A General Consistent Decentralized Simultaneous Localization And Mapping Solution

Guillaume Bresson, Romuald Aufrère, Roland Chapuis

00 (2018) 1–20

A General Consistent Decentralized Simultaneous Localization And Mapping Solution

Guillaume Bresson[a,*], Romuald Aufrère[a,b], Roland Chapuis[a]

[a]*Institut Pascal - UMR 6602 CNRS, Clermont Université, Université Blaise Pascal - Aubière, France*
[b]*LIMOS - UMR 6158 CNRS, Clermont Université, Université Blaise Pascal - Aubière, France*

Abstract

In this paper, we propose a new approach to the decentralized Simultaneous Localization And Mapping (SLAM) problem. The goal is to demonstrate the feasibility of decentralized localization using low-density maps built with low-cost sensors. This problem is challenging at different levels. Indeed, each vehicle localization tends to drift over time independently of one another making the global localization of a fleet hard to achieve. To counter this effect, called SLAM inconsistency and which has been stated numerous times in the literature, we introduce a model to represent the natural drift of SLAM algorithms. Its integration inside an Extended Kalman Filter is explained along with simulations validating its use. The second part of this paper presents the fusion architecture designed to solve the different problems arising in a decentralized scheme. It avoids data incest, which is an important source of inconsistency, and integrates the previously mentioned SLAM drift in the estimates produced. This architecture also separates the SLAM classically used for mono-vehicle applications from the high-level decentralized part offering flexibility regarding sensors and algorithms at a low-level. Other aspects, involved by the multi-vehicle settings, are also taken into account (communication losses, latencies, desynchronizations, unknown initial positions of the fleet members and data association). The whole algorithm has been tested in various scenarios with vehicles equipped with a single camera and an odometer. The results, from both simulated and real scenarios, show that our approach can work in real time with very small bandwidth requirements.

Keywords:
multi-vehicle, decentralized, SLAM, EKF, monocular, drift, bias, consistency

1. Introduction

Applications involving autonomous vehicles have been vastly studied among the mobile robotics community over the last decades. The focus was mostly on the perception systems and automatic guidance of a single vehicle. While some great results have been demonstrated, many applications are yet to be tackled as they require a fleet of cooperating vehicles. The emergence of fast and reliable communication means for Intelligent Transportation Systems could heavily contribute to the expansion of these collaborative approaches in the near future. Moreover, it is also a first step towards self-driving cars which will certainly require to share information to ensure safety.

Some authors have already investigated potential applications and showed impressive results. We can cite: fast exploration [1][2], localization without direct observation [3], risk assessment in dangerous situations [4], augmented reality for coordinated transportation [5], augmented reality for cellphones [6], improved localization accuracy [7], dense 3D reconstruction distributed over smartphones [8]... In the literature, multi-vehicle algorithms have been applied to field robotics [5], indoor environments [3], AUVs [9], UAVs [10] and heterogeneous teams [11].

Many of the examples cited above rely on vehicles able to localize themselves in unknown environments and share their poses with the rest of the team. One of the most common method to do so is for the vehicle to build a map of the environment while simultaneously localizing itself inside this map. Simultaneous Localization And Mapping (SLAM) has been widely used in single-vehicle applications [12][13] and naturally extends to multi-vehicle scenarios. Indeed, a decentralized approach can quicken the mapping of an area [14] and has also been proven to provide a better localization accuracy [15] than mono-vehicle SLAM systems.

Nevertheless, very little algorithms have been designed to address all the issues arising when dealing with several vehicles. Indeed, there are many aspects to consider when devel-

*Corresponding author
Email addresses: guillaume.bresson@univ-bpclermont.fr (Guillaume Bresson), romuald.aufrere@univ-bpclermont.fr (Romuald Aufrère), roland.chapuis@univ-bpclermont.fr (Roland Chapuis)

oping a multi-vehicle process (data incest, unknown initial positions, bandwidth requirements, data association...) that are tightly bound to the application aimed.

Furthermore, certain issues, already present in single-vehicle systems, become an overriding concern in a multi-vehicle scheme. It is, for example, the case of the natural SLAM drift as each vehicle will drift differently from one another. This problem has already been stated several times [16][17] and remains an open question. Even though some methods have been developed to decrease the influence of this drift [18], none of them have addressed its modeling and integration inside existing filtering methods.

In this paper, we present a complete algorithm suited to a multi-vehicle use [19]. Our objective is to propose an approach that can work with any feature-based SLAM. Here, we apply our algorithm to a low-cost monocular EKF-SLAM solution. Our contributions are the following:

- The development of a decentralized SLAM system that ensures the consistency of the computed localization at a global level (vehicle fleet) and takes into account the communication aspects involved by multi-vehicle applications (quantity of data sent, latencies, desynchronizations, communication losses).
- A general framework that can be applied to various feature-based SLAM algorithms and that is applied here to a monocular setting for the first time to our knowledge.
- The integration of a model representing the natural SLAM drift which guarantees the integrity of the localization provided by the mono-vehicle (low-level) SLAM, allows the use of absolute information (GPS data, geo-referenced landmarks) and can perform loop closing seamlessly.
- The real time application of our approach to various scenarios (simulated and real) with 2 or 3 vehicles.

The rest of this paper will be organized as follows: Section 2 will present the state of the art regarding multi-vehicle SLAM (Subsection 2.1) and the mono-vehicle constraints on the decentralized part (Subsection 2.2). Then, Section 3 will be about the SLAM divergence. The model used to represent the drift will be exposed (Subsection 3.1) followed by its integration into a SLAM algorithm (Subsection 3.2). Some simulation results concerning its single-vehicle application will be presented (Subsection 3.3). Section 4 will first introduce the architecture built to cope with the decentralized aspects (Subsection 4.1) and how unknown initial positions of vehicles are dealt with thanks to the static part of the drift model (Subsection 4.2). Subsection 4.3 will expose the data association used in this multi-vehicle context. Finally, in Section 5, the experiments will be presented. Results from a realistic simulator (Subsection 5.1) and real data (Subsection 5.2) will be discussed.

2. State of the art

2.1. Multi-vehicle localization

One major aspect when building an algorithm for a fleet of vehicles is choosing between a distributed or centralized scheme. This choice has a big impact on the design of the whole algorithm and the communication strategy. A centralized method is easy to implement as all the vehicles only need to send information to a centralizing entity. This central node (a vehicle or a dedicated computer) just has to fuse all the incoming information and share the resulting map. This scheme has been used in many algorithms [11][20] due to its simplicity. However, it suffers from significant drawbacks. First, in case of failure, the vehicules are unable to access the global map. Other than robustness, the communication requirements for a centralized approach are higher than for a decentralized one. Indeed, the global map, only managed inside the central node, must be regularly sent to all the vehicles of the team. Depending on the size of the map, it can be difficult to achieve within a satisfying amount of time. Last but not least, the centralized scheme forces every vehicle to be at reach of the entity handling the global fusion. With a distributed approach, each vehicle is able to relay the information through the whole network.

2.1.1. Data incest

Whatever the strategy, it is necessary to be careful about the way data coming from different vehicles are handled. Double-counting information can cause the overconfidence of the estimates produced (vehicle poses and landmark positions). This aspect, also known as data incest, is a common topic in the multi-vehicle community. It has already been stated in cooperative localization [21][22] and some solutions have been proposed [23][24]. When building a common map, a special attention must be given to this issue as landmarks are frequently updated and could easily be mixed in the estimation process. A 4-step example is shown in Figure 1 in which a landmark estimate is used twice.

In [25], the authors use a dedicated network architecture (communications are limited to neighbors) to avoid data incest. The authors of [26] only exchange submaps once they are closed (not updated anymore). Each data is consequently sent only once thus preventing data incest. In the approach presented in [11], the authors solely exchange a high-level (topological) map which keeps tracks of the different submaps. The topological map has the advantage to be light and can be easily replaced when outdated. In [27], a graph-based approach is presented. The map of each vehicle is compressed and sent to neighbors. A cache filters the maps already exchanged. The common map is then computed between neighbors based on common landmarks. Similarly, in [28][29], a consensus is sought between neighbors in order to find the best common map that avoids double counting information.

In this paper, we present a new approach, based on previous works on cooperative localization [24], which requires no dedicated network architecture and provides a near-optimal global map at every moment as opposed to the previously cited methods (see Section 4).

2.1.2. Communication constraints

Ideally, a decentralized algorithm should be independent of the number of vehicles involved regarding bandwidth requirements [25]. However, it requires a limiting dedicated network

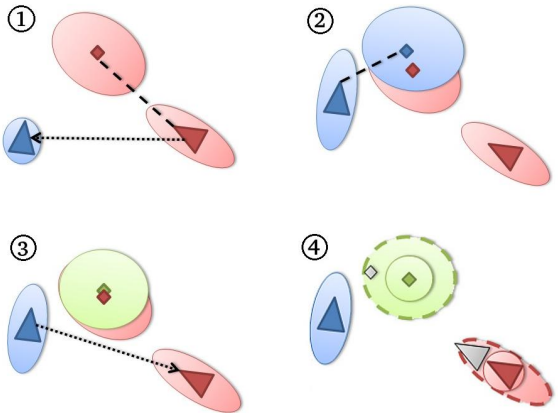


Figure 1. An example of data incest where a landmark estimate is taken into account twice. Triangles represent vehicles and diamonds, landmarks. **1.** A new landmark is mapped by the red vehicle. It is then exchanged to the blue one. The ellipses correspond to the localization uncertainties. **2.** A new landmark is mapped by the blue vehicle. **3.** Both landmarks actually represent the same one, they are consequently fused (green landmark). The result is exchanged to the red vehicle. **4.** The red vehicle finds that the received landmark and its own can be fused together (same landmark). However, the received landmark already integrates the estimate computed by the red vehicle. The result is overconfident and the true positions (gray triangle for the vehicle and gray diamond for the landmark) are outside of the estimated uncertainties. The true uncertainties are the ones in dashed lines.

architecture to fulfill this condition. Instead of this constraint, we chose to minimize the quantity of information to send in order to have the smallest bandwidth needs possible and so a potentially large number of vehicles communicating together. This aspect is often neglected in the literature despite its fundamental implication on the design of the algorithm [26][30] and is mainly raised in approaches only exchanging high-level topological maps which are light [11][27][31].

Apart from the bandwidth needed, communication losses, latencies or desynchronizations can occur. When they are not handled, communication losses often lead to data that is definitely lost [30]. With an exchange strategy based on entire submaps, it can be problematic [26]. Algorithms built around neighbors communications are usually able to request missing data as it is done at a more local level [27]. Similarly, in [32], the authors used Particle Filters to apply observations no matter the order, thus allowing to solve the data loss case. Latencies and desynchronizations are naturally avoided by using the inverse form of the Kalman Filter. Indeed, with the Information Filter, the update steps become additive, making the order in which they are performed meaningless [9][33][34]. Other approaches chose to exchange every piece of information only once in order to make latencies harmless [26][30].

Communication constraints are often neglected because of the complexity of rolling out multi-vehicle applications. Most of the time, multi-vehicle results come from Matlab simulations [15] [22][27][35].

2.1.3. Data association

In classic single-vehicle SLAM algorithms, data association is mostly a tracking issue. Projecting a previously initialized landmark in the sensor space and finding the best matching is sufficient for most applications. Nevertheless, in a multi-vehicle context, such algorithms are difficult to use as they involve a heavy communication constraint in order to be able to project every piece of data back in the sensor space of each vehicle. Moreover, the fact that the location of vehicles with relation to each other is usually unknown makes it even harder. Data association algorithms based on sensor features should be avoided in a multi-vehicle context. In [1][25], the authors start with known distances between the vehicles which limits the potential applications. The authors of [36] define a rendezvous in order for each vehicle to have a direct observation of the other ones. In [37] and [38], a GPS is first used to have a rough estimate of the localization of the fleet. Then, grid maps are matched thanks a genetic algorithm so as to refine the prior knowledge provided by the GPS.

Among the data association processes developed for the multi-vehicle context, we can cite [34] in which the authors propose to analyze the 3 closest landmarks of each point. Thanks to some measures (distances and angles), it is then possible to associate together landmarks sharing the same configuration. Similarly, in [39], landmarks are regrouped by 3. A Delaunay triangulation is performed in order to obtain a unique map. The perimeters and areas of these triangles are then provided to a RANSAC algorithm which aims at finding the best matching between two maps from two different vehicles. These approaches require that the landmarks mapped by 2 different vehicles are almost the same.

Data association algorithms able to solve the kidnapped vehicle problem (locate a lost vehicle given a map and observations) are interesting as they can be adapted to multi-vehicle applications. The Joint Compatibility Branch and Bound (JCBB) [40] method could be considered if an accurate enough idea of the distance between vehicles is known beforehand. Indeed, JCBB consists in finding a set of jointly compatible landmarks with relation to Mahalanobis distances. However, with high uncertainties concerning the poses of the different vehicles, this method would be inefficient and would mostly provide wrong matchings. An alternative is the Geometric Constraints Branch and Bound (GCBB) [41]. The idea is to define constraints between pairs of landmarks from the map and new observations. A tree of possible pairings is then established and explored. The most likely subset of correspondences between the map and observations is returned. The main advantage of this method is that geometric constraints are robust when considered for a set of landmarks (instead of associations on a case by case basis). The other interesting aspect of this algorithm is that it only requires several common landmarks to find proper associations between two maps.

2.2. Mono-vehicle constraints on the decentralized part

2.2.1. Low-level SLAM

To be able to decorrelate the decentralized part of the algorithm from the sensors used, the classic SLAM process (called low-level from now on) must be independent of the rest. However, there is a wide variety of SLAM algorithms, making this

constraint hard do satisfy. We propose here an abstraction for feature-based SLAM as they are commonly used.

Although it is essential for multi-vehicle SLAM systems to be as much as possible independent of the sensors used, it is necessary to keep in mind that decentralized algorithms should not rely on expensive sensors as the cost is multiplied by the number of vehicles composing the fleet. However, sensors must be discriminative enough so as to track landmarks and recognize previously mapped areas. Cameras seem to be a good compromise between cost and information-richness. Monocular SLAM solutions have been widely studied [42][43] for these reasons. Consequently, we opted for a low-level SLAM based on a single-camera system and an odometer which are fused in an Extended Kalman Filter. A Harris detector provides the different features that are then tracked in images in order to have accurate 3D landmarks (30 points are tracked per image). It is these points that will then be used in the decentralized part of the algorithm to find common map portions between vehicles. Interested readers can refer to [44] and [18] for more information about the low-level algorithm. It is, to our knowledge, the first time that a fully decentralized monocular SLAM algorithm is presented.

2.2.2. SLAM inconsistency

One of the main problem of SLAM algorithms is inconsistency (true position of the vehicle or a landmark outside of the estimated uncertainty) [16][17][45]. One of the source of inconsistency is correlated to the linearization involved by the use of non-linear models. It causes SLAM-based methods to drift over time [46], making the filter compute biased vehicle localizations and so divergent landmark estimates. Consistency cannot be guaranteed [47]. With optimization approach, non-linear solvers are also affected and can give wrong results under the form of local minima [39]. Another source of inconsistency has been notified by Julier and Uhlmann in [48]. They showed that considering every piece of sensor data independently of the previous one can also cause inconsistency. Of course, using sensors from whose errors tend to accumulate over time (odometry for instance) will also cause this phenomenom.

The drift is closely linked to the distance traveled. The longer it is, the more significant will be the bias affecting the localization. Experimental results also show that the error affecting the orientation makes the filter diverge more than a position inaccuracy [49][50].

Most of the work so far has consisted in finding a way to avoid sources of inconsistency. In [51], for example, the authors create, for a single trajectory, several maps, called submaps. When linearization errors become too important (high uncertainty for the vehicle), the current submap is closed and a new one is initialized. The reference frame is changed, making it free from the previous drift. A global map keeps track of all the submaps and the links between them. At a local level, the drift is thus limited, but it is not the case when considering the global map. Smoothing approaches [52][53] propose to keep all past vehicle poses to re-linearize the state when necessary. While it allows for more consistent results, it requires to keep (and exchange in our multi-robot setting) all past poses and information needed to perform re-linearization which can be costly when done regularly. Covariance Intersection [54] is one way to model unknown correlations between landmarks and the vehicle pose. However, it usually implies that the results will be globally pessimistic. Another idea to avoid inconsistency was proposed by Roumeliotis et al. in [55]. In this paper, the authors proposed not to model the dynamic behaviour of the vehicle but only to consider the sensors used. The main advantage is that it allows to directly model the errors affecting the sensors (responsible for attitude estimation here) with linear equations. Nevertheless, it does not mean that cumulative errors are avoided.

In many algorithms, divergence is corrected thanks to loop closing [56]. It usually consists of a separated process (because of the inconsistency of the estimates) whose role is to detect previously visited locations. When such a match is found, a part of the drift can be computed thanks to the distance between the vehicle estimate during the first passing and the second. However, the drift is only partially corrected and the results are still overconfident [17][57]. In [58], once the loop closure is identified, errors are redistributed in a probabilistic manner around the past trajectory. It allows to avoid overconfidence but it still does not guarantee consistency. Another solution is to integrate absolute information (GPS positions for instance). Indeed, it has the advantage to be drift-free as it does not refer to a relative frame. Nevertheless, a low-cost GPS can be affected by the surroundings, especially in urban environments. In this case, geo-referenced infrastructures with or without communication capabilities can be an alternative [59]. With the emergence of the 802.11p standard, Vehicle To Infrastructure (V2I) or Vehicle To Vehicle (V2V) communications can be considered as a viable solution for collaborative approaches in the coming years. Otherwise, a simple database of geo-referenced landmarks can help to reduce the drift [60]. The authors of [61] proposed to search geo-referenced satellite images in a camera to improve the vehicle localization. It is an interesting way to constrain the divergence.

Nevertheless, estimating the drift at one moment does not allow to represent it at any time. As a consequence, in this paper, we chose to develop a model representing the SLAM drift. We integrate it in an EKF so as to estimate it together with the vehicle state and the map. Still, when it is possible, the methods cited above can be used to occasionally estimate the drift.

3. SLAM divergence

3.1. Bias model

The inconsistency of SLAM algorithms creates a gap between where the vehicle thinks it is and its true location. This error is not covered by the uncertainty (covariance matrix) associated to the vehicle estimate. This gap can be seen as a bias concerning the vehicle localization. The bias affecting the vehicle depends on the length of the path traveled and is thus tightly linked to the current curvilinear abscissa (distance traveled).

The direct consequence is that landmarks mapped at different moments will be subject to different bias values (the same applies to vehicle poses).

We define the bias as the 3 parameters (translation and rotation) expressing the difference between the real pose and the estimated pose of the vehicle:

$$\mathbf{b}_s = \begin{pmatrix} b_{x_s} & b_{y_s} & b_{\theta_s} \end{pmatrix}^T$$

where $(b_{x_s}\ b_{y_s})$ is a position bias and b_{θ_s} an angular one. The choice of this 2.5D representation is mainly motivated by the fact that the drift almost exclusively concerns these 3 parameters in the case of a still camera. Of course, the equations that follow can be extended to 6D.

This bias estimation is indexed according to the current curvilinear abscissa s. $\mathbf{b}_s$ is associated to its covariance matrix $\mathbf{P}_{\mathbf{b}_s}$ which indicates the uncertainty regarding the localization divergence.

Let s be the curvilinear abscissa associated to the moment k. The estimated and biased pose (2D position and orientation) of the vehicle $\mathbf{v}_k = \begin{pmatrix} x_k & y_k & \theta_k \end{pmatrix}^T$ can be corrected with the bias $\mathbf{b}_s$. The unbiased pose of the vehicle $\mathbf{v}_{u_k}$ can then be expressed as follows:

$$\mathbf{v}_{u_k} = \begin{bmatrix} \cos(b_{\theta_s}) & -\sin(b_{\theta_s}) & 0 \\ \sin(b_{\theta_s}) & \cos(b_{\theta_s}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v}_k + \mathbf{b}_s \tag{1}$$

The bias integration can also be done for landmarks. For a landmark position $\mathbf{l}_{i_k} = \begin{pmatrix} \mathbf{l}_{i_{x_k}} & \mathbf{l}_{i_{y_k}} \end{pmatrix}^{\mathrm{T}}$, affected by a bias $\mathbf{b}_s$, its unbiased version can be computed:

$$\mathbf{l}_{u_{i_k}} = \begin{bmatrix} \cos(b_{\theta_s}) & -\sin(b_{\theta_s}) \\ \sin(b_{\theta_s}) & \cos(b_{\theta_s}) \end{bmatrix} \mathbf{l}_{i_k} + \begin{bmatrix} b_{x_s} \\ b_{y_s} \end{bmatrix} \tag{2}$$

We chose to define and integrate the bias in the world frame $\mathfrak{R}_w$ in an additive manner but it is still necessary to take into account the rotation induced by the angular drift. Integrating this rotation in $\mathfrak{R}_w$ has the advantage to allow to completely correct a trajectory whose angle is biased. The bias can be seen as a frame change allowing to pass from a localization in a biased world to a pose taking into account the drift.

The typical evolution of the drift during trajectories suggests that a dynamic model is required. Autoregressive models are a great fit as they link each new estimate to the chronologically previous one. Indeed, the evolution of the drift clearly illustrates that each new estimate is tightly linked to the previous one. However, indexing its evolution according to the elapsed time is wrong as it would mean that a stationary vehicle could drift. The evolution equation of a continuous stochastic linear system [62] is:

$$\dot{\mathbf{b}}(s) = \mathbf{A}_c \mathbf{b}(s) + \mathbf{B}_c \mathbf{u}(s) + \mathbf{M}_c \boldsymbol{\varepsilon}(s) \tag{3}$$

where $\mathbf{b}$ is the state vector, $\mathbf{A}_c$ the state matrix (linking $\mathbf{b}$ temporally), $\mathbf{u}$ the input vector, $\mathbf{B}_c$ the matrix linking $\mathbf{u}$ to $\mathbf{b}$, $\boldsymbol{\varepsilon}$ the evolution noise (assumed white) and $\mathbf{M}_c$ the matrix linking the evolution noise to the state. s is the curvilinear abscissa.

In the case of the bias evolution, we consider an approximative model based on a simple random walk. Indeed, the impact that can have the rotation matrix on the vehicle position is integrated in Equation (1) and is consequently not directly involved in the drift evolution. As a consequence, for the drift evolution, $\mathbf{A}_c$ and $\mathbf{B}_c$ are equal to zero:

$$\dot{\mathbf{b}}(s) = \boldsymbol{\varepsilon}(s) \tag{4}$$

By discretizing the system, we obtain:

$$\begin{aligned} \mathbf{b}_s &= \mathbf{A}\mathbf{b}_{s-\Delta s} + \boldsymbol{\varepsilon}_{\Delta s} \\ &= \mathbf{b}_{s-\Delta s} + \boldsymbol{\varepsilon}_{\Delta s} \end{aligned} \tag{5}$$

where $\mathbf{b}_{s-\Delta s}$ is a previous bias estimate and $\boldsymbol{\varepsilon}_{\Delta s}$ is a white noise representing the drift occurring between $s - \Delta s$ and s (Δs being the distance traveled by the vehicle between $k-1$ and k). The state matrix $\mathbf{A}$ of the discrete model is defined as follows:

$$\begin{aligned} \mathbf{A} &= \exp(\mathbf{A}_c \Delta s) \\ &= 1 \end{aligned} \tag{6}$$

As $\boldsymbol{\varepsilon}_{\Delta s}$ cannot be directly estimated, it is necessary to characterize its variance $\mathbf{P}_{\varepsilon_{\Delta s}}$. The variance of $\boldsymbol{\varepsilon}_{\Delta s}$ is given by:

$$\mathbf{P}_{\varepsilon_{\Delta s}} = \int_0^{\Delta s} \exp(\mathbf{A}_c s) \mathbf{P}_{\boldsymbol{\varepsilon}} \exp(\mathbf{A}_c^T s) ds \tag{7}$$

where $\mathbf{P}_{\boldsymbol{\varepsilon}}$ is the covariance matrix of the white noise $\boldsymbol{\varepsilon}(s)$:

Thanks to Equations (5) and (7), we can infer the current bias uncertainty:

$$\mathbf{P}_{\mathbf{b}_s} = \mathbf{P}_{\mathbf{b}_{s-\Delta s}} + \Delta s \mathbf{P}_{\boldsymbol{\varepsilon}} \tag{8}$$

Even though this model is sufficiently accurate for SLAM algorithms, it is not perfect. Indeed, we consider here that the evolution of the model is independent of the physical state of the vehicle, meaning that the bias will evolve similarly if the vehicle is turning or just moving forward. However, by properly characterizing $\mathbf{P}_{\boldsymbol{\varepsilon}}$, this model is sufficient to guarantee the consistency of the computed localizations. This stationarity hypothesis allows to empirically quantify the evolution of the bias uncertainty:

$$\mathbf{P}_{\boldsymbol{\varepsilon}} = \frac{\mathbf{D}^2}{s} \tag{9}$$

where $\mathbf{D}^2$ represents the quadratic divergence observed over a distance s (s should be large enough to be relevant). $\mathbf{P}_{\boldsymbol{\varepsilon}}$ can thus be easily characterized for each vehicle and the environment in which they are evolving.

We will now show the impact that can have the angular bias integration on the vehicle position. To do so, we consider a simple example where a vehicle moves forward on the $\vec{x}$ axis. While moving, we will consider that the vehicle is only affected by an angular drift so as to identify how b_{Θ_s} affects b_{x_s} or b_{y_s}. However, as b_{x_s} and b_{y_s} are purely additive on x_k and y_k, adding a drift on these parameters does not change the evolution of b_{Θ_s}. After s meters and based on Equation (1), the unbiased

estimation of y_k (the divergence on x_k being meaningless), y_{u_k} , can be expressed as follows:

$$y_{u_k} = s \sin(b_{\Theta_s}) \tag{10}$$

This drift can be considered small and Equation (10) can be simplified:

$$\begin{aligned} y_{u_k} &\approx s b_{\Theta_s} \\ &\approx s \sum_{i=0}^{s} \Delta b_{\Theta_i} \end{aligned} \tag{11}$$

The associated variance is:

$$\sigma^2_{y_{u_k}} = s^2 var[\sum_{i=0}^{s} \Delta b_{\Theta_i}] \tag{12}$$

As the drift evolution noise is assumed white, the sum variance is the variance sum:

$$\sigma^2_{y_{u_k}} = s^3 \sigma^2_{b_\Theta} \tag{13}$$

We can notice that the angular bias has a considerable impact on the position drift.

The model, defined thanks to Equation (5), shows that each bias estimate is linked to the previous one. Let consider the link $cov(\mathbf{b}_{s-\Delta s}, \mathbf{b}_s)$ between the uncertainties which are associated to the biases $\mathbf{b}_{s-\Delta s}$ and $\mathbf{b}_s$. This relationship can be expressed as:

$$\begin{aligned} cov(\mathbf{b}_{s-\Delta s}, \mathbf{b}_s) &= E[\mathbf{b}_{s-\Delta s}\mathbf{b}_s] - E[\mathbf{b}_{s-\Delta s}]E[\mathbf{b}_s] \\ &= E[\mathbf{b}^2_{s-\Delta s} + \mathbf{b}_{s-\Delta s}\boldsymbol{\varepsilon}] - E[\mathbf{b}_{s-\Delta s}]E[\mathbf{b}_{s-\Delta s} + \boldsymbol{\varepsilon}] \\ &= E[\mathbf{b}^2_{s-\Delta s}] + E[\mathbf{b}_{s-\Delta s}\boldsymbol{\varepsilon}] \\ &\quad -E[\mathbf{b}_{s-\Delta s}](E[\mathbf{b}_{s-\Delta s}] + E[\boldsymbol{\varepsilon}]) \end{aligned} \tag{14}$$

The noise $\boldsymbol{\varepsilon}$ is assumed white and so $E[\boldsymbol{\varepsilon}] = 0$. As this noise is independent of $\mathbf{b}_{s-\Delta s}$, $E[\mathbf{b}_{s-\Delta s}\boldsymbol{\varepsilon}] = 0$ and so:

$$\begin{aligned} cov(\mathbf{b}_{s-\Delta s}, \mathbf{b}_s) &= E[\mathbf{b}^2_{s-\Delta s}] - E[\mathbf{b}_{s-\Delta s}]^2 \\ &= \mathbf{P}_{\mathbf{b}_{s-\Delta s}} \end{aligned} \tag{15}$$

Bias uncertainties are thus always linked by the previous variance. The major advantage of this is that if the value of the bias can be computed at one curvilinear abscissa, the information can be easily spread to all the different bias estimates and still be consistent.

3.2. Bias integration

The integration of the bias inside a SLAM algorithm must be separated from the classic SLAM process for several reasons. In order to build a generic framework for existing feature-based SLAM algorithms, this separation is essential as each algorithm would require a special bias integration. Moreover, directly integrating the bias is not trivial. Indeed, integrating the bias would mix the uncertainty of the landmark with the one of the bias. With new observations, the uncertainty (including the bias part) would be lowered thus causing inconsistency.

The separation naturally makes the inconsistent "low-level" SLAM algorithm independent of the rest. The organization of the whole system is given in Figure 2.

We use an EKF to handle the bias and the integration of information that can help to estimate the divergence (loop closing, etc.). In this paper, the low-level is based on a monocular EKF solution.

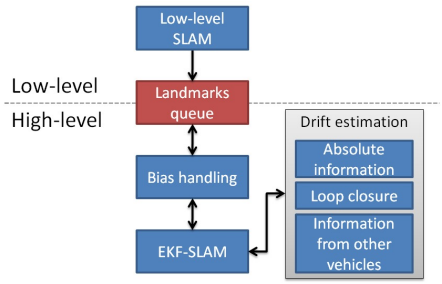


Figure 2. Organization of the whole system

Linking a bias estimate to the landmarks concerned or to the previous bias is done thanks to the high-level EKF. The first step is to connect the different bias estimates together. Indeed, as explained before, the value of the bias at one moment is different from the one coming just after. It means that in order to fully integrate the bias into the EKF, it would be necessary to insert a new estimate each time a new landmark, or vehicle pose, is added. The main problem is that the size of the state vector would quickly grow and become intractable. To avoid this issue, we chose to insert a new bias estimate periodically based on the distance traveled (every 5 meters in our case).

As the bias cannot be directly estimated, its initialization in the Kalman Filter is straightforward ($\boldsymbol{\varepsilon}$ equals 0):

$$\mathbf{b}_s = \mathbf{b}_{s-\Delta s} \tag{16}$$

The lack of knowledge regarding the value of $\boldsymbol{\varepsilon}$ is integrated into its covariance. We first initialize this covariance with infinite values. The idea is to use the EKF to refine it. We define an observation function h_b:

$$h_b(\mathbf{b}_{s-\Delta s}, \mathbf{b}_s) = \mathbf{b}_s - \mathbf{b}_{s-\Delta s} \tag{17}$$

It produces an observation equals to zero, which gives a null innovation in the Kalman equations. It means that only the covariance matrix of $\mathbf{b}_s$ ($\mathbf{P}_{\mathbf{b}_s}$) along with its link with $\mathbf{P}_{\mathbf{b}_{s-\Delta s}}$ are affected. We set $\mathbf{P}_{\mathbf{b}_o} = \mathbf{P}_{\mathbf{b}_{s-\Delta s}} + \Delta s \mathbf{P}_{\boldsymbol{\varepsilon}}$ and define the observation error as follows:

$$\mathbf{R}_b = \mathbf{P}_{\mathbf{b}o} - \mathbf{P}_{\mathbf{b}s-\Delta s} \tag{18}$$

We then compute the Jacobian matrix $\mathbf{H}_b$ associated to the observation function h_b to use it in the Kalman update equations.

$$\mathbf{K} = \mathbf{P}\mathbf{H}_b^T \left(\mathbf{H}_b \mathbf{P} \mathbf{H}_b^T + \mathbf{R}_b\right)^{-1} \tag{19}$$

$$\mathbf{P} = \mathbf{P} - \mathbf{K}\mathbf{H}_b\mathbf{P} \tag{20}$$

with $\mathbf{P}$ being the covariance matrix corresponding to the global state vector of the high-level EKF.

This update state will not only link together two consecutive bias estimates but also all the previous bias estimates (see Eq. (15)). With an example where a state vector is composed of four bias estimates, the covariance matrix will be defined as follows:

$$\begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_0 & \mathbf{P}_0 & \mathbf{P}_0 \\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_1 & \mathbf{P}_1 \\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_2 \\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 \end{bmatrix} \tag{21}$$

Integrating landmarks (and vehicle poses) in a similar manner will link them with their corresponding bias estimate (and with the rest of the state vector). The procdure is similar. The landmark is first inserted in the state vector thanks to Equation (2). $\mathbf{P}$ is then augmented with an infinite variance for the landmark. An update step of the filter is required to refine it. We consider an observation $\mathbf{z}_l = \mathbf{l}_i$ with its covariance $\mathbf{R}_l = \mathbf{P}_{\mathbf{l}_i}$, $\mathbf{l}_i$ being the landmark coming from the low-level SLAM and $\mathbf{P}_{\mathbf{l}_i}$ its uncertainty. We define h_l, the non-linear observation function allowing to pass from a landmark taking into account the bias to one ignoring it (the opposite of Equation (2)).

$$h_l(\mathbf{b}_i, \mathbf{l}_{u_i}) = \begin{bmatrix} \cos(b_{\theta_i}) & \sin(b_{\theta_i}) \\ -\sin(b_{\theta_i}) & \cos(b_{\theta_i}) \end{bmatrix} \left(\mathbf{l}_{u_i} - \begin{bmatrix} b_{x_i} \\ b_{y_i} \end{bmatrix} \right) \tag{22}$$

with $\mathbf{b}_i$ being the bias estimate associated to the landmark $\mathbf{l}_{u_i}$.

The Jacobian $\mathbf{H}_l$ associated to h_l is then computed and used in the update similarly to Equations (19) and (20).

With this bias integration, the system is aware of the drift. Finding loop closures becomes easier by taking into account the drift as it gives a strong hint at when to look for them. Be they loop closures, geo-referenced points or a common landmark between two vehicles, they all can be fused in the high-level SLAM. It is a simple 3D point-3D point fusion where one of the landmark is already in the state vector and connected to its bias estimate. The fusion will impact the whole state vector. Let consider a landmark $\mathbf{l}_{u_i}$ already in the state vector. A landmark $\mathbf{l}_j$, just received (from the low-level or a distant vehicle), is associated to $\mathbf{l}_{u_i}$ (the data association process is described later in Subsection 4.3). Let $\mathbf{b}_j$ be the bias estimate in the state vector corresponding to $\mathbf{l}_j$. We consider the new landmark as an observation $\mathbf{z}_f = \mathbf{l}_j$ with its covariance $\mathbf{R}_f = \mathbf{P}_{\mathbf{l}_j}$. It allows to define the following simple observation function h_f:

$$h_f(\mathbf{l}_{u_i}, \mathbf{b}_j) = \begin{bmatrix} \cos(b_{\theta_j}) & \sin(b_{\theta_j}) \\ -\sin(b_{\theta_j}) & \cos(b_{\theta_j}) \end{bmatrix} \left(\mathbf{l}_{u_i} - \begin{bmatrix} b_{x_j} \\ b_{y_j} \end{bmatrix} \right) \tag{23}$$

The innovation of the Kalman update can be computed:

$$\mathbf{\Delta} = \mathbf{z}_f - h_f(\mathbf{l}_{u_i}) \tag{24}$$

The Kalman gain can be calculated similarly to Equation (19) by replacing $\mathbf{R}_b$ by $\mathbf{R}_f$ and $\mathbf{H}_b$ with $\mathbf{H}_f$ which is the Jacobian associated to h_f. $\mathbf{H}_f$ is equal to zero everywhere except for the landmark $\mathbf{l}_{u_i}$ and the bias $\mathbf{b}_j$. The update will naturally impact the entire state vector and will help to estimate the drift affecting the vehicle (and landmarks) on the whole trajectory.

3.3. Simulation results

3.3.1. 1D examples

Let us consider a vehicle evolving along a line at one unit per iteration. This speed is noised so as to simulate a localization bias. The estimation of the bias and the vehicle pose are integrated inside a Kalman filter. The bias cannot be directly estimated and its covariance evolves with relation to the maximum drift given. Drift values are randomly generated. Figure 3 shows the vehicle evolution on one dimension.

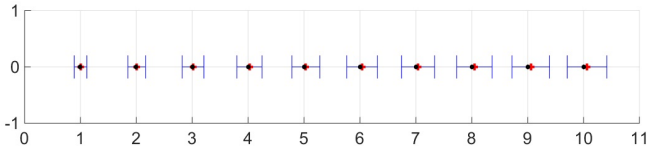


Figure 3. 1D localization example of a vehicle with bias integration. Red crosses represent vehicle pose estimates and black dots, real poses. In blue, the uncertainty computed from the bias integration.

We can notice that the vehicle pose slowly drifts but that the uncertainty still covers the real pose. Consistency is ensured in this simulation.

Let now consider that after 5 units, a landmark observed at the first unit is re-observed. The observation is fused. Figure 4 shows its impact on the bias. The values displayed here are the ones computed at each step (the feedback on unit 1 to 4 is therefore not visible).

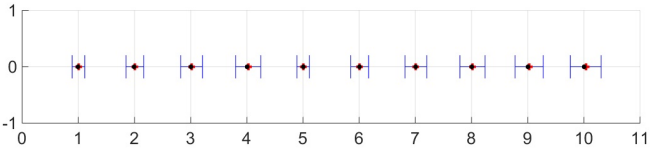


Figure 4. 1D localization example of a vehicle with bias correction. Same color code as the previous figure.

The vehicle position is corrected at the fifth unit and then diverges progressively as no new information on the drift is available. The landmark fusion allowed to decrease the bias uncertainty to what it was at unit 1 where the landmark was first seen. The uncertainty bias profile with and without the fusion is available in Figure 5 in red.

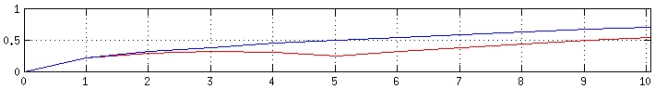


Figure 5. Evolution and correction of the bias uncertainty on a 1D example. In blue, the uncertainty without any fusion. In red, the uncertainty with the landmark fused at unit 5.

As soon as the landmark is fused, the uncertainty is lowered. At this moment (unit 5), the vehicle position at unit 3 is the farthest from the fusion (which closes the loop between unit 1 and 5) and so the one the less affected by it.

3.3.2. 2.5D simulations

The simulations presented here will expose the impact of the angular bias on the computed pose and its integrity. We first consider a trajectory of 100 meters where a vehicle can drift of 0.001 radian per meter. So as to illustrate the weight of the angular drift, only a very small position drift is added here. Figure 6 shows the trajectory performed while considering a random realization of the drift.

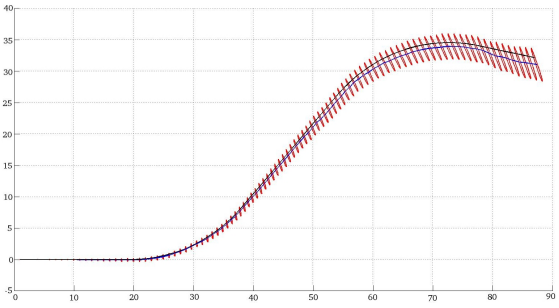


Figure 6. Simulation trajectory with an angular drift. In blue, the trajectory with the drift. In black, the ground truth. The red ellipses are the uncertainties computed at 3σ that integrate the angular drift.

We can notice that despite the angular drift, consistency is preserved all along the trajectory thanks to the bias evolution and integration.

Let now show that, whatever the realization of the angular bias, the position uncertainty integrating this drift does include the different possible trajectories. To illustrate this point, we ran another simulation with the same context as previously. In this simulation, 1000 realizations of an angular bias have been generated based on a maximal drift of 0.001 radian per meter. The results can be seen in Figure 7.

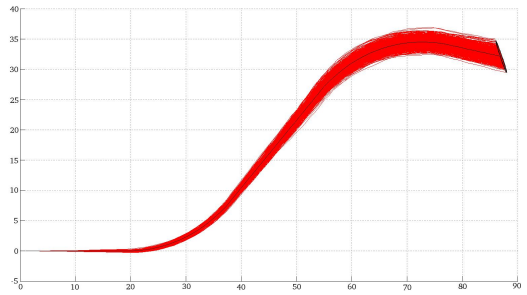
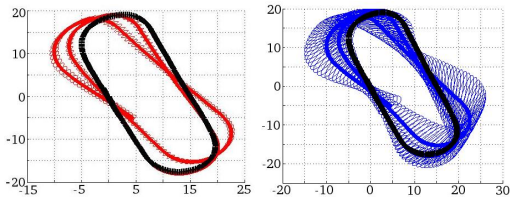


Figure 7. Different trajectory realizations with an angular drift (in red). The black ellipse is the uncertainty at 3σ computed based on the defined bias evolution.

We can see that these 1000 realizations are almost all included in the position uncertainty computed at 3σ. It shows that the defined drift model is coherent and able to properly model an angular divergence.

In the next simulation, we present a loop closing situation (two turns on a ring). Odometric data has been generated and the measurements have been noised in order to simulate the behavior of a real vehicle. Simulated landmarks have also been created. A set of 8 points have been defined at the beginning of the trajectory which are then used to perform loop closures and estimate the bias. The data association is known. Figure 8(a) shows the results provided by a classic EKF not taking into account the drift or performing data associations. Figure 8(b) shows the same trajectory but with the bias integration (still no associations).

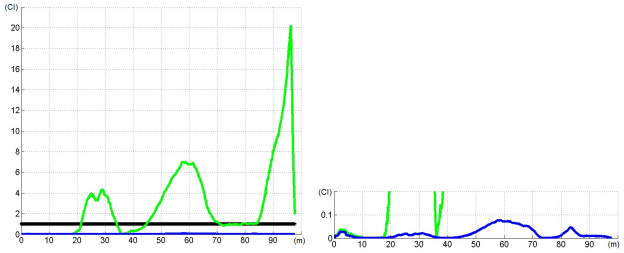
We can see a strong angular drift that is growing along with



(a) Trajectory without taking into account the bias (b) Trajectory taking into account the bias

Figure 8. Trajectories performed without closing the loops. In black is the ground truth. The colored curves and ellipses are the computed vehicle estimates.

the distance traveled. Without taking into account the bias, the computed vehicle positions are not consistent with relation to the ground truth whereas it is the case as soon as the bias model is active. In order to illustrate this aspect, we computed the Consistency Index (as defined in [63]) for both localization results. The CI is based on the Normalized Estimation Error Squared (NEES) and the Chi-square test. A CI lower than 1 means that the estimate is consistent with the ground truth. Estimates whose CI is greater than 1 are optimistic (inconsistent). Figure 9 shows the CI for this trajectory.

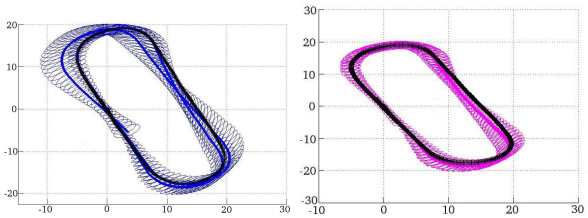


(a) Consistency Index without closing the loop (b) Zoomed view

Figure 9. Consistency Index without closing the loop. In blue is the CI with bias integration. In green is the CI without bias estimation. The black line represents the CI below which consistency is ensured.

Figure 10(a) shows the whole trajectory when the loop is closed the first time the vehicle returns to the starting point. The first lap is properly corrected and the uncertainty is still coherent. During the second turn, the vehicle starts to drift similarly to the first turn but the bias evolution is able to cover it. Figure 10(b) now illustrates the trajectory when the two loop closures are performed.

Consistency is still ensured (see Figure 11) and the bias estimation is able to offer a better localization (see the Root-Mean-Square Error in Figure 12). The example of the loop closure could easily be replaced by GPS information or maps from other vehicles. The next section will now present how this algorithm has been extended to a fleet localization.



(a) Trajectory when closing the loop the first time (b) Trajectory when closing the loop at each lap

Figure 10. Trajectories performed when closing the loops

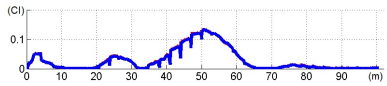


Figure 11. Consistency Index when closing the loop with bias integration.

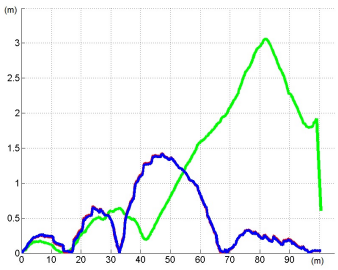


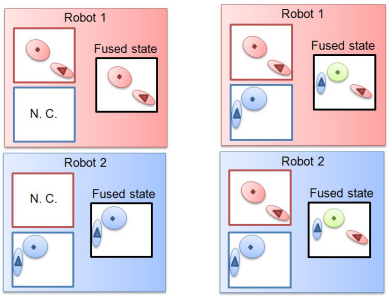
Figure 12. Root-Mean-Square Error with bias integration (in blue) and without bias integration (in green).

4. Multi-vehicle SLAM

4.1. Multi-vehicle architecture

Concerning data incest (double counting information), the strategy often used consists in never exchanging already fused information so as to avoid considering a piece of data twice. In [24], the vehicle states are separated in what are called substates. The idea is to freely exchange the independent substates without fearing overconfidence. When a substate is received, it just replaces the old one. The substates can be regularly fused in order to obtain a global state that takes advantage of all the available information. Only the substates are shared as they are independent. With all the substates shared, each vehicle is capable of computing the same global state as the other.

Though designed for multi-vehicle localization, this method can be adapted to fit our needs. The biggest change is that not only vehicle localizations must be sent but also maps. The maps need to be separated in submaps in order to be independent and avoid data incest. A 2-vehicle example is exposed in Figure 13. In this example, without communication, the global map obtained by each vehicle corresponds to its own submap. Once the submaps are shared, each vehicle can fuse them in order to obtain a global map where the different data (landmarks and vehicles) are expressed in a common frame. This way, landmarks common between submaps can help correct the drift and estimate the relative distance and orientation between vehicles.



(a) Before communication (b) After communication

Figure 13. Separation in submaps so as to avoid data incest. In both figures, the squares at the left are the submaps (one for each vehicle) and the square at the right is the global map fused from the submaps. In red, information from Robot 1 and in blue, information from Robot 2. In green, information found common (and fused) between both robots.

We define a substate $\mathbf{X}_{r_k}$ of a vehicle r at a moment k as follows:

$$\mathbf{X}_{r_k} = \begin{pmatrix} \mathbf{v}_{r_k} & \mathbf{M}_{r_k} & \mathbf{B}_{r_k} \end{pmatrix}^T \tag{25}$$

where $\mathbf{v}_{r_k}$ is the vehicle state as defined previously, $\mathbf{M}_{r_k}$ is the map of the n accurate landmarks at the moment k, defined as $\mathbf{M}_{r_k} = (\mathbf{l}_{r_0} \dots \mathbf{l}_{r_n})^T$, and $\mathbf{B}_{r_k}$ the m bias estimates concerning the vehicle r at the moment k, $\mathbf{B}_{r_k} = (\mathbf{b}_{r_0} \dots \mathbf{b}_{r_m})^T$.

The fused state, or decentralized map, represents the integration of the drift for each substate and the fusion of landmarks common between vehicles when there are any. It is defined as:

$$\mathbf{X}_{d_k} = \begin{pmatrix} \mathbf{V}_{u_k} & \mathbf{M}_{u_k} & \mathbf{B}_k \end{pmatrix}^T \tag{26}$$

with $\mathbf{V}_{u_k} = (\mathbf{v}_{u_{0_k}} \dots \mathbf{v}_{u_{p-1_k}})^T$ (p being the number of vehicles) computed individually with Equation (1) and $\mathbf{M}_{u_{r_k}}$, the landmarks computed thanks to Equation (22) or, when already in the state vector, fused with Equation (23). $\mathbf{B}_k$ represents the bias estimates from the different vehicles when they are connected with Equation (17).

Maps grow and improve over time in a SLAM algorithm, conversely to the localization approach described in [24], thus preventing the replacement of a whole substate (and of the decentralized map) each time it changes because of the computational burden involved. It means that sending a landmark to all the vehicles every time it is updated in the low-level is not a viable solution. Indeed, it would generate a large network traffic and above all, it would force to break the decentralized map in order to rebuild it with the new landmark estimate. Because of this constraint, we chose to only use landmarks from the low-level in the decentralized part of the algorithm (in substates and in the fused state) when they can be considered accurate (cumulated standard deviation on the 3 axes below a threshold). They also remain in the low-level algorithm while they help the vehicle localization. However, the updates will not be brought to the high-level for the same reasons as mentioned above. This

loss is negligible since only accurate landmarks are transferred. This constraint is mainly valid when sensors do not provide accurate depth immediately (as it is the case with a monocular SLAM solution). With other sensors (laser and stereovision for instance), landmarks are accurate when initialized, allowing to use them right away in the decentralized part of the algorithm.

Anyway, thanks to this choice, we are able to avoid having to break the global map. Indeed, only new landmarks (coming from the low-level or the network) will be added, thus constantly improving the overall map. It reinforces the separation established in Subsection 3.2. It also means that the SLAM algorithm used in the low-level can use any sensor as long as it provides landmark estimates with their uncertainties along with the vehicle pose. The architecture, designed for a multi-SLAM use, is depicted in Figure 14 with the same 2-vehicle example as before.

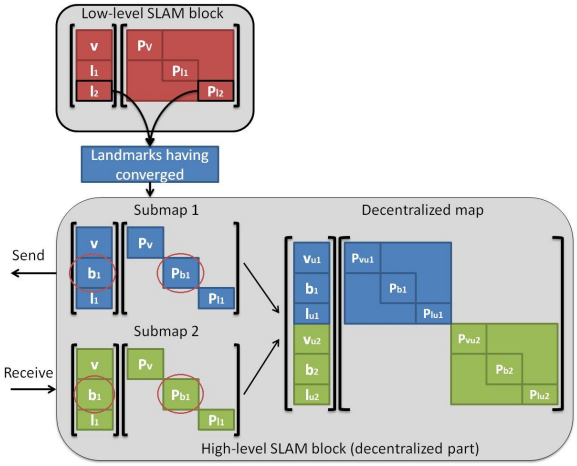


Figure 14. Multi-vehicle architecture (2-vehicle example)

Figure 14 maintains the structure already presented in Figure 2. The top gray block represents the classic SLAM. In this example, the vehicle pose and 2 landmark estimates are shown. One of them has converged (squared in black) and is thus transmitted to the decentralized process along with the vehicle estimate. It allows the creation of the submap 1 (blue submap). A new bias, initialized to represent the drift affecting the landmark and the vehicle, is inserted into the submap (circled in red in the first submap). Submap 1 can then be sent to the other vehicles of the team. In this example, another vehicle has communicated its submap (in green here). The vehicle pose has been received along with a landmark and the associated bias estimate (circled in red in the submap 2). By fusing these two submaps inside an EKF, the global decentralized map is obtained (right part of the bottom gray block) where everything is expressed in a frame common to both vehicles.

One can notice that cross-covariances are not copied from the low-level algorithm and not sent (or received) to (from) other vehicles. Indeed, as only accurate landmarks are kept, cross-covariances are very weak, making them negligible compared to the drift affecting all the landmarks. It allows to lighten the communications even more. However, landmarks are still linked by the SLAM drift which affects every landmark. The integration of the bias, thanks to Equations (1) and (2), will naturally retrieve the cross-covariances between landmarks and vehicle poses. In the decentralized map of the example given in Figure 14, it can be seen that the bias estimate affecting each vehicle is linked to the concerned landmark and vehicle. By finding a common landmark between submaps 1 and 2, the decentralized map will become entirely connected (see Subsection 3.2).

Real life applications imply to be robust against communication failures. It can be split into several problems: latencies, desynchronizations and communication losses. Concerning desynchronizations, they are mostly due to the fact that each sensor delivers information at a different speed. In a decentralized scheme, it is easy to understand that the number of vehicles involved could potentially lead to unmanageable situations. In the architecture proposed here, it is not the case as the low-level only provides high-level information (landmarks instead of raw measurements). Desynchronizations are thus easy to handle as they can be avoided by sharing a time line between the different vehicles thanks to the Network Time Protocol (NTP) for instance. With every piece of data timestamped, it becomes possible, with a constant velocity model, to have an estimate of each vehicle pose whenever it is needed.

Concerning latencies, the exchange strategy prevent them from having any effect. Indeed, landmarks pass from the low-level to their hihg-level corresponding submap once. It is a copy from the low-level to the high-level and so no network communications are involved. As a consequence, only one version of each landmark will exist and be sent to other vehicles. It means that it does not matter when it is received because this landmark will never be updated in the submap but only in the decentralized map which is not exchanged. It is also interesting because any vehicle can share all the submaps available even if it did not build them. Vehicles can thus act as relays to transfer information, increasing the communication range and adding redundancy. If a vehicle receives the same landmark twice, it is not considered the second time (same landmark that is received so there is no information loss). The system must thus be able to identify landmarks. As a consequence, each landmark is identified with a unique index (vehicle number and landmark number) defined locally.

The unique index per landmark has also the advantage to provide a solution to data loss. Each time a new landmark is copied onto the high-level SLAM, the index is incremented. For a distant vehicle, identifying missing landmarks becomes easy as there will be a gap between the indices (for example landmarks $\mathbf{l}_1$ and $\mathbf{l}_3$ coming from vehicle 1 are received by vehicle 2, $\mathbf{l}_2$ is missing). A request can then be sent to the concerned vehicle which will respond with the missing landmarks. This mechanism lightens the communications as it avoids sending the substates all the time but only when necessary.

4.2. *Static and dynamic biases*

The dynamic bias model presented in Subsection 3.1 totally suits the decentralized needs as each vehicle drifts independently of the others. However, it is still necessary to express

the whole state (vehicle, landmarks and biases) of the fleet in a common frame. Indeed, in the low-level (and in the corresponding submap), everything is expressed with relation to a local frame initialized at the starting point of the vehicle. The easiest way to get around this problem is for each vehicle to express the whole team in its own reference frame.

Nevertheless, doing so requires to know the distance and orientation between each vehicle. In multi-vehicle scenarios, considering this as a prerequisite drastically limits the potential applications. Instead of forcing the system to start with known positions, we decided to take advantage of this knowledge only if available [64]. To do so, we integrate it under the form of a static bias which is actually the initial value of the first bias estimate. If the distance and orientation between a vehicle and a frame, common to the whole team, is approximately known, it can be used to initialize the first bias estimate inserted into the decentralized part of the SLAM. Approximations can also be taken into account in the covariance matrix associated to the bias estimate. Let us consider 2 vehicles evolving in a 200-square-meter zone for instance. If we define the reference frame as the center of this area, each bias covariance could be initialized with a 15-meter-radius uncertainty to be sure to include the common frame. A similar method can be followed for relative orientations. An example is visible in Figure 15.

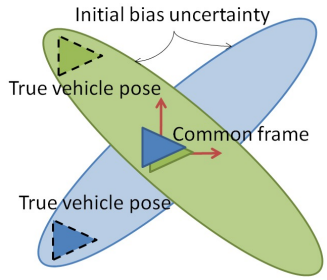


Figure 15. Example of static bias initializations for two vehicles. The plain triangles are the initial estimated positions in the common frame. The ellipses represent the static bias uncertainties. The dashed triangles are the true poses.

The main advantage of integrating a static part to the bias is to represent any kind of knowledge available before the beginning of a trajectory, be it an actual value or an uncertainty. Another essential aspect is the preservation of consistency. Indeed, even with no knowledge about the relative poses between vehicles, covariance matrices can still be initialized with infinite values. It also allows us to design a data association algorithm based on compatibility between landmarks. With consistency ensured, associations between landmarks coming from different vehicles can first be filtered depending on Mahalanobis distances.

4.3. Data association in a fleet context

The state-of-the-art analysis provided in Subsection 2.1 concerning data association in multi-vehicle SLAM has already given some insights about what existing algorithms could suit a general approach. Data association algorithms that rely heavily on identical spatial settings for landmarks are not ideal. Though the geometric organization of landmarks is crucial to recognize previously seen areas, it is important to bear in mind that only a few landmarks can be common between two maps of the same zone. Of course, it depends on the sensors, filtering method or feature selection algorithm used in the low-level.

The previously mentioned JCBB algorithm can be a good fit coupled with the static bias integration. However, if the prior knowledge about the team organization is weak, Joint Compatibility will not help. The uncertainty ellipses will be large and intersect each other frequently making it impossible to find a proper configuration of landmarks. The consistency ensured by the bias notion can still be used as it can discard some wrong matches and can greatly help once the initial bias uncertainty has been refined.

Subsection 2.1 exposed another algorithm called Geometric Constraints Branch and Bound (GCBB). A tree of the different association possibilities (between new landmarks and the decentralized map in our case) is built and explored. Some constraints are defined in order to find the subset of observations that best matches a map. Even though it is possible to define new constraints, GCBB is mostly based around a binary geometric constraint. In order to be added to the current hypothesis, an observation and its associated landmark must maintain a coherent spatial organization with the already selected observations (hypothesis) and the landmarks with which they have been associated. A hypothesis is considered correct if the number of landmarks composing it is above a threshold (usually set to 5 or 6 common landmarks to avoid wrong associations). Figure 16 explains, with an example, how this process works with two steps of the algorithm.

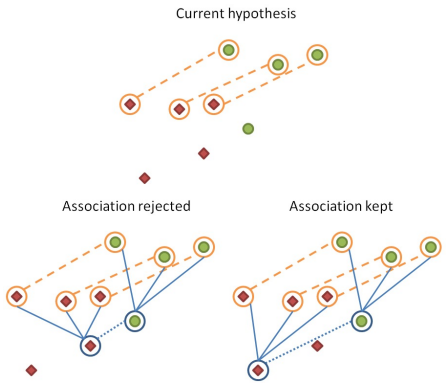


Figure 16. Data association example. In orange, the current association hypothesis. Red diamonds are landmarks from the decentralized map. Green circles are observations (new landmarks). Circled in blue are the points whose association is currently being tested. To be confirmed, the distances to the points in the hypothesis must be similar between new landmarks and the global map (plain blue lines).

When this data association algorithm was first presented in [41], the authors stated that it can be computationally costly because of the tree search. Indeed, with a high number of landmarks, the amount of solutions to test quickly increases. The bias integration allows us to add a filtering method in order to prune unlikely branches of the tree beforehand and thus reduce

the exploration time. We can limit the number of landmarks that can be associated to those which are individually compatible in terms of uncertainties. The Individual Compatibility (IC) between two landmarks, $\mathbf{l}_{u_i}$ and $\mathbf{l}_{u_j}$, respectively associated to their covariance matrix $\mathbf{P}_{\mathbf{l}_{u_i}}$ and $\mathbf{P}_{\mathbf{l}_{u_j}}$, can be computed as follows:

$$D_{ij}^2 = \nu_{ij}^T \mathbf{C}_{ij}^{-1} \nu_{ij} < \chi_{d,\alpha}^2 \tag{27}$$

with:

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

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

The acceptation threshold (below which an association is possible) is based on the Chi-squared distribution $\chi_{d,\alpha}^2$. Here, $d = 2$ (degrees of freedom) and we set the desired confidence, α to 0.95. Only individually compatible landmarks remain in the tree making its exploration faster.

This algorithm is used to detect loop closing and areas mapped by another member of the team. The data association is smartly triggered to avoid useless processing. Only new points are checked against the decentralized map. If no potential associations are found, the landmark is directly added to the global map and linked to the rest of the state vector. Otherwise, it is kept inside the data association process and if enough associations sharing the same spatial configuration are available, the fusion algorithm is triggered. As stated before, a vehicle passing through an already explored area will not map the same landmarks. Instead of analyzing the common landmarks at one moment, we chose to look for them in map portions. It means that landmarks with potential matches are kept in the association process for a certain time. If, during this period, enough landmarks are found to certify that a place has been recognized, a fusion will be performed. If it is not the case, the landmarks will be added to the global map once enough time has passed. The algorithm is presented in Figure 17.

procedure DATA ASSOCIATION(l_{new})
  % $\mathbf{l}_{new}$: new landmarks (from low-level or received)
  % n: number of new landmarks
  % $\mathbf{l}_{association}$: landmarks that can be associated with the
  % global map
  for $i = 0$ to $n - 1$ **do**
    % compute if compatible with the current decentral-
    % ized map with Eq. (27)
    if individuallyCompatible($\mathbf{l}_{new_i}$, $\mathbf{M}_{u_k}$) **then**
      $\mathbf{l}_{association} = [\mathbf{l}_{association}\ \mathbf{l}_{new_i}]$
    else
      % add landmark using Eq. (2) and (22)
      addToGlobalMap($\mathbf{l}_{new_i}$)
    end if
  end for
  % if enough landmarks to trigger the association
  if $\mathbf{l}_{association}$ > associationThreshold **then**
    % find associations with GCBB
    $\mathbf{l}_{associated}$ = GCBB($\mathbf{l}_{association}$)
    if $\mathbf{l}_{associated}$ > associationThreshold **then**
      % fuse associated landmarks with Eq. (23)
      fuseInGlobalMap($\mathbf{l}_{associated}$, $\mathbf{X}_{d_k}$)
    end if
  end if
  % remove old potential associations based on time
  $\mathbf{l}_{old}$ = removeOldAssociations($\mathbf{l}_{association}$)
  addToGlobalMap($\mathbf{l}_{old}$)
end procedure

Figure 17. Association algorithm

5. Results

Different experiments have been conducted to further validate our general framework and drift-aware approach.

In the experiments that follow, we consider the application of our method to electrical vehicles capable of transporting people in urban areas. The objective is to ease and fasten small trips around town centers or specific locations. The number of vehicles can vary depending on the size of the environment. In these experiments, we present 2 and 3-vehicle cases due to the available resources (it is also true for simulations as the computational power to generate data becomes too important with more vehicles).

Regarding network aspects, each vehicle is capable of communicating wirelessly with the 802.11b standard by sending UDP datagrams. The messages, containing localization information and landmarks, are broadcast to the whole fleet. No specific processing are applied to ensure the delivery or handle losses at the network level. Only the actions handled by the architecture and described previously are used.

5.1. Realistic simulations

The multi-vehicle context makes the deployment of any solution difficult. The cost and the technical support required to realize such operations are substantial. To avoid these drawbacks, the first experiment was carried out on a simulator.

The simulator used (Cobaye)[1] presents a realistic physics with vehicles responding similarly to real ones. The environment in which the simulation took place has been mapped from a real experimental platform (PAVIN) which will be later used for the real experiments. PAVIN recreates an urban setting with roads, sidewalks, roundabouts, crosswalks, traffic lights, buildings and so on. The simulator is only used to provide sensor data. The processing for each application (complete SLAM) is performed on computers totally separated from the data generation (one computer per vehicle). This way, network communications are still happening as it would with real experiments.

Concerning the simulated sensors equipped on the vehicles, it consists of a camera providing 800x600 black and white images and an odometer. The camera is running at 10 Hz and the vehicles are moving at 2 meters per second. In the trajectory performed, two vehicles, equipped with the same sensors, are involved. The algorithm is running in real time on both computers. Pictures illustrating the environment and typical camera

[1] Cobaye is developed by 4D-virtualiz: http://www.4d-virtualiz.com

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

Figure 18. Environment and camera outputs provided by the simulator

outputs can be found in Figure 18. As for all the other experiments of this article, the environment is not known beforehand.

It is also worth noting that, in all the experiments of this paper, we do not directly measure the distance separating fleet members even if it is possible. We only use common landmarks, shared between the vehicles, to estimate their relative distance and orientation. However, integrating a direct observation is possible and interesting as it would allow to estimate easily the static bias. This piece of information could be used by the high-level Kalman filter and fused with the vehicle pose.

This first trajectory (scenario 1) illustrates one of the advantage of a fully decentralized approach: there is no leading vehicle. The idea is to change the front vehicle during the trajectory. It is also a good way to show that our algorithm is able to deal with long distances between vehicles (here more than 10 meters). The trajectories are around 110-meter long each. An overview is depicted in Figure 19.

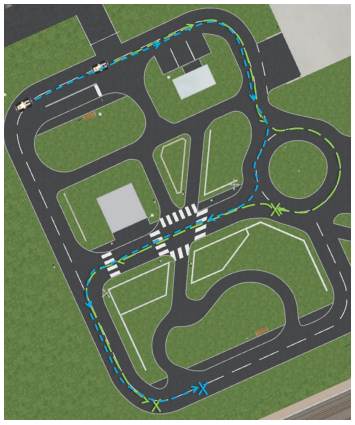


Figure 19. Scenario 1: Overview of the simulated trajectory. Crosses correspond to stops. The green vehicle stops in the roundabout while the blue one takes the convoy lead for the rest of the trajectory.

The two vehicles are separated by 11 meters and start in a column formation (green vehicle followed by the blue one). After the first bend, the front vehicle takes the roundabout with a longer path than the back vehicle. The vehicle which is leading stops while the other continues its path leaving several meters between them. The trajectory goes on until the final stop. The common portions of the map are not very long and require the data association to work efficiently. Concerning the bias initialization, the front vehicle starting point serves as a common frame. The back vehicle is thus inaccurate because its initial localization, $(0,0,0)^T$, is wrong in this common frame (its true initial localization is closer to $(-11,0,0)^T$). We chose to initialize its bias uncertainty with values generating a 20-meter-radius circle so as to not guide much the association process. To be as clear as possible, the results presented for this trajectory (and all the following) will only be those of one vehicle. Indeed, information shared in the team are the same, making the different decentralized maps identical. Here, the results are coming from the vehicle first leading the convoy (and so the trajectory of the other one is received thanks to network communications). The trajectories performed by the low-level SLAM are visible in Figure 20. Less than 400 landmarks have been mapped during this experiment (approximately 190 landmarks per vehicle). The low-level algorithm was tracking around 30 features per frame.

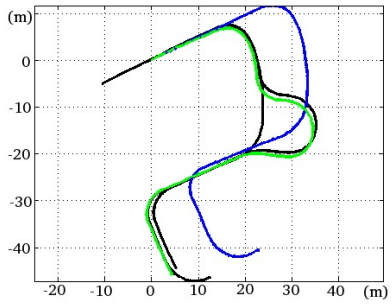
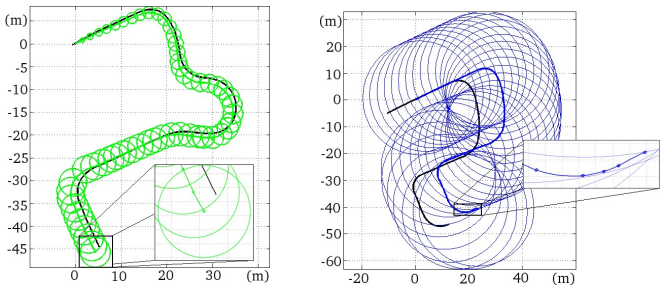


Figure 20. Scenario 1: SLAM trajectories as perceived by the front vehicle. In green, the front vehicle and in blue, the one starting at the back along with their small uncertainties. The ground truth is in black.

The blue vehicle trajectory is shifted by 11 meters because of the lack of knowledge about the distance to the common frame. The uncertainties of both vehicles remain very small during the whole trajectory, meaning that the integrity of the localization is not preserved. Figures 21(a) and 21(b) show how the integration of the drift affects the uncertainties. A closer look at the uncertainties computed by the low-level SLAM algorithm is provided.

The integration of the bias model allows to achieve consistent localization throughout the trajectory. In the case of Figure 21(a), the evolution of the bias uncertainty is visible as the vehicle starts perfectly localized. The closer look given to the end of the trajectory shows that the uncertainty coming from the low-level SLAM is far from including the path really fol-



(a) Bias integration for the front vehicle

(b) Bias integration for the back vehicle

Figure 21. Scenario 1: Bias integration with no associations performed

lowed by the vehicle (very small ellipses). The evolution of the uncertainty is less obvious in Figure 21(b). Indeed, the static bias uncertainty provided at the beginning of the trajectory is significant (it generates a 20-meter-radius circle), making the evolution of the dynamic part less visible. The zoom on the end of the trajectory shows, once again, the inconsistency of the classic SLAM.

The localization results for both trajectories with landmark associations are given in Figure 22. For clarity purposes landmarks are not visible, only the trajectories and uncertainties are.

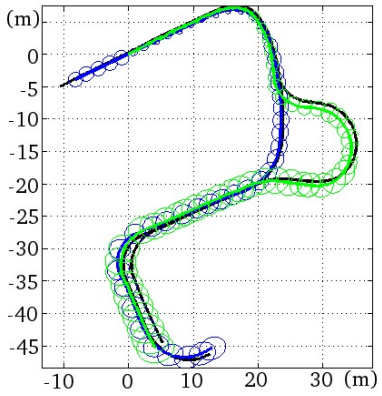


Figure 22. Scenario 1: Trajectories computed by the two vehicles when the data association is active

The localization obtained thanks to the bias integration is close to the ground truth. The bias uncertainties have been greatly reduced, especially for the blue vehicle which started with a large initial covariance. The vehicle localizations maintain the consistency even with the reduced uncertainties.

Common portions of maps have been found. Still, it is worth noting that not much landmarks are common to both vehicles. Indeed, with a slightly different orientation for the vehicle, the feature selection can be totally different. For instance, after the roundabout, the green vehicle tends to map landmarks from the right side of the road whereas the blue one has almost only mapped those on the left side.

The quality of the estimated distance separating the vehicles is exposed in Figure 23 and compared to the ground truth.

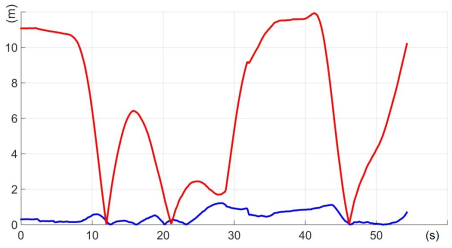
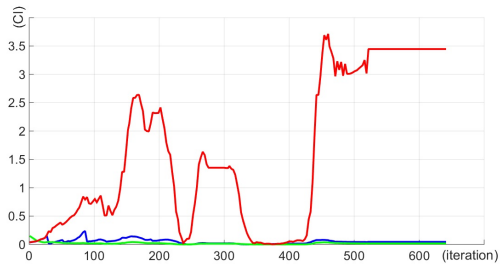
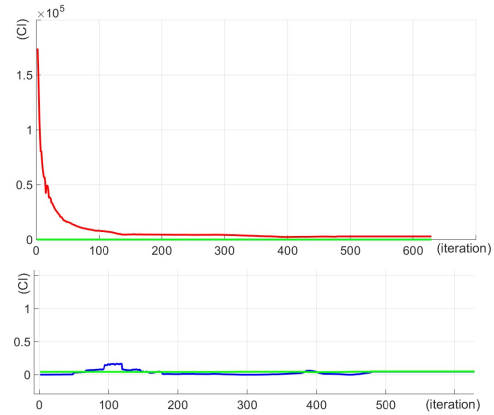


Figure 23. Scenario 1: Root-Mean-Square Error regarding the distance between the two vehicles according to the elapsed time. In blue, RMSE computed with the decentralized algorithm. In red, RMSE based on the SLAM estimates without considering drift.

As the distance between the vehicles changes a lot, it is a good indicator of how good the bias model is. Indeed, with new associations, all the previous vehicle estimates are corrected because linked to the bias evolution. This retro-action helps estimate the relative distance between the two vehicles (mean error below 30 centimeters). We can see in Figure 23 that it follows closely the real distance. The biggest gaps (around 1 meter) occur when no associations are found. An evaluation of the consistency is provided in Figure 24 based on the Consistency Index and shows that the bias integration allows for a consistent estimation even without information to reduce the drift.



(a) CI for the front vehicle



(b) CI for the back vehicle (with a zoomed view)

Figure 24. Scenario 1: Consistency Index for back and front vehicles. In red, CI when considering only the low-level information. In blue, CI with bias integration and no associations performed. In green, CI when the bias is integrated and associations are performed.

The CI is way more important for the back vehicle when the drift is not considered as the initial error is not covered. However, even for the front vehicle, which starts perfectly localized, consistency is not ensured (CI above 1) throughout the trajectory when the bias is not integrated.

Concerning the network communications, the data transfer rate is low. The bandwidth used by each vehicle is depicted in Figure 25. Less than 10KB per second are needed per vehicle which means the fleet size could be increased without fearing network congestions.

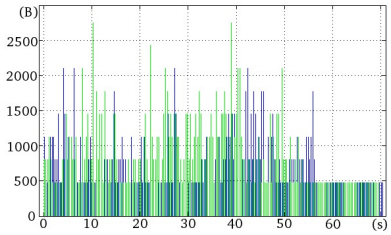


Figure 25. Scenario 1: Quantity of data sent depending on the elapsed time. The blue and green curves are respectively for the vehicles starting at the back and at the front of the convoy. The quantity of data is expressed in bytes.

We replayed the same trajectories but, this time, we cut the network during 10 seconds to test the algorithm robustness against communication failures. As expected, localization results are almost identical. With no communications, the algorithm continues to use only information coming from the low-level. As soon as the network is available, exchange are possible. Because of the gap between landmark indices, vehicles are able to notify that some landmarks are missing and request them. When received, those landmarks are integrated in the decentralized map and used to find associations. As trajectories are almost identical, we only present the impact of the communication interruption on the bandwidth used in Figure 26.

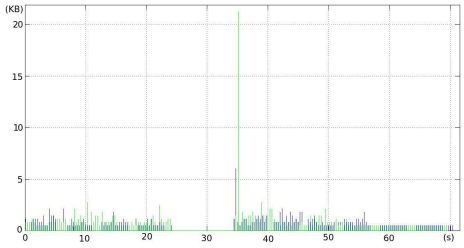


Figure 26. Scenario 1: Quantity of data sent depending on the elapsed time. Same color code as previously. The quantity of data is expressed in Kilobytes.

After the interruption, big spikes can be noticed. It is due to the answer of each vehicle to the request sent by the other. The blue vehicle has missed 65 landmarks that were mapped by the green vehicle during this period of time and that are thus sent all at once when requested. The green vehicle requested 15 missing landmarks to the blue vehicle. Even with this important interruption, the bandwidth needed is still far from technological constraints.

The results exposed in this subsection confirm that our decentralized approach is viable. In terms of processing power, it requires a standard laptop (Intel Core i5 running at 2.4GHz) to make the algorithm run in real time. Concerning the amount of memory used, it remains low throughout the trajectory (several hundreds of KB) as the number of landmarks mapped is quite small. The bandwidth needed is far from limiting as only 10KB per second per vehicle are enough (missing information can be spread through time to limit bandwidth needs). With technological constraints over several Megabytes per second, a large fleet of vehicles can be considered. The sensors utilized can be changed and a low-cost solution (one camera with an odometer) has been proven to work in simulation. In order to fully validate our approach, real experiments have also been carried out.

5.2. Real-life experiments

The context of the experiments on real data is very similar to the one exposed in Subsection 5.1. The vehicles are equipped with a single camera (Marlin F-1318 providing 1024x768 black and white images) and use odometric information to feed a simple motion model. The experiments are performed on the PAVIN platform (Figure 27(b)) with electrical vehicles called VIPALABs (Figure 27(a)). The camera of each vehicle is running at 15 frames per second and a RTK GPS is embedded to provide a ground truth. As for the simulated scenario, the application is executed on a laptop embedding an Intel i5 processor running at 2.4GHz. Each vehicle is using a single laptop in a fully decentralized way. Our algorithm is running in real time at 15Hz. All the vehicles are driven by operators. Communications between vehicles are established using the Wi-Fi 802.11b standard. Some pictures coming from the cameras of the experiments are provided in Figure 28 for a glimpse of the environment in which the experiments take place.

(a) Electrical vehicle VIPALAB

(b) Urban platform PAVIN

Figure 27. Experimental means

One advantage of our approach is that it does not require a column-like formation. Indeed, as the algorithm is fully decentralized, there is no leader in the fleet thus allowing to change the order of a convoy, as showed previously, or to have vehicles evolving side by side.

This configuration offers many possible applications, for instance in the agricultural domain where several vehicles can harvest a field side by side thus going faster than a single vehicle. To test this setting, two vehicles moving side by side on our experimental platform were used (scenario 2). Due to the

(a) (b) (c) (d)

Figure 28. Various camera outputs

size of the road, both vehicles were moving slowly, at approximately 1 meter per second to avoid any accident. Likewise, for safety reasons, only a long straight line of around 40 meters was accomplished. The two vehicles were separated by 2 to 3 meters depending on their variable speed. An overview of the trajectory is given in Figure 29.

Figure 29. Scenario 2: Overview of the trajectory. Circles correspond to starting points of the vehicles and crosses to stops.

As for the previous experiment, the starting point of the green vehicle is used as the common frame for both vehicles. Both vehicles wirelessly communicate their SLAM trajectories and maps. The trajectories performed by the SLAM algorithm without any decentralized processing or bias integration are exposed in Figure 30. 240 landmarks were mapped by both vehicles (around 120 for each).

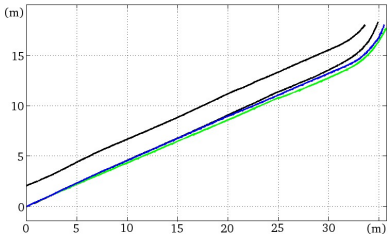
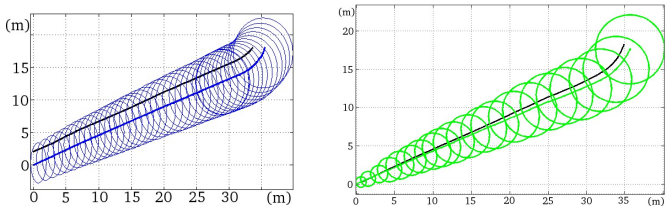


Figure 30. Scenario 2: SLAM trajectories as perceived by the right vehicle. In green, SLAM trajectory of the right vehicle and in blue, of the left one. Ground truth is in black for each vehicle.

The uncertainties displayed in Figure 30 are barely noticeable further demonstrating the inconsistency of SLAM algorithms. To cover the initial translation error of the blue vehicle, we initialized the bias with an uncertainty of 3 meters for the lateral error and 1 meter for the longitudinal error. Figures 31(a) and 31(b) show the SLAM trajectories with the drift integration.

Similarly, consistency is maintained throughout the trajectory even though the drift remains quite small due to the small distance traveled. In Figure 31(a), we can see that the initial



(a) Bias integration for the left vehicle

(b) Bias integration for the right vehicle

Figure 31. Scenario 2: Bias integration with no associations performed

value of the bias estimate ensures that the true location of the blue vehicle is included. The localization results obtained when the data association is active are presented in Figure 32.

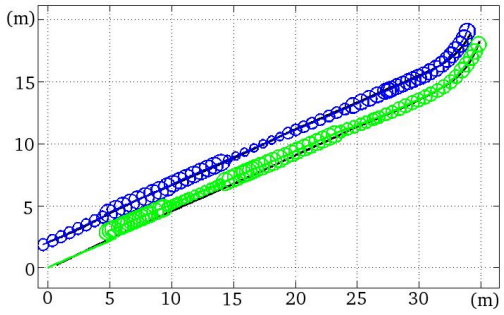
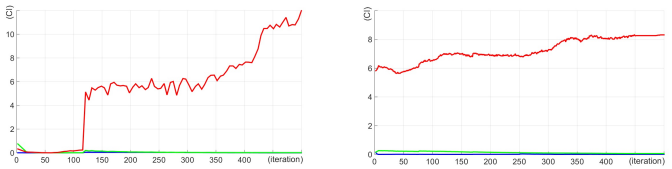


Figure 32. Scenario 2: Trajectories computed by both vehicles with associations performed

The distance between the two vehicles has been recovered and the uncertainties are consequently greatly reduced. It can be noticed that the right vehicle (green one) has a small jump in its localization after several meters. It is caused by an association between landmarks affected by an uncertainty still big enough to allow this jump. Still, the localization is consistent as the vehicle uncertainty is increased by this inaccuracy.

We computed the Consistency Index for both vehicles with and without bias integration as well as when considering data association or not. Results can be seen in Figure 33 where it can be noticed that consistency is ensured for the whole trajectory with our approach.

During the first part of the trajectory, the distance computed by the decentralized algorithm oscillates with a maximum error of 50 centimeters. When associations are found (second part of



(a) CI for the right vehicle (b) CI for the left vehicle

Figure 33. Scenario 2: Consistency Index for both vehicles. In red, CI when considering only the low-level information. In blue, CI with bias integration and no associations performed. In green, CI when the bias is integrated and associations are performed.

the trajectory), the gap is well estimated. The error drops below 10 centimeters (see Figure 34).

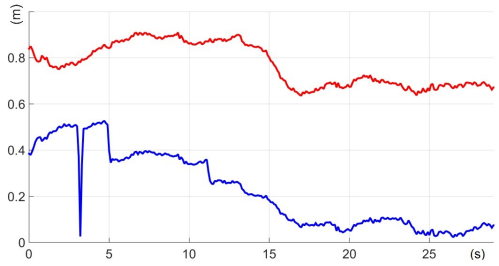


Figure 34. Scenario 2: Root-Mean-Square Error regarding the distance between the two vehicles according to the elapsed time. In blue, RMSE computed with the decentralized algorithm. In red, RMSE based on the SLAM estimates without considering drift.

As expected, the exchange rate remains very small. The bandwidth utilized did not go above 5KB/s per vehicle which remains very far from the current technological limitations.

The last experiment of this paper (scenario 3) presents an example of trajectory with 3 vehicles and an important drift. Each vehicle was moving at 2 meters per second and was equipped with the same sensors as for the previous experiments (a camera and an odometer). Each trajectory was around 125-meter long. The overview of the trajectory is depicted in Figure 35.

Figure 35. Scenario 3: Overview of the trajectory with 3 vehicles. Circles correspond to starting points of the vehicles and crosses to stops.

The 3 vehicles are evolving in a convoy even if the last vehicle starts with a different orientation. The queue and the head of the convoy were separated by approximately 25 meters and the vehicle in the middle is around 15 meters ahead of the rear vehicle. As all the vehicles were not starting at the exact same time, these gaps change quickly after the start of the trajectory. With a 3-vehicle setting, two vehicles start with an unknown initial position. We used the starting point of the leading vehicle (magenta one) as the common frame for the 3 vehicles. It means that the other 2 vehicles are inaccurately localized in this frame. Moreover, the vehicle at the back starts with a large angular bias (around π rad).

The SLAM trajectories, without any association performed, are visible in Figure 36. The number of landmarks tracked was reduced to 15 per image, leading to approximately 100 landmarks mapped per vehicle.

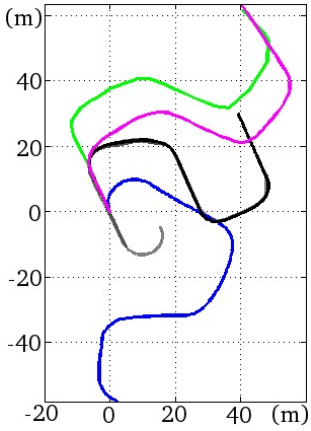


Figure 36. Scenario 3: SLAM trajectories as perceived by the front vehicle. SLAM trajectory of the front vehicle in magenta and its ground truth in black, SLAM trajectory of the vehicle in the middle of the convoy in green with its ground truth in dark gray and SLAM trajectory of the rear vehicle in blue with its ground truth in light gray (as vehicles are following each other, the ground truth for the 3 vehicles are almost superimposed).

With longer distances traveled, the divergence becomes very important, especially the angular error which makes the bends looser than they should be. As expected, the blue vehicle totally diverges compared to its ground truth because of the initial angular error. The two other vehicles also end up far from their ground truth due to a considerable angular drift. For the sake of clarity, we will not expose the uncertainties of the 3 trajectories when taking into account the bias. However, the initial bias uncertainty covers way more than the initial real position for the blue vehicle. Indeed, the idea was to put the data association in a difficult situation with many potential matchings. The initial angular bias uncertainty has been set in order to integrate an initial angular error of 2π radians.

When the data association process is enabled, the decentralized process is able to compute the trajectories visible in Figure 37. Once again, for clarity reasons, only the trajectories are showed.

It illustrates that our decentralized application is able to estimate the gaps separating the vehicles thanks to several associations. The angular drift is, of course, not corrected as it is common to the three vehicles. Without an absolute information (GPS) or a loop closure, it is impossible to correct the orientation as it is affecting the vehicles similarly. The angular diver-

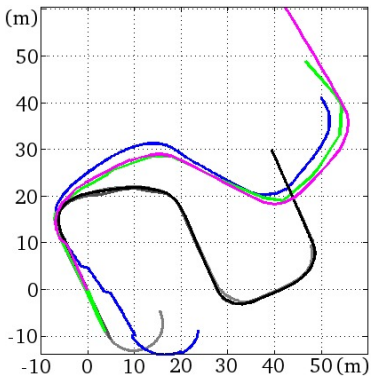


Figure 37. Scenario 3: Trajectories computed by the three vehicles with associations performed. Same color code as previously.

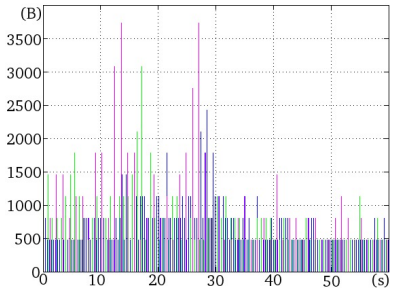


Figure 39. Scenario 3: Quantity of data sent depending on the elapsed time. The blue, green and magenta curves are respectively for the vehicles starting at the back, middle and front of the convoy. The quantity of data is expressed in bytes.

gence is at least constrained in the whole team by associations.

A small gap between the trajectory of the blue vehicle and the two others can be observed. This is due to the fact that the blue vehicle is starting in a bend whereas the others are starting in straight lines. The consequence is that the drift is different at the beginning of the trajectory between the two leading vehicles and the last one. When associations are found, the bias uncertainties are already too large to fully recover the distance separating the vehicles. However, a major part of it is well-estimated and only a small gap remains.

The evolution of the distance separating the different vehicles according to the elapsed time is visible in Figure 38 with each case detailed.

These different results illustrate that our decentralized SLAM process is able to recover a distance profile which is quite similar to the real one. Errors can be explained by two reasons. The first one is related to the evolution of the drift concerning the vehicle position. Each vehicle is affected by a drift which, even though similar, is not the same. One vehicle diverges differently from another and the true distance between the vehicles cannot be entirely recovered due to the high uncertainties surrounding the vehicle positions. A possible improvement for this part would require a more accurate bias model. Moreover, divergence is more likely during bends. It would be interesting to integrate this aspect into the model in order to have an evolution that follows more closely the true nature of the drift.

The second reason for these errors is related to the fact that we only use a high-level data association process. In order to keep our framework general, we chose to only perform associations between 3D points. The direct consequence is that it is almost impossible to lower the error concerning the distance between the vehicles below the threshold used to notify landmarks convergence. By re-projecting the 3D points in the sensor space and tracking them in images, we would be able to have a better accuracy. Of course, this can only be done by specializing our framework for the sensors used by the low-level SLAM. Another solution would be to use sensors providing much accurate landmarks such as stereovision systems or lasers.

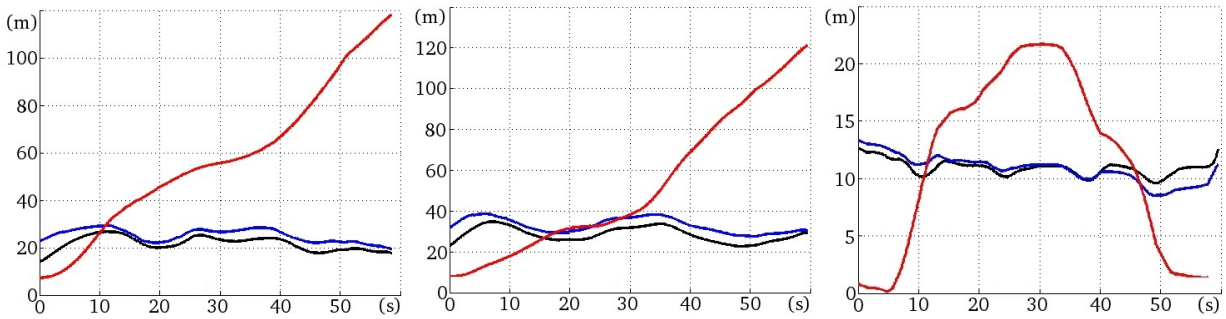
Figure 39 shows the bandwidth requirements for each vehicle.

Once again, the quantity of data sent is very far from technological limitations. It can be observed that the magenta vehicle sends a higher amount of information at once than the other vehicles. This is due to the fact that we used different exchange timers for the three vehicles to show that vehicles do not need to have synchronized communications. The magenta vehicle was sending information less frequently than the two others thus explained the higher quantity of data sent each time. Each vehicle required a little bit less than 10KB/s which is very encouraging for approaches which could use more or different kind of landmarks (edges for instance).

The quality of the vehicle localizations could be improved by mapping more landmarks. Even though the network requirements would not be a problem, the computational time needed would be considerable. Beyond 40 landmarks tracked per image, and with the additional cost of the decentralized layer, it is difficult to increase the number of points in the decentralized SLAM. However, with several optimizations, this limitation could be less restrictive than it is now (feature selection and matching computation moved to the GPU for instance). The decentralized layer is quite light in terms of computational requirements except when linking any information to its bias estimate. Indeed, this operation depends on the size of the state vector and can lead to heavy multiplications inside the Kalman filter. A solution here would be to only connect together bias estimates and compute unbiased landmarks or vehicle poses when needed. This approach is currently being tested. It is also possible to store most of the landmarks out of the state vector and load them only when coming close to where they were first observed. Anyway, our algorithm was running in real time all along the experiments with a camera acquiring 15 images per second, which is suitable for most applications.

6. Conclusion

A general decentralized SLAM solution has been presented. We introduced several innovative elements to solve the multi-vehicle SLAM problem while still maintaining the consistency of the computed localizations. A model representing the natural SLAM drift has been exposed. Its integration inside a SLAM algorithm thanks to an Extended Kalman Filter has also been



(a) Distance between the blue and green vehicles (b) Distance between the blue and magenta vehicles (c) Distance between the green and magenta vehicles

Figure 38. Scenario 3: Distances separating the vehicles all along the trajectory. In black is the true distance (RTK GPS), in red is the distance without data association, in blue is the distance estimated thanks to the decentralized algorithm.

explained. Several simulation results validating its behavior have been showed. They also demonstrate how easy the drift model makes the integration of loop closures or absolute information.

A specifically designed decentralized SLAM architecture has been exposed. By separating the classic SLAM algorithm from the decentralized processing, the use of various feature-based SLAM algorithms is allowed. The organization of the architecture avoids data incest and also handles the communication constraints (losses, latencies, etc.). The decentralized algorithm does not need the initial positions of the vehicles of the team. A prior knowledge can easily be integrated, be it under the form of a distance and orientation to a common frame or an uncertainty. By finding common landmarks between maps from different vehicles, the algorithm is able to recover the relative distance and orientation between the different members of the fleet. A data association process, built for identifying such common map portions, has also been presented.

We carried out several experiments, both simulated and with real data (with 2 and 3 vehicles), to validate our algorithm with vehicles embedding only a camera and an odometer. The results, obtained in real time, show that the bandwidth required by our application is very small. Good localization results are also exposed in these experiments while preserving consistency. The drift model could be improved by making its evolution more tied to the steering angle which seems to be the biggest cause of inconsistency. Current works are directed towards the integration of landmarks in the decentralized map in order to reduce to computational requirements and extend the number of landmarks that could be used.

References

[1] W. Burgard, M. Moors, C. Stachniss, F. Schneider, Coordinated Multi-Robot Exploration, IEEE Transactions on Robotics 21 (3) (2005) 376–386.

[2] C. M. Gifford, R. Webb, J. Bley, D. Leung, M. Calnon, J. Makarewicz, B. Banz, A. Agah, Low-Cost Multi-Robot Exploration and Mapping, in: IEEE International Conference on Technologies for Practical Robot Applications, 2008, pp. 74–79.

[3] H. S. Lee, K. M. Lee, Multi-Robot SLAM Using Ceiling Vision, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 912–917.

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

[5] H. Li, 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.

[6] K. C. Pucihar, P. Coulton, Towards Collaboratively Mapped Multi-View Mobile Augmented Reality, in: Workshop on Mobile Augmented Reality: Design Issues and Opportunities, Mobile Human Computer-Interaction, 2011.

[7] I. M. Rekleitis, G. Dudek, E. E. Milios, Multi-Robot Collaboration for Robust Exploration, in: IEEE International Conference on Robotics and Automation, Vol. 4, 2000, pp. 3164–3169.

[8] N. Barrena, J. R. Sánchez, A. García-Alonso, A Distributed and Collaborative vSLAM Framework for Real-Time Localisation in Huge Environments for Mobile Devices, in: Eurographics, 2013.

[9] J. V. Diosdado, I. T. Ruiz, Decentralised Simultaneous Localisation and Mapping for AUVs, in: 2nd SEAS DTC Technical Conference, 2007, p. A14.

[10] L. L. S. Ong, M. Ridley, J.-H. Kim, E. Nettleton, S. Sukkarieh, Six DoF Decentralised SLAM, in: Australasian Conference on Robotics and Automation, 2003, pp. 10–16.

[11] T. A. Vidal-Calleja, C. Berger, J. Solà, S. Lacroix, Large Scale Multiple Robot Visual Mapping with Heterogeneous Landmarks in Semi-structured Terrain, Robotics and Autonomous Systems 59 (9) (2011) 654–674.

[12] T. Bailey, H. Durrant-Whyte, Simultaneous Localization and Mapping (SLAM): Part II, IEEE Robotics and Automation Magazine 13 (3) (2006) 108–117.

[13] H. Durrant-Whyte, T. Bailey, Simultaneous Localization and Mapping: Part I, IEEE Robotics and Automation Magazine 13 (2) (2006) 99–110.

[14] W. Burgard, M. Moors, D. Fox, R. Simmons, S. Thrun, Collaborative Multi-Robot Exploration, in: IEEE International Conference on Robotics and Automation, Vol. 1, 2002, pp. 476–481.

[15] J. W. Fenwick, P. M. Newman, J. J. Leonard, Cooperative Concurrent Mapping and Localization, in: IEEE International Conference on Robotics and Automation, Vol. 2, 2002, pp. 1810–1817.

[16] J. A. Castellanos, J. Neira, J. D. Tardós, Limits to the Consistency of EKF-Based SLAM, in: 5th IFAC Symposium on Intelligent Autonomous Vehicles, 2004.

[17] A. Martinelli, N. Tomatis, R. Siegwart, Some Results on SLAM and the Closing the Loop Problem, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 2917–2922.

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

[19] G. Bresson, R. Aufrère, R. Chapuis, Consistent Multi-robot Decentralized SLAM with Unknown Initial Positions, in: 16th International Conference on Information FUSION, 2013.
[20] A. Gil, O. Reinoso, M. Ballesta, M. Juliá, Multi-robot visual SLAM using a Rao-Blackwellized particle filter, Robotics and Autonomous Systems 58 (1) (2010) 68–80.
[21] S. P. McLaughlin, R. J. Evans, B. Krishnamurthy, Data Incest Removal in Survivable Estimation Fusion Architecture, in: International Conference on Information Fusion, Vol. 1, 2003, pp. 229–236.
[22] M. Hua, T. Bailey, P. Thompson, H. Durrant-Whyte, Decentralised Solutions to the Cooperative Multi-Platform Navigation Problem, IEEE Transactions on Aerospace and Electronic Systems 47 (2) (2011) 1433–1449.
[23] A. Bahr, M. R. Walter, J. J. Leonard, Consistent Cooperative Localization, in: IEEE International Conference on Robotics and Automation, 2009, pp. 3415–3422.
[24] N. Karam, F. Chausse, R. Aufrère, 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.
[25] E. Nettleton, S. Thrun, H. Durrant-Whyte, S. Sukkarieh, Decentralised SLAM with Low-Bandwidth Communication for Teams of Vehicles, in: Field and Service Robotics, 2006, pp. 179–188.
[26] S. B. Williams, G. Dissanayake, H. Durrant-Whyte, Towards Multi-Vehicle Simultaneous Localisation and Mapping, in: IEEE International Conference on Robotics and Automation, 2002, pp. 2743–2748.
[27] A. Cunningham, M. Paluri, F. Dellaert, DDF-SAM: Fully Distributed SLAM using Constrained Factor Graphs, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.
[28] R. Aragues, E. Montijano, C. Sagues, Consistent Data Association in Multi-Robot Systems with Limited Communications, in: Robotics: Science and Systems, 2010, pp. 51–58.
[29] R. Aragues, J. Cortes, C. Sagues, Dynamic Consensus for Merging Visual Maps under Limited Communications, in: IEEE International Conference on Robotics and Automation, 2010, pp. 3032–3037.
[30] M. Pfingsthorn, B. Slamet, 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.
[31] H. J. Chang, C. S. G. Lee, Y. C. Hu, Y.-H. Lu, Multi-Robot SLAM with Topological/Metric Maps, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 1467–1472.
[32] A. Martin, M. R. Emami, Just-in-time Cooperative Simultaneous Localization and Mapping, in: International Conference on Control, Automation, Robotics and Vision, 2010, pp. 479–484.
[33] R. H. Deaves, D. Nicholson, D. W. Gough, L. A. Binns, P. Vangasse, P. Greenway, Multiple Robot System for Decentralized SLAM Investigations, in: Sensor Fusion and Decentralized Control in Robotic Systems III, Vol. 4196, 2000, pp. 360–369.
[34] S. Thrun, Y. Liu, Multi-Robot SLAM With Sparse Extended Information Filers, in: 11th International Symposium of Robotics Research, 2003, pp. 254–266.
[35] M. M.D.P., W. S. Wijesoma, B. Kalyan, N. M. Patrikalakis, P. Moghadam, Collaborative Multi-Vehicle Localization and Mapping in High Clutter Environments, in: International Conference on Control, Automation, Robotics and Vision, 2010, pp. 1422–1427.
[36] X. S. Zhou, 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.
[37] H. Li, F. Nashashibi, A new method for occupancy grid maps merging: Application to multi-vehicle cooperative local mapping and moving object detection in outdoor environment, in: International Conference on Control, Automation, Robotics and Vision, 2012, pp. 632–637.
[38] H. Li, F. Nashashibi, Multi-vehicle cooperative localization using indirect vehicle-to-vehicle relative pose estimation, in: IEEE International Conference on Vehicular Electronics and Safety, 2012, pp. 267–272.
[39] A. Cunningham, K. M. Wurm, W. Burgard, 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.
[40] J. Neira, J. D. Tardós, Data Association in Stochastic Mapping Using the Joint Compatibility Test, IEEE Transactions on Robotics and Automation 17 (6) (2002) 890–897.
[41] J. Neira, J. D. Tardós, J. A. Castellanos, Linear time vehicle relocation in SLAM, in: IEEE International Conference on Robotics and Automation, Vol. 1, 2003, pp. 427–433.
[42] J. Montiel, J. Civera, A. J. Davison, Unified Inverse Depth Parametrization for Monocular SLAM, in: Robotics: Science and Systems, Philadelphia, USA, 2006.
[43] A. J. Davison, Real-Time Simultaneous Localisation and Mapping with a Single Camera, in: IEEE International Conference on Computer Vision, Nice, France, 2003, pp. 1403–1410.
[44] G. Bresson, T. Féraud, R. Aufrère, P. Checchin, R. Chapuis, A New Strategy for Feature Initialization in Visual SLAM, in: IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Perception and Navigation for Autonomous Vehicles in Human Environment, 2011, pp. 115–120.
[45] S. Huang, G. Dissanayake, Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM, IEEE Transactions on Robotics 23 (5) (2007) 1036–1049.
[46] Y. Bar-Shalom, X. R. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation, Wiley-Interscience, 2001.
[47] S. Julier, J. Uhlmann, Building a Million Beacon Map, in: Sensor Fusion and Decentralized Control in Robotic Systems IV, Vol. 4571, 2001, pp. 1–9.
[48] S. Julier, J. Uhlmann, General Decentralized Data Fusion with Covariance Intersection, CRC Press, 2001, chapter 12.
[49] T. Bailey, J. Nieto, J. Guivant, M. Stevens, E. Nebot, Consistency of the EKF-SLAM Algorithm, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 3562–3568.
[50] U. Frese, A Discussion of Simultaneous Localization and Mapping, Autonomous Robots 20 (1) (2006) 25–42.
[51] C. Estrada, J. Neira, J. D. Tardós, Hierarchical SLAM: real-time accurate mapping of large environments, IEEE Transactions on Robotics 21 (4) (2005) 588–596.
[52] F. Dellaert, M. Kaess, Square Root SAM: Simultaneous localization and mapping via square root information smoothing, The International Journal of Robotics Research 25 (12) (2006) 1181–1203.
[53] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, F. Dellaert, iSAM2: Incremental smoothing and mapping using the Bayes tree, The International Journal of Robotics Research.
[54] S. J. Julier, J. K. Uhlmann, Using covariance intersection for slam, Robotics and Autonomous Systems 55 (1) (2007) 3–20.
[55] S. Roumeliotis, G. Sukhatme, G. A. Bekey, et al., Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization, in: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, Vol. 2, IEEE, 1999, pp. 1656–1663.
[56] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, J. Tardós, A comparison of loop closing techniques in monocular SLAM, Robotics and Autonomous Systems 57 (12) (2009) 1188–1197.
[57] G. Bresson, R. Aufrère, R. Chapuis, Loop Closing in a Drift-Aware Monocular SLAM, in: IFAC Intelligent Autonomous Vehicles Symposium, 2013.
[58] D. M. Cole, P. M. Newman, Using Laser Range Data for 3D SLAM in Outdoor Environments, in: IEEE International Conference on Robotics and Automation, 2006, pp. 1556–1563.
[59] D.-S. Seo, D. Won, G.-W. Yang, M.-S. Choi, S.-J. Kwon, J. W. Park, A Probabilistic Approach for Mobile Robot Localization under RFID Tag Infrastructures, in: International Conference on Control, Automation and Systems, 2005, pp. 1797–1801.
[60] G. Bresson, R. Aufrère, R. Chapuis, Making Visual SLAM Consistent with Geo-Referenced Landmarks, in: IEEE International Conference on Intelligent Vehicles, 2013.
[61] G. Conte, P. Doherty, Vision-Based Unmanned Aerial Vehicle Navigation Using Geo-Referenced Information, EURASIP Journal On Advances In Signal Processing (2009) 10–32.
[62] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, Vol. 1, MIT Press Cambridge, 2005.
[63] L. M. Paz, J. D. Tardós, J. Neira, Divide and Conquer: EKF SLAM in O(n), IEEE Transactions on Robotics 24 (5) (2008) 1107–1120.
[64] G. Bresson, R. Aufrère, R. Chapuis, Real-time Decentralized Monocular SLAM, in: International Conference on Control, Automation, Robotics and Vision, 2012.