot. In such case, the robot essentially builds on the teammate robot's map to navigate. If the deformation between the map merged is big, since the scan-matching is a progressive process the previously merged map is going to be overwritten with current better estimates in the best case. In some cases however in the current robot's location lacks enough laser features, the robot may end up considering the merged map as a good map and updates its position accordingly, constituting like a jump in the deformed merged map and mapping all further observations accordingly, creating two locally consistent maps but a globally deformed total map. A solution to such phenomenon is to find more than one constraint between the robots graph at two different and distant locations before really merging the maps. Doing so, we can be sure that the merged map and the contribution of the cooperative robot is going to be consistent with the original map created by the current robot. However, this may delay any cooperative work to a later time step when a second constraint can be recovered. To save time, we look "manually" for constraints in the environment via scan-matching. This step is often very fast since the current landmark observation gives an estimate of the transformation. We hence look in both graph convergence points of the trajectory close enough from each other for scan-matching and far enough from the current robot position to yield a second good constraint to do optimization with.

Finally, when merging too maps acquired by two heterogeneous robots such as two maps acquired at different altitudes, good features from the original robot's map cannot be erased since they were acquired through a longer mapping process and have high scores, while the new merged map yields typically one observation which is not enough to decrease a good score to yield a change in the map structure. Thus, only points that are originally unknown grid cells will be updated. All the others in most the cases remain unchanged.

VI. Experimentation in case of Multiple Robots

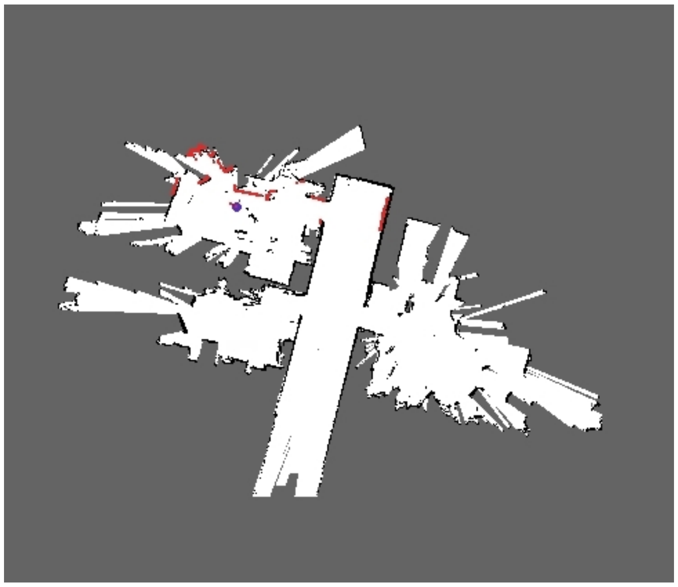

Fig. 9. Map by robot 1

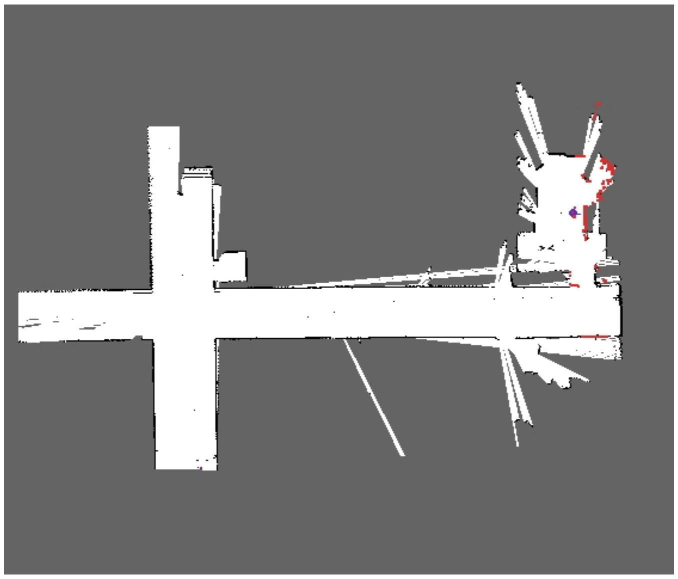

Fig. 10. Map by robot 2

We propose here to check the validity of our multi-robots mapping system. Two robots 1 and 2 explore asynchronously the environment and stop when they find a landmark. Fig. 9 and 10 represent the maps acquired by robot1 and robot2. As it can be seen, the frames of reference are taken randomly, and no information about the relative pose estimates is given, which is the general case we tried to solve. The map merging module updates two different pose graphs until the two robots make a landmark observation. Robot 1 is a Humanoid robot while robot2's map has been acquired by holding a computer connected to laser sensor and a camera (which simulates a UAV)and walking through the environment. A commonly retrieved landmark, placed where the robot1 stands in Fig. 9 will allow us to verify our map merging system.

Adding the landmark measurement constraint allows to calculate a first estimate of the transform between the two robots. Fig. 11 shows the recovered trajectories and Fig. 12 shows the merged map given to robot 1. The map contains mapping inconsistencies since the corridors in robot 1's map and robot 2's map don't overlap perfectly. As it has been said before, the merged map gives only one observation update and on the overall does not affect the robot 1's map that has been formed through a longer mapping process. This fused map even if it contains inconsistencies can still be of great use for navigation.

Fig. 12 shows an example of inconsistency between the original map and the merged map. The ray represented in red is given by the map merging module to update the new merged map. As it can be seen, the ray does not perfectly overlap with robot 1's map. Once again, this is due to the few number of constraints we have between our graphs. To add more consistency to the map merging step, we look retrospectively for a new constraint which is shown in Fig. 13.

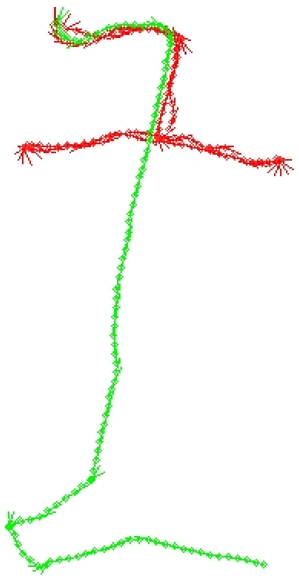

Fig. 11. Recovered trajectory of robot 1 and 2

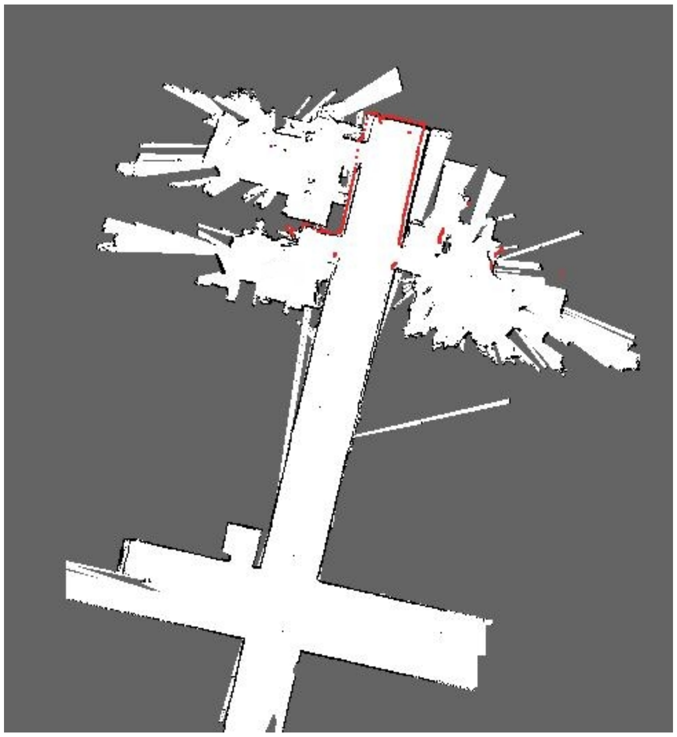

Fig. 12. Misalignment of laser rays with the fused map

The laser scans originated from the first fused map show the nature of inconsistencies existing in the overall map. These can be further corrected adding a scan-matching constraint between the graphs. These constraints are associated with covariance matrices that has to account precisely for the nature of deformation existing. Once the second constraint is successfully detected and added between the graphs, the map merging module runs a batch optimization to yield a better estimate of the transform between the two robots. The result is shown in Fig. 14 as the new trajectory have a slightly different deformation in rotational and Y translational axes. As it can be noticed, the laser ray from robot 2's map overlaps in a more consistent way with robot 1 map. The process provides a more accurate global map.

VII. Conclusion and Future Work

This work presented a SLAM approach suitable for many types of robots like UAVs, wheeled and humanoids as well as a Multi-robot map merging method. On the single robot exploration part, 6D robust simultaneous localization was performed by merging data from MARG sensor and 2D fast front-end SLAM taking into account low semantics in the map such as walls, floors and obstacles lying on the floor to yield an even more consistent map estimate. On the multiple robot part, we made use of a 3D landmark detection system to check for landmarks seen in common and recover relative poses between robots. Map merging can take place by maintaining independent trajectory graphs for each robot and adding constraints corresponding to inter-robot observations. These measurement consist in landmark observation as well as additional constraints added by scan-matching to increase the consistency of the globally merged maps. Results were given in case of mapping with two heterogeneous robots.

In our current map merging implementation, the main purpose was to refine the relative pose estimate by manually looking for additional constraints between team robots graphs. By using one global transform we do not perfectly recover from twisted transformation estimates that can exist between locally deformed area. As a future work, it would be interesting to take an intermediate approach between [26] and [23], calculating transforms for different locally merged maps and parts of a more global map.

References

[1] Regis Vincent, Dieter Fox, Jonathan Ko, Kurt Konolige, Benson Limketkai, Benoit Morisset, Charles Ortiz, Dirk Schulz, and Benjamin Stewart. Distributed multirobot exploration, mapping, and task allocation. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):229–255, April 2008.

[2] W. Burgard, M. Moors, C. Stachniss, and F.E. Schneider. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21(3):376 – 386, june 2005.

[3] K. Konolige D. Fox, J. Ko, B. Limketkai, D. Schulz, and B. Stewart. Distributed multi-robot exploration and mapping. *Proceedings of the IEEE*, 2006.

[4] W. Burgard G. Grisetti, C. Stachniss. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Robotics and Automation*, 2005.

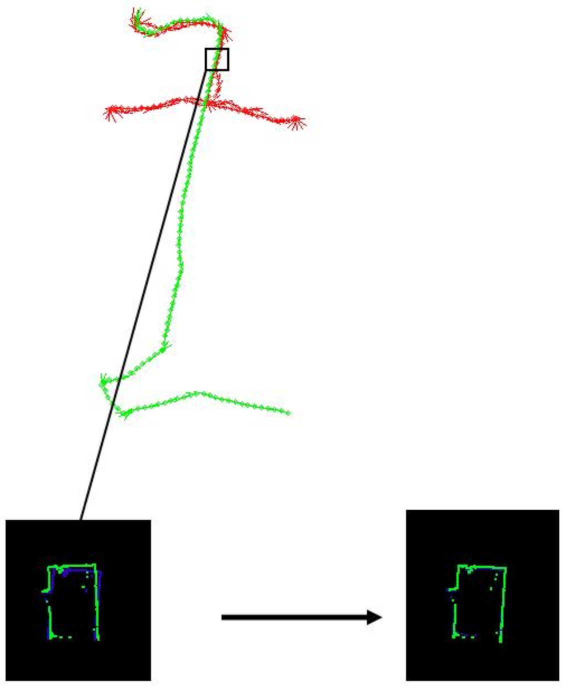

Fig. 13. Adding a new constraint to the fusion process

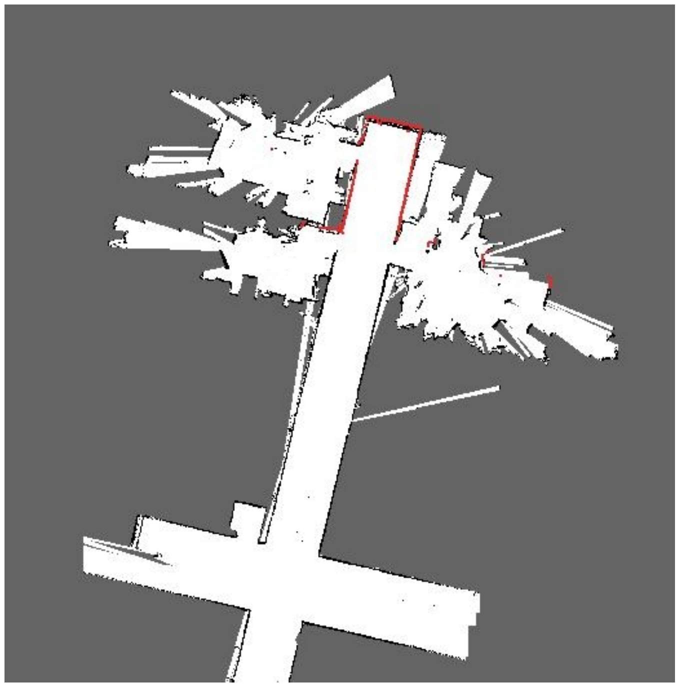

Fig. 14. Fused map with improved consistency

[5] A. Eliazard and R. Parr. Dp-slam:fast robust simultaneous localization and mapping without predetermined landmarks. *Proceeding of the Intrnational Conference on Artificial Intelligence*, 2003.

[6] Shaojie Shen, N. Michael, and V. Kumar. Autonomous multi-floor indoor navigation with a computationally constrained mav. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 20 –25, may 2011.

[7] A. Bachrach, A. de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. Range - robust autonomous navigation in gps-denied environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1096 –1097, may 2010.

[8] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2878 –2883, may 2009.

[9] Cyrill Stachniss, Maren Bennewitz, Giorgio Grisetti, Sven Behnke, and Wolfram Burgard. How to learn accurate grid maps with a humanoid. In *ICRA*, pages 3194–3199. IEEE, 2008.

[10] Tim Bailey and Hugh Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, September 2006.

[11] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press, August 2005.

[12] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics (TRO)*, 24(6):1365–1378, December 2008.

[13] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, page 2009.

[14] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[15] Andrew J. Davison, lan D. Reid, Nicholas D. Molton, and Oliview Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1–16, 2007.

[16] D. Fox D. Hahnel, W. Burgard and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. *International Conference on In Intelligent Robots and Systems*, 2003.

[17] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, Kyoto, Japan, November 1-5 2011.

[18] E.B. Olson. Real-time correlative scan matching. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4387 –4393, may 2009.

[19] Aldebaran Robotics. www.aldebaran-robotics.com.

[20] S. Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. *The International journal of Robotics Research*, 20(5):335–363, 2001.

[21] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 476 –481 vol.1, 2000.

[22] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robtics Research*, 25(12):1243–1256, 2006.

[23] Been Kim, Michael Kaess, Luke Fletcher, John Leonard, Abraham Bachrach, Nicholas Roy, and Seth Teller. Multiple relative pose graphs for robust cooperative mapping, 2010.

[24] N. Michael, S. Shen, K. Mohta, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots, 2012.

[25] V. Lepetit M. Ozuyal, M. Calonder and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[26] Kai Ni, Drew Steedly, and Frank Dellaert. Tectonic sam: exact, out-of-core, submap-based slam. In *In Proc. IEEE International Conference on Robotics and Automation*, pages 1678–1685, 2007.