Consistent Multi-robot Decentralized SLAM with Unknown Initial Positions

Guillaume Bresson, Romuald Aufrère, Roland Chapuis

Consistent Multi-robot Decentralized SLAM with Unknown Initial Positions

Guillaume Bresson*, Romuald Aufrère*† and Roland Chapuis*

*Institut Pascal - UMR 6602 CNRS – †LIMOS - UMR 6158 CNRS
Clermont Université, Université Blaise Pascal - Aubière, France
firstname.name@univ-bpclermont.fr

***Abstract*—This paper presents a multi-vehicle decentralized SLAM algorithm. We expose the different problems involved by this decentralized setting, such as network aspects (data losses, latencies or bandwidth requirements) or data incest (double-counting information), and address them. In order to ease the data association process and also guarantee the consistency of the vehicles localizations, we introduce a new model to represent the natural drift affecting SLAM algorithms. By integrating this model, loop closures, associations between robots and absolute information can be easily taken into account. A general framework has been designed thus allowing to use any SLAM algorithm. To demonstrate the feasibility of our approach, we applied it to a monocular solution. It is the first time, to our knowledge, that a monocular decentralized SLAM is presented. A multi-robot data association algorithm, based on geometric constraints is also exposed in this paper. The validation is performed thanks to a simulator presenting a realistic physics. The results show that the localizations consistency is preserved. It also demonstrates that multi-vehicle monocular SLAM is viable in urban environments.**

I. Introduction

Intelligent vehicles are now able to accomplish difficult tasks. Driving autonomously from one point to another, detecting moving objects, performing complex maneuvers are examples of what a single robot is capable of. However, approaches involving several vehicles are only starting to be considered even though many applications would benefit from them. Indeed, it brings new challenges in all the fields concerned by autonomous navigation (data fusion, data association...) as well as issues related to communication, exchange strategies...

Among the applications concerned by multi-robot formation, we can cite: fast exploration [1], risk assessment in dangerous situations [2], coordinated transportation systems with augmented reality [3] or collaboration in a heterogeneous team of robots [4]. All these examples of research activities have in common the fact that robots need to be able to localize themselves, without prior knowledge, and share the information with the rest of the group.

Self-localization in an unknown environment is, most of the time, accomplished using Simultaneous Localization And Mapping (SLAM) methods [5]. A map is built incrementally in order to help the localization process. In a multi-vehicle process, this method still applies as the maps can be shared among the robots so as to have a global map (referenced in a frame common to the whole fleet) taking into account all the locations visited by the different robots.

However, many algorithms rely on a centralized scheme [6][4]. All vehicles communicate their maps to a single entity which then agglomerates them in order to obtain a global map. This one is shared and improved by the team. Nevertheless, the problem is that in the case of a failure of the centralizing unit, the whole system will cease to work. Furthermore, vehicles must be within reach of the central entity to have access to the map. On the other hand, centralized methods have the advantage to be easier to develop as they naturally extend the concepts introduced in mono-vehicle.

Decentralized systems are more complex as they require a more elaborate data exchange scheme. Many algorithms proposed in the literature only focus on specific points. Often, they do not consider network aspects [7][8] or only present simulation results [9][10]. However, one key element is discussed in almost every decentralized approaches: data incest [11][9]. Double-counting information can have terrible consequences as it can cause the inconsistency of the estimation process which will consequently compute wrong vehicles and landmarks localizations.

The main objective of mixing data coming from the different vehicles is to put their respective maps in a common frame to estimate their inter-distances. To do so, it is necessary for a vehicle to recognize an area previously visited by itself or another robot. Data association is consequently an important aspect of the system but is not the only one. Indeed, in order to help the data association process, the localization must be consistent (true position of the vehicle inside the estimated uncertainty). It means that data incest must be avoided but also that consistency must be guaranteed at a local level.

SLAM algorithms themselves have been demonstrated to naturally drift over time because of the highly non-linear models used [12], thus preventing to ensure the consistency of the localization [13]. This effect becomes more visible when the robot travels over long distances. Solutions have been proposed to avoid this problem [14] without considering the drift itself. Loop closure (or integration of absolute information) is one of the mean allowing to correct the divergence of the SLAM process. However, even with these corrections, the consistency of the localization is still not ensured. Moreover, modeling the divergence can help fusing geo-referenced data and loop closures in the SLAM process.

In this paper, we propose a multi-vehicle decentralized

SLAM algorithm:

- integrating network aspects (bandwidth required, data latencies, data losses...)
- avoiding data incest and ensuring consistency of the localization at a local and global level
- based on a general framework that can be applied to any SLAM algorithm
- applied to a monocular solution for the first time to our knowledge
- validated using a simulator presenting a realistic physics and environments mapped from real locations

The paper is divided as follows: Section II will present the notion of bias and the model developed to integrate the drift of the SLAM. Section III will introduce the general framework designed to handle the decentralized part of the algorithm. Then, Section IV will expose the data association process in charge of identifying landmarks common to the different vehicles. Finally, Section V will detail the experiments conducted and the results obtained.

II. Drift model: notion of static and dynamic bias

As explained before, the SLAM naturally drifts over time. It means that the estimation process is affected by a bias which is evolving according to the distance traveled. This behavior is depicted in Figure 1. The bias affecting the localization at one moment is thus different from the one at another moment. The direct consequence is that the landmarks mapped are subject to a different bias estimate.

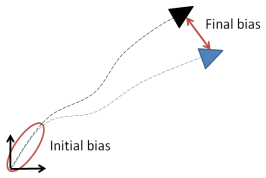


Fig. 1. Example of a vehicle drift and its evolution. The black curve represents the ground truth and the blue one its estimate.

We chose to model the bias as the 3 parameters (position and angle) allowing to pass from the frame affected by the divergence to the unbiased one. It can be defined as:

$$\mathbf{b} = \begin{pmatrix} b_x & b_y & b_\theta \end{pmatrix}^T$$

b_x and b_y correspond to a position bias whereas b_θ represents an angular bias. The vehicle pose is estimated with the same parameters:

$$\mathbf{v} = \begin{pmatrix} v_x & v_y & v_\theta \end{pmatrix}^T$$

An estimate of a landmark i is defined as follows in two dimensions:

$$\mathbf{l}_i = \begin{pmatrix} l_{x_i} & l_{y_i} \end{pmatrix}^T$$

The 3 previously defined estimates are associated to their covariance matrices, respectively: $\mathbf{P_b}$, $\mathbf{P_v}$ and $\mathbf{P}_{\mathbf{l}_i}$. In the context of our monocular SLAM, the vehicle and bias poses are actually defined as a 6D model (3D position and the 3 associated angles). The definitions given in this paper easily extend to this model. However, we chose to present them in a planar environment as it simplifies the equations and eases the global understanding of our application.

The transformation from the biased frame to the consistent one is simple as the bias is additive. Let u be the unbiased frame. The vehicle estimate in this frame is defined by the following relationship:

$$\mathbf{v}_u = \begin{bmatrix} cos(b_\theta) & -sin(b_\theta) & 0 \\ sin(b_\theta) & cos(b_\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v} + \mathbf{b} \tag{1}$$

In a similar manner, the bias is taken into account for a landmark i thanks to this relationship:

$$\mathbf{l}_{u_i} = \begin{bmatrix} cos(b_\theta) & -sin(b_\theta) \\ sin(b_\theta) & cos(b_\theta) \end{bmatrix} \mathbf{l}_i + \begin{bmatrix} b_x \\ b_y \end{bmatrix} \tag{2}$$

As the drift grows along with the distance traveled, we chose to represent its evolution with an autoregressive model of order 1 (AR(1)) indexed on the curvilinear abscissa:

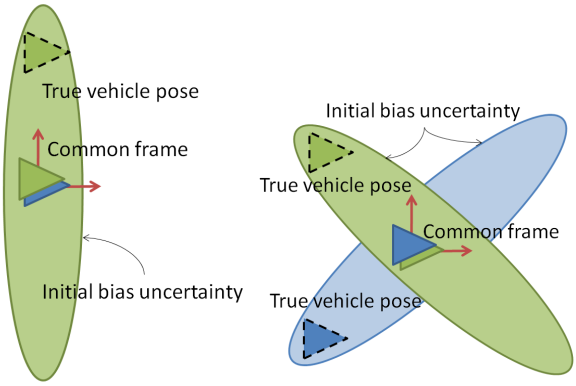
$$\mathbf{b}_s = \mathbf{b}_{s-1} + \boldsymbol{\varepsilon} \tag{3}$$

$$\mathbf{P}_{\mathbf{b}s} = \mathbf{P}_{\mathbf{b}s-1} + \Delta s \mathbf{P}_{\boldsymbol{\varepsilon}} \tag{4}$$

where s is the current curvilinear abscissa and $s-1$ the previous one. $\boldsymbol{\varepsilon}$ is a white noise, corresponding to the SLAM drift, that cannot be characterized. Nevertheless, thanks to equation (4), it is possible to integrate this lack of knowledge into the bias covariance matrix $\mathbf{P}_{\mathbf{b}s}$. Δs is the distance between $s-1$ and s and $\mathbf{P}_{\boldsymbol{\varepsilon}}$, is the uncertainty characterizing the noise $\boldsymbol{\varepsilon}$. This covariance matrix can be defined experimentally depending of the observed divergence.

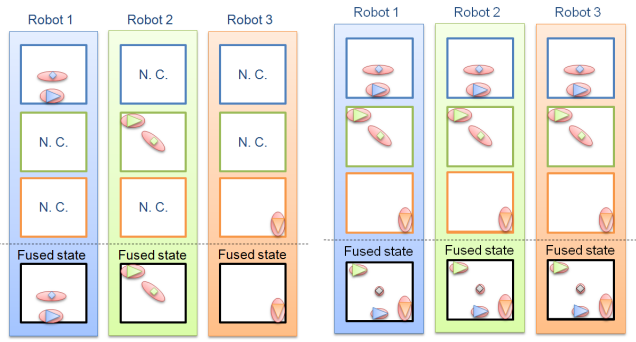
In a multi-vehicle context, it is important to bear in mind that each vehicle will be affected by a different bias, hence the fact that robots will have to share their respective biases. One important advantage of the model exposed in this section is that it can be initialized with any particular drift or uncertainty. In a multi-vehicle scenario, it is frequent to have an approximate idea of where a vehicle can be located compared to another one (or to a common frame). This approximation could drive the association process and facilitates the search for common portion of maps between vehicles. This knowledge can be included directly into the initial bias state if it is accurate. Otherwise, it can be used to give a first covariance matrix, coherent with the multi-vehicle setting. Two examples of initializations are given in Figure 2.

In the first case, the reference frame of the blue vehicle is considered as the common frame. As we do not know the distance between the vehicles, expressing the green vehicle in the common frame will put it at the exact same place as the blue robot. The initial bias covariance of the green vehicle allows it to include its true pose in this common frame. This first bias value will help the association algorithm and ensures that the consistency is maintained at a higher level. Indeed, with such an initialization, the decentralized state will always be consistent. In the second example depicted in Figure 2, the result is similar. The only difference is that the common frame is not the vehicle original reference. It forces each vehicle to



(a) The common frame is the one of the blue vehicle. Dashed triangle: true pose. Plain triangle: estimate.
(b) The common frame is a new one. Dashed triangles: true poses. Plain triangles: estimates.

Fig. 2. Examples of biases initializations



(a) Before communication (b) After communication

Fig. 3. Submaps separation to avoid data incest

start the trajectory with a specific bias. In both cases, if an accurate value of the distance between the vehicles is known, it can easily be integrated through the bias, thus avoiding to carry important uncertainties.

This initial value can be seen as a static bias which actually represents the distance separating the two vehicles (Figure 2(a)) or the distance to the common frame (Figure 2(b)). This static part of the bias will be refined when associations between different vehicles are found.

III. Multi-vehicle architecture

The decentralized structure must be able to integrate the bias. Indeed, it cannot be simply added to a classic SLAM algorithm as it would be mixed with the vehicle uncertainty and thus lowered by new observations. The multi-vehicle architecture must also avoid data incest. As a consequence, it is important to separate properly data coming from different vehicles.

One of the strategy frequently used to avoid data incest is to never exchange fused information. In [15], the authors chose to separate the states of the vehicles in substates. Those substates can freely be sent to other vehicles without fearing overconfidence. Indeed, when a new substate is received by a vehicle, it replaces the old one (each substate is timestamped to avoid erasing a newer substate). All the substates can be fused in order to obtain the global state. This fused state is never exchanged. As every vehicle is sharing the same substates, they are all able to compute the global state.

The approach described in [15] is actually designed for multi-vehicle localization. As we intend to provide a multi-vehicle SLAM, we chose to adapt this method to our needs. The core idea is the same: the maps coming from different vehicles are separated, forming submaps, and the algorithm never exchanges the global fused map. An example with 3 vehicles is given in Figure 3.

The main advantage of this method is that each vehicle will be able to have the same global map with relation to the others. By following a similar approach to [15], if an updated landmark was received, it should replace the old one in the submap. However, it would mean having to rebuild the whole map to integrate it, which is not a suitable solution. This is why only accurate landmarks are transmitted to submaps. A precise landmark will not have to be replaced thus allowing to only enhance the global map. Moreover, finding common map portions requires accurate landmarks not to perform false data associations.

Landmarks are transferred from a classic SLAM algorithm (often referred to as low-level SLAM), when considered accurate, to their submaps in the decentralized process. By splitting the low-level SLAM from the decentralized part of the algorithm, we are able to make it much more general. Indeed, the classic SLAM can use any sensor and filtering method as long as it furnishes accurate landmarks and their uncertainties along with vehicle information (pose and uncertainty). The whole architecture employed in this paper is illustrated with a 2-vehicle example in Figure 4.

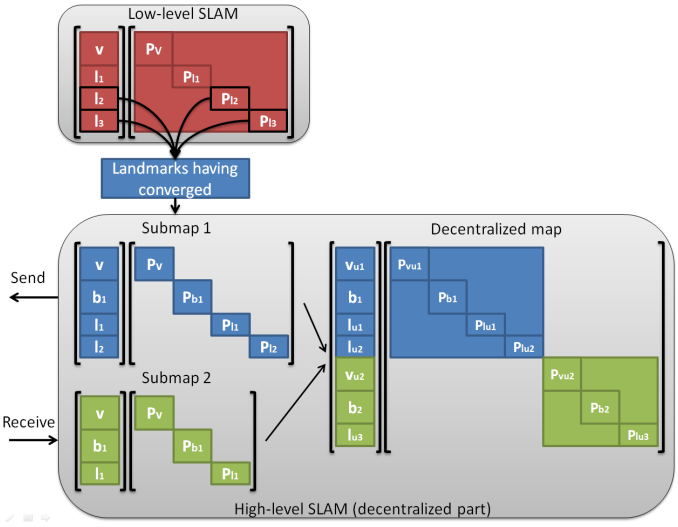


Fig. 4. Global architecture of the decentralized SLAM

The top part of the figure is the classic SLAM. When landmarks have converged (here, l_2 and l_3) they are transmitted to the high-level which adds them to the corresponding submap

(submap 1). This one can then be sent to other vehicles of the team. In this example, landmarks are coming from a second vehicle and received through the network. It allows the creation of a new submap (submap 2). Submaps can then easily be fused to get the global map.

With such an architecture, data received later than expected is not a problem. It will just be inserted into the corresponding submap. Data losses can also be countered. Each landmark is associated to a unique index in its own submap and is copied from the low-level SLAM once and for all. When another vehicle receives landmarks and sees that an index is missing in the corresponding submap, a request can be emitted to the concerned vehicle which will respond with the missing landmark.

One can notice that bias estimates are considered only in the decentralized part of the process. As landmarks are accurate within the low-level SLAM, almost all the error can be attributed to the SLAM drift. The direct consequence is that it is possible to connect all the landmarks of a submap to the different bias estimates and so between them. It is a major advantage in a multi-vehicle context as it means that cross-covariances do not have to be sent via the network. It can also be seen that each landmark, or vehicle pose, is not affected by a unique bias. It would normally be the case but we chose to have a common bias estimate for landmarks having converged at close times. This simplification is suitable as long as the distance between the initialization of new bias estimates is not too important. By doing so, we are able to lighten the computational burden along with the memory requirements.

Before a new point is added to the global map, it is first tested to check if it can be associated with one of the landmarks already in the global map. The data association process will be detailed in Section IV. In the case of an association between two landmarks coming from the same submap (one already in the global map while the other is not), it is similar to a loop closure. Indeed, it means that a previously mapped landmark has been recognized, thus closing the loop. As all the landmarks coming from this submap are linked together through their bias, an association will be able to estimate the drift impacting the localization and so reduce it for the whole trajectory without any special processing. On the other hand, an association between landmarks from different submaps will allow to put them in a common frame. It will refine the static bias estimate provided at the beginning of the trajectory and give an estimation of the drift affecting the vehicles involved. In both cases, the processing is the same.

When a point is not associated, it is simply added to the decentralized map (see Figure 4). The current bias estimate is taken into account thanks to a Kalman filter. This step will connect the new landmark to the rest of the global map. Readers interested in this point can refer to [16] for more details.

IV. Multi-vehicle data association

In SLAM algorithms, data association is often limited to the tracking of previously initialized features. It is usually done by projecting the landmarks back in the sensor space and finding the best matching. However, in a multi-vehicle context, data association at the high-level cannot be accomplished with such methods. Indeed, each map has its own frame and the relative distance between those frames is not known, making it impossible to project landmarks. Combined with the network constraint (amount of data to send limited), it is necessary to build an algorithm that is not based on the sensors used. Several approaches chose to start with a known distance between robots [17][1] thus limiting many potential multi-vehicle applications. In [18], the authors rely on a rendezvous (direct observation of the vehicles) to put the different maps in a common frame which is also restrictive.

Specific multi-vehicle data association processes have been developed. In [19], the authors propose a method based on the analysis of the 3 closest landmarks to each new point. Some measures (distance and angle) are then established. Two landmarks from two different robots sharing the same configuration will be considered for a fusion. A tree of possibilities is explored to determine the best matchings. However, it requires that several landmarks, in at least two different maps, have the same points surrounding them. This assumption is difficult to satisfy in dense environments where the features extracted can be different or at least not the same as the ones extracted with another vehicle. In [20], landmarks are brought together by 3 to form triangles. A Delaunay triangulation is computed to obtain these triangles. It has the advantage to give a unique map. Features such as the perimeter and the area of the triangle are then given to a RANSAC algorithm to find the best match between two maps. As for the previous approach cited, if the landmarks from the two maps are not same, the Delaunay triangulation will be too different to find proper matchings.

Loop closure algorithms can be considered for multi-vehicle data association if they can solve the kidnapped robot problem (given a map and observations, find where the robot is in the map). Indeed, the kidnapped robot problem is very close to associating landmarks from different robots. However, in our case, thanks to the bias integration, we can use landmarks uncertainties to help the association process. In that sense, the Joint Compatibility Branch and Bound (JCBB) [21] can be interesting if the bias uncertainty is not too important. The objective of JCBB is to find the set of jointly compatible landmarks according to Mahalanobis distances. If the bias uncertainty for vehicles is very important, almost every configuration of landmarks will match thus furnishing wrong data associations.

The Geometric Constraints Branch and Bound (GCBB) algorithm [22] follows a similar tree-approach but based on compatible distances between landmarks (see Figure 5). The idea is to test unary and binary constraints between new points and landmarks already in a map. The unary constraint can be defined depending on the sensor used, for instance two evaluated points must have similar surroundings etc. The binary constraint checks that the distance between two new points is similar to the one between two points from the map. The tree is defined and explored according to these criteria.

```
% called with: GCBB([ ], 0)
procedure GCBB(H, i)
    % H : current hypothesis
    % i : new landmark to test
    % m : number of new landmarks to test
    % n : number of landmarks in the map
    if i ≥ m then
        % better association found?
        if numPairings(H) > numPairings(best) then
            best = H
        end if
    else
        for j = 0 to n − 1 do
            if unary(i, j) & binary(i, j, H) then
                % landmarks i and j associated
                GCBB([H j], i + 1)
            end if
        end for
        % can do better?
        if numPairings(H)+m−(i+1) > numPairings(best)
then
            % pairing (i,j) refused
            GCBB([H −1], i + 1)
        end if
    end if
end procedure
```

Fig. 5. GCBB algorithm

At the end, the subset sharing the best resemblance with the map is returned. This approach is suitable for a multi-vehicle scenario because it only needs several points to be common between the two overlapping maps to find a match.

The main drawback of this algorithm is the time required to explore the tree. This time can be significantly reduced by pruning the tree before going through it. To do so, we take advantage of the consistency provided by the bias. Indeed, thanks to it, we can reduce the number of points that can potentially match a landmark to those which are individually compatible with it. As the bias is taken into account into the whole decentralized map, this constraint will work with loop closures (one vehicle concerned) and when repositioning two maps in a common frame (multiple vehicles concerned). Individual Compatibility (IC) is computed as follows:

$$D_{ij}^2 = \nu_{ij}^T \mathbf{C}_{ij}^{-1} \nu_{ij} < \chi_{d,\alpha}^2 \tag{5}$$

where:

$$\nu_{ij} = \mathbf{l}_{u_i} - \mathbf{l}_{u_j}$$
$$\mathbf{C}_{ij} = \mathbf{P}_{\mathbf{l}_{u_i}} + \mathbf{P}_{\mathbf{l}_{u_j}}$$

$\chi_{d,\alpha}^2$ is the Chi-squared distribution. It defines the maximum threshold of acceptation. In our case, $d = 2$ and α, the desired confidence, is set to 0.95. The landmarks that satisfy IC are the ones that will compose the tree. The distances between those landmarks are then computed before starting the GCBB process so as to avoid to compute several times the same data.

The architecture, presented in Section III, suits the decentralized data association process well. Indeed, with all the landmarks connected to their respective biases estimates in the global map, the algorithm just has to check landmarks arriving in a submap with relation to this global map. If they cannot be associated, the landmarks will be added to the global map, otherwise a fusion will be triggered. As explained before, two vehicles traveling through the same area will not necessarily map the same landmarks. Consequently, we decided that landmarks arriving in a submap are kept for a certain time in order to feed the association process before being added to the global map. By doing so, we are also able to increase the minimum number of landmarks necessary to validate a fusion which, as a consequence, increases its likelihood.

Figure 6 shows some results of this data association process based on real data. Here, it consists of a loop closing: a vehicle started a trajectory and then came back to where it started (150 meters later) while still mapping landmarks. The different landmarks mapped during the common part of the trajectory and the associations found are visible. The gap between the two turns of the trajectory is due to the natural SLAM drift.

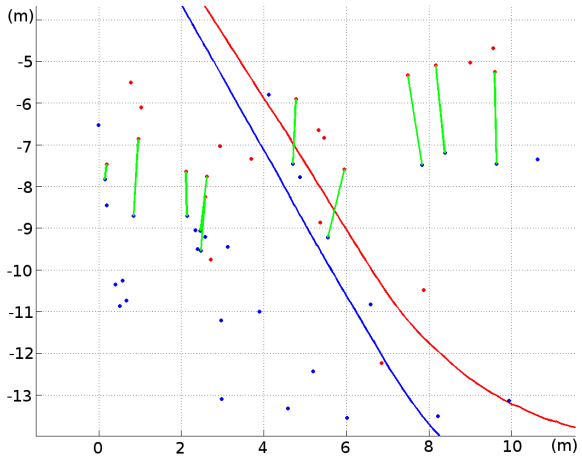


Fig. 6. Associations found in the case of a loop closing. The blue landmarks are mapped during the first passage (blue trajectory) of the vehicle and the red ones during the second passage (red trajectory). The green lines are the matchings found through the method described in this section.

V. Experiments

The following experiments were realized thanks to a simulator for convenience purposes. However, it is important to note that the simulator presents a realistic physics. Besides, the environments available are mapped from real locations. The simulator is only used to generate sensor data in real time and does not perform any other processing than that. The algorithm is totally separated and is ran by other computers. Real network exchanges are performed as an instance of the application is executed for each vehicle in the team. Our decentralized algorithm was running in real time. Some pictures of the environment are visible in Figure 7.

(a) Environment used (b) Environment used (c) Camera output (d) Camera output

Fig. 7. Different illustrations of the simulator and its environment

In the case of our experiments, we used a 2-vehicle setting. Both were equipped with a camera running at 10Hz, an odometer and RTK GPS only used for comparison purposes. They were moving at 2 meters per second. The SLAM algorithm was running in real time for both vehicles (two instances). TCP/IP packets were sent to communicate the different landmarks and vehicle poses. The low-level SLAM is based on an Extended Kalman Filter. It is part of previous works on the SLAM topic and it is detailed in [23]. In a nutshell, EKFs have the advantage to be light with relation to the number of landmarks required. This aspect is very important as it allows to limit the amount of data sent via the network to a very acceptable level. The trajectories accomplished are shown in Figure 8.

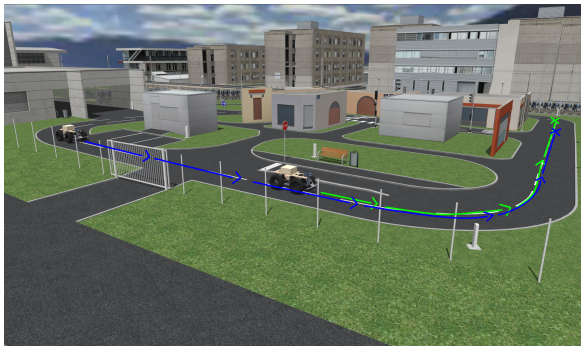


Fig. 8. Overview of the trajectories accomplished by the two vehicles

In the experiments presented here, another unary constraint has been added to the association algorithm (see Section IV). This one is not mandatory but helps the overall process. As we are using a monocular SLAM, we chose to keep the patches (pixels extracted around the image features) used to track landmarks in the low-level. When a new landmark arrives in the decentralized part of the algorithm, its patch is tested against the ones of the landmarks in the global map. In order to be associated together, the landmarks must have similar patches.

Both vehicles are separated by 20 meters. With such a setting, it is difficult to localize one vehicle with respect to the other. We chose the starting point of the front vehicle as a common frame in a similar way to Figure 2(a). In this configuration, the back vehicle is not accurate in the frame of the leading one. The objective is so to find common points in order to estimate the space separating the vehicles. As the results obtained by both vehicles are similar (same processing, based on the same information), we will only present those coming from the front vehicle. Still, as it is a decentralized process, it has received the information shared by the other robot (landmarks and position). Figure 9 shows the two SLAM trajectories perceived by the front (green) vehicle.

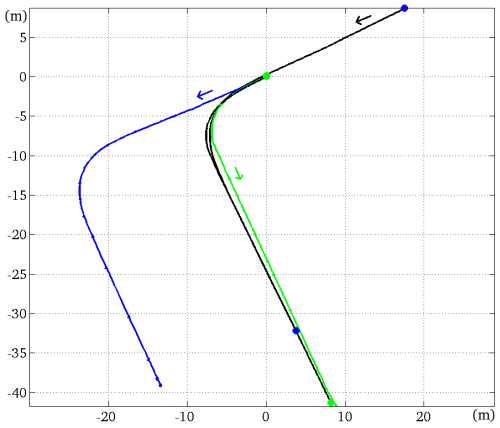


Fig. 9. SLAM-only trajectories perceived by the green vehicle. In blue is the trajectory of the back vehicle and in green the one of the leader. In black is the ground-truth associated to both trajectories. The colored dots indicate the beginnings and endings of the true trajectory corresponding to the color.

In this figure, we can notice that the back vehicle is brought 20 meters forward compared to its ground truth. It was expected as we try to put the localization given by this vehicle (with relation to its starting point) in the common frame, here defined as the one of the front vehicle. Without the bias integration and the data association presented, this distance cannot be estimated, hence the results furnished by the SLAM algorithms. The small gap visible between the green vehicle estimation and its ground truth is bound to the SLAM drift. The uncertainty associated to the estimates do not make the localization consistent as they are too small. Figure 10 shows the results of the front vehicle SLAM when taking into account the drift. It also provides a zoom on the trajectory to show the original SLAM uncertainty.

When integrating the bias, we can see that the consistency of the localization is guaranteed. The zoom on the original uncertainty also shows how far from consistency can be a SLAM algorithm. In order to initialize the static bias of the other vehicle (queue), we chose to give it an approximate uncertainty ensuring that it becomes consistent compared to its

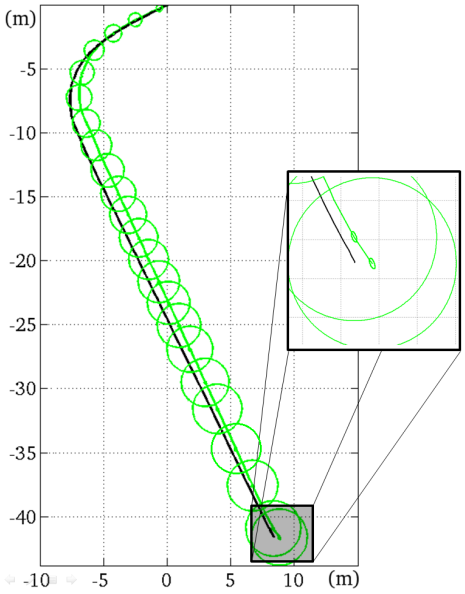


Fig. 10. SLAM-only trajectory of the front vehicle with the bias integration. The green line is the SLAM trajectory and the black one the ground truth. The big green ellipses represent the uncertainty with bias integration. The small green ellipses visible in the zoom are the uncertainty of the SLAM process.

ground truth. The vehicle is located at a little bit less than 20 meters from its decentralized frame of reference. We initialized the bias uncertainty with a 22-meter circle around its estimate. Similarly to Figure 10, Figure 11 shows the results of the back vehicle with the bias integration.

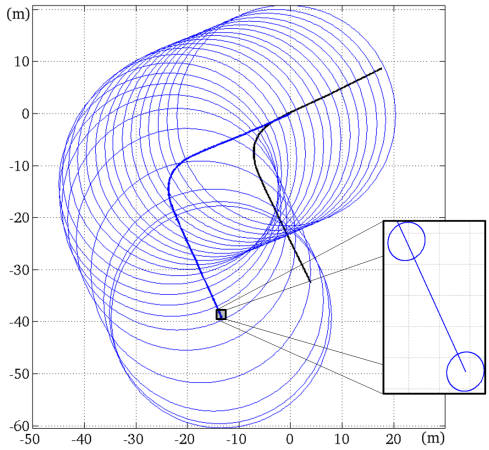


Fig. 11. SLAM-only trajectory of the back vehicle with the bias integration. The blue line is the SLAM trajectory and the black one the ground truth. The big blue ellipses represent the uncertainty with bias integration. The small blue ellipses visible in the zoom are the uncertainty given by the SLAM process.

As with the previous vehicle, the consistency is ensured. The research area for matchings will still be very important and will consequently rely a lot on the data association algorithm. The zoom also demonstrates that it is impossible without integrating an initial bias (here, in the form of the uncertainty) to achieve consistency in such a setting. Figure 12 shows the results when associations are sought between incoming landmarks and the decentralized map.

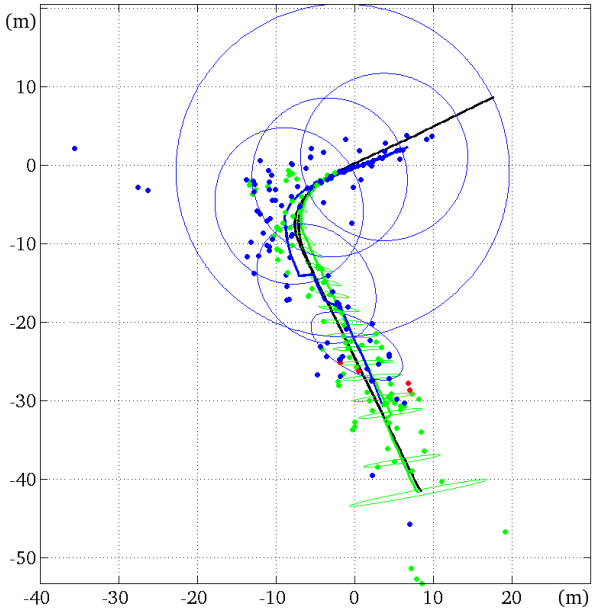


Fig. 12. Decentralized trajectories of both vehicles. The green line is the trajectory of the front vehicle. The blue one is the trajectory of the back vehicle. Dots represent landmarks. Those in red are the ones found common to both vehicles. The black trajectories represent the ground truth. The ellipses are the uncertainty associated to each vehicle.

For the sake of clarity, only a few uncertainty ellipses are displayed in this figure. We can see that the blue vehicle is able to estimate its distance to the green one. The uncertainties are thus greatly reduced when the association is found and the consistency of the localizations is still achieved. It can also be noticed that only a few landmarks were mapped in both trajectories. It made the task of the association process harder as there are not many common points between both submaps. The amount of data sent by the two vehicles is visible in Figure 13.

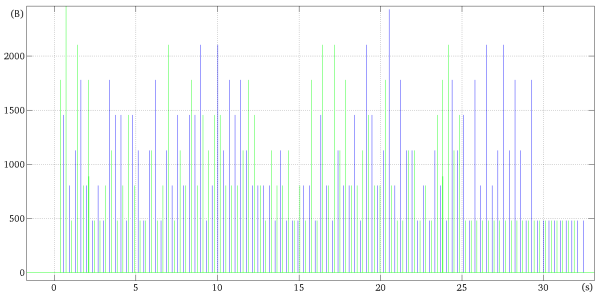


Fig. 13. Amount of data sent according to the elapsed time. In green are the results (in bytes) for the front vehicle and in blue for the back vehicle.

The quantity of data sent is very far from the technological limits (less than 10 KB of data cumulated over a second for a vehicle). Expanding the number of vehicles will consequently not be a problem regarding network aspects. Concerning the quality of the computed localizations, it is possible to compare the distance between the two vehicles given by the ground

truth to the one provided by our decentralized algorithm. These results are depicted in Figure 14.

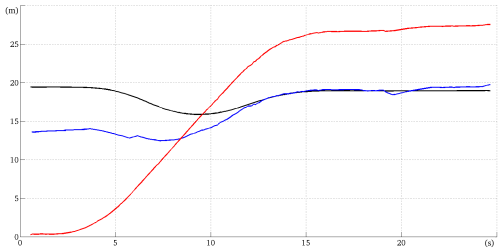


Fig. 14. Distance between both vehicles during the trajectory. In black is the true distance separating the vehicles. In red is the one computed from the SLAM results. In blue is the distance computed thanks to the decentralized localizations. The closer to the black curve, the better the results are.

Once the associations have been found, the distance computed by the decentralized algorithm comes very close to the ground truth. At the beginning of the trajectory, the distance estimated in the decentralized algorithm is still quite far from the real distance. This is due to the fact that associations have been found near the end of the trajectory. An association found sooner would have avoided this. To do so, it would be necessary to increase the number of landmarks tracked in the images in order to have more landmarks in the global map.

VI. Conclusion

In this paper, a multi-vehicle decentralized SLAM process has been exposed. Several innovations are proposed to achieve this goal. We introduced a model of the drift affecting every SLAM algorithm. It has the advantage to make the vehicle localization consistent. In a multi-robot context, the bias is also used to estimate the distance between vehicles when it is unknown. By providing initial values or uncertainties to the vehicles, we are able to put them in a common frame and drive the association process.

An architecture, built for a multi-vehicle usage, has been presented. By separating the data coming from the different vehicles of a team in submaps, we are able to avoid data incest which leads to overconfident estimations. This architecture can be plugged into any SLAM algorithm. A multi-robot data association process has also been introduced. It is based on a Geometric Constraints Branch and Bound algorithm adapted to our own constraints.

This approach has been applied to a monocular EKF-SLAM. The experiments were realized in real time, using data from a realistic simulator. The results demonstrate good localization results and lay the foundation for further developments. The number of landmarks available is the key element of our method success. However, it is also tightly bound to the computational time required. For future work, we plan to enhance this aspect and present longer trajectories with real data.

References

[1] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated Multi-Robot Exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[2] A. Kleiner and D. Sun, "Decentralized SLAM for Pedestrians without direct Communication," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 1461–1466.

[3] H. Li and F. Nashashibi, "Multi-vehicle Cooperative Perception and Augmented Reality for Driver Assistance: A Possibility to 'See' Through Front Vehicle," in *IEEE International Conference on Intelligent Transportation Systems*, 2011.

[4] T. A. Vidal-Calleja, C. Berger, J. Solà, and S. Lacroix, "Large Scale Multiple Robot Visual Mapping with Heterogeneous Landmarks in Semi-structured Terrain," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 654–674, 2011.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[6] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman, and H. Durrant-Whyte, "Decentralised Cooperative Localisation for Heterogeneous Teams of Mobile Robots," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2859–2865.

[7] M. Pfingsthorn, B. Slamet, and A. Visser, "A Scalable Hybrid Multi-Robot SLAM Method for Highly Detailed Maps," in *RoboCup 2007: Robot Soccer World Cup XI*, 2007, pp. 457–464.

[8] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte, "Towards Multi-Vehicle Simultaneous Localisation and Mapping," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 2743–2748.

[9] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully Distributed SLAM using Constrained Factor Graphs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[10] A. Martin and M. R. Emami, "Just-in-time Cooperative Simultaneous Localization and Mapping," in *International Conference on Control, Automation, Robotics and Vision*, 2010, pp. 479–484.

[11] R. Aragues, J. Cortes, and C. Sagues, "Dynamic Consensus for Merging Visual Maps under Limited Communications," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 3032–3037.

[12] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 2001.

[13] S. Julier and J. Uhlmann, "Building a Million Beacon Map," in *Sensor Fusion and Decentralized Control in Robotic Systems IV*, vol. 4571, 2001, pp. 1–9.

[14] C. Estrada, J. Neira, and J. D. Tardós, "Hierarchical SLAM: real-time accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[15] N. Karam, F. Chausse, R. Aufrère, and R. Chapuis, "Localization of a Group of Communicating Vehicles by State Exchange," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 519–524.

[16] G. Bresson, R. Aufrère, and R. Chapuis, "Loop Closing in a Drift-Aware Monocular SLAM," in *IFAC Intelligent Autonomous Vehicles Symposium*, 2013, accepted.

[17] E. Nettleton, S. Thrun, H. Durrant-Whyte, and S. Sukkarieh, "Decentralised SLAM with Low-Bandwidth Communication for Teams of Vehicles," in *Field and Service Robotics*, 2006, pp. 179–188.

[18] X. S. Zhou and S. I. Roumeliotis, "Multi-robot SLAM with Unknown Initial Correspondance: The Robot Rendezvous Case," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 1785–1792.

[19] S. Thrun and Y. Liu, "Multi-Robot SLAM With Sparse Extended Information Filers," in *11th International Symposium of Robotics Research*, 2003, pp. 254–266.

[20] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, "Fully Distributed Scalable Smoothing and Mapping with Robust Multi-robot Data Association," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 1093–1100.

[21] J. Neira and J. D. Tardós, "Data Association in Stochastic Mapping Using the Joint Compatibility Test," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, 2002.

[22] J. Neira, J. D. Tardós, and J. A. Castellanos, "Linear time vehicle relocation in SLAM," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 427–433.

[23] G. Bresson, T. Féraud, R. Aufrère, P. Checchin, and R. Chapuis, "Parsimonious Real Time Monocular SLAM," in *IEEE International Conference on Intelligent Vehicles*, 2012, pp. 511–516.