Loop Closing in a Drift-Aware Monocular SLAM

Guillaume Bresson, Romuald Aufrère, Roland Chapuis

Loop Closing in a Drift-Aware Monocular SLAM

Guillaume Bresson*, Romuald Aufrère*† and Roland Chapuis*
*Institut Pascal - UMR 6602 CNRS – †LIMOS - UMR 6158 CNRS
Clermont Université, Université Blaise Pascal - Aubière, France
firstname.name@univ-bpclermont.fr

***Abstract*—This paper presents a real-time Consistent Monocular EKF-SLAM process. We introduce the notion of bias which allows to model the natural drift of the SLAM process. Thanks to it, the consistency of the filter is guaranteed. By connecting the bias to the different landmarks and to the vehicle pose, the estimates become tightly bound to the SLAM drift. It means that a loop closure, for instance, will naturally estimate the bias and so correct the vehicle pose and landmark positions without any special processing. We developed a dedicated architecture in order to integrate the bias. It uses an Extended Kalman Filter and has the advantage to be totally decorrelated from the classical SLAM process. Thanks to it, any algorithm, with any kind of sensors or methods, can be used instead of the monocular SLAM employed in this paper, as long as it produces landmark estimates and their uncertainty. This approach was validated with a real experiment composed of a long loop.**

I. Introduction

The Simultaneous Localization And Mapping (SLAM) problem is considered as the key element for truly autonomous vehicles. Over the past two decades, many researchers involved in the mobile robotics field have proposed different methods to achieve this goal [12]. A very broad range of solutions with different approaches and sensors has been published [1]. However, some issues still need to be tackled in order to have systems working regardless of the trajectory performed and at a reasonable cost.

One of the least expensive solutions relies on a system based on a single camera. It has also the advantage of being easy to calibrate. Stereo vision, though cheap, requires an important calibration phase. Nevertheless, with a single camera, estimating the depth of a landmark is not an easy task. Indeed, it requires several observations of the same point with enough parallax to refine accurately its position. Meanwhile, the successive linearizations necessary to achieve this precision are risky because done far from the true value [4]. Consequently, estimated landmarks, and so the vehicle pose, can be wrong. Some solutions are able to drastically reduce this undesirable effect [6][8] without totally removing it. Using only one camera also adds the problem of the scale factor estimation. Indeed, with such a setting, it is impossible to estimate the scale of the map built. More problematic, this scale factor slowly drifts along the trajectory inducing inconsistency [20]. This effect can be countered by using an odometer and thus constraining the scale factor. The odometer error is cumulative and also tends to diverge. However, the results produced by these algorithms are good enough to rely on these sensors [5]. In this paper, a single camera, odometric information and the steering angle are used to perform SLAM.

However, besides scale factor and odometry, SLAM processes themselves tend to drift over time [3] providing wrong vehicle and landmark localizations. Though good convergence properties have been demonstrated in the linear case [11], real world applications imply non-linear equations that do not guarantee the consistency of the estimation process (Extended Kalman Filter and Particle Filters) [17]. It can be seen as: the longer the distance traveled, the more the vehicle localization will be subject to inconsistency. This means that the true pose of the vehicle will not be included in the uncertainty of the estimated vehicle state. Several authors have discussed the importance of this phenomenon [7][18] and how it is tightly bound to the error concerning the orientation estimate [2][15]. Although these results have not been demonstrated, experiments tend to confirm them. Another central element is the distance between the point around which the model is linearized and the true point. The farther the estimate is, the more likely inconsistency happens [16]. This is a crucial point that authors try to avoid by using submapping methods [14]. By limiting the size of the submaps, it is possible to reduce the linearization errors and consequently the drift. However, it only works at a submap level. The global map, keeping track of all the submaps, is still affected by the drift and it is only by finding links between the submaps that the drift can be reduced.

Being able to recognize previously mapped areas, and so reduce the SLAM drift, has become a whole topic called loop closing. It is a difficult task because of the inconsistency of the estimation process. Indeed, the estimated vehicle pose is not reliable and so not a good help to track loop closure. It means that a special algorithm, separated from the estimation process, must be dedicated to identifying previously mapped areas. Once a loop closure is detected, the new knowledge must be incorporated into the global map with a special treatment. Most approaches use a camera to identify loop closing. Visual bag of words [13], joint compatibility between landmarks [10] or classifiers specifically trained to recognize previously mapped areas [21] are examples of methods able to detect loop closing with a high degree of certitude. However, these are only detectors and as such they do not consider integrity. Indeed, even after closing the loop, when the drift is drastically reduced, consistency is still not guaranteed. For instance, the part of the trajectory located at the opposite side of the loop might still be affected by drift.

As mentioned before, monocular SLAM is particularly

prone to linearization errors and so, is even more concerned by the SLAM drift. Combined with the changing scale factor induced by the use of a single camera (though odometry will limit this problem), consistency is impossible to achieve with long trajectories. In this paper, we propose a general framework to take into account the SLAM drift in an Extended Kalman Filter. This framework can be used with any kind of SLAM algorithm as long as it provides landmark estimates and uncertainties. Though it is applied here to the monocular problem, any other sensor can be utilized without any constraint. Loop closure algorithms presented before can also be added in order to certify when fusion between landmarks must be accomplished. The drift can also be estimated thanks to georeferenced information which can easily be integrated thanks to the bias framework.

Our contributions are the following:

- Modeling the SLAM drift
- Integrating it in an EKF in order to guarantee data integrity and perform seamlessly loop closing
- Building a general framework usable by any estimation algorithm
- Validating our approach with real experiments in a monocular context

This paper will be organized as follows: Section II will explain the notion of bias which can be seen as the SLAM drift. Then, section III will introduce the architecture built around the bias and how it is integrated into the EKF. Section IV will eventually present the experiments realized and detail the results obtained.

II. Notion of bias

The direct consequence of the SLAM divergence is that there is a gap between where the vehicle thinks it is and where it truly is. This gap can be seen as a bias of the estimation process. Indeed, though information is properly interpreted inside the vehicle frame, the vehicle pose itself is affected by a bias which has an impact on the landmark locations in a global frame. By estimating accurately this bias, it is possible to correct the vehicle pose and so the landmark positions.

The bias can also be perceived as the parameters allowing the transformation from the estimation frame to a non-divergent frame. As we are evolving with 6D of freedom (3D position and 3 associated angles), the transformation is set by 6 parameters:

$$\mathbf{b} = \begin{pmatrix} b_x & b_y & b_z & b_{\alpha_x} & b_{\alpha_y} & b_{\alpha_z} \end{pmatrix}^T$$

with b_x, b_y and b_z being a position bias and b_{α_x}, b_{α_y} and b_{α_z} an angular one. Similarly, a vehicle can be defined as:

$$\mathbf{v} = \begin{pmatrix} v_x & v_y & v_z & v_{\alpha_x} & v_{\alpha_y} & v_{\alpha_z} \end{pmatrix}^T$$

Considering that the vehicle state $\mathbf{v}$ is expressed in a biased frame, we can define G, an unbiased frame which actually represents a consistent estimation of the true world. Passing from the biased frame to G can be done easily for the vehicle state as the bias is additive:

$$\mathbf{v}_G = \mathbf{R}\mathbf{v} + \mathbf{b} \tag{1}$$

with $\mathbf{R}$ being:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{3D}(b_{\alpha_z}, b_{\alpha_y}, b_{\alpha_x}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}$$

where $\mathbf{R}_{3D}$ is the product of 2D rotation matrices. With a landmark i:

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

we can express the transformation to the non-divergent frame as:

$$\mathbf{l}_{G_i} = \mathbf{R}_{3D}(b_{\alpha_{z\,i}}, b_{\alpha_{y\,i}}, b_{\alpha_{x\,i}})\mathbf{l}_i + \begin{pmatrix} b_{x_i} \\ b_{y_i} \\ b_{z_i} \end{pmatrix} \tag{2}$$

where $\mathbf{b}_i$ is the bias associated to the landmark $\mathbf{l}_i$.

Nevertheless, a loop closure, or a sensor giving an absolute localization, is needed in order to estimate the bias and have an idea of the drift. Still, without it, we need to be able to guarantee the consistency of the given localization. Consequently, each parameter composing the bias has a dedicated uncertainty. The non-linear functions (1) and (2) linking the biased state to the true one allow to compute an associated covariance matrix for each landmark and for the vehicle pose thanks to the corresponding Jacobians.

Restricting the bias to a single estimation is inaccurate as it tends to change depending on the distance traveled. For instance, if we consider that at the beginning of a trajectory the bias is non-existent, after several meters it will not be the case. Consequently, its evolution can be indexed according to the distance traveled, meaning the curvilinear abscissa. To do so, it is necessary to have a dynamic model allowing to make the bias evolve following the curvilinear abscissa and so the trajectory. The value of the bias at one point of the trajectory is tightly bound to the distance between its previous estimation and the distance between those two estimations. We used an AR(1) model to represent the evolution of the bias. It can be written as:

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

where $\boldsymbol{\varepsilon}$ is a 6-parameter vector containing the drift between the curvilinear abscissas $s-1$ and s. $\boldsymbol{\varepsilon}$ is assumed to be a white noise and so cannot be characterized. This lack of knowledge will be integrated into its associated covariance $\mathbf{P}_{\boldsymbol{\varepsilon}}$. Without any estimation of the bias (with a loop closing for instance), its uncertainty ($\mathbf{P}_{\mathbf{b}_s}$ here) will grow in order to keep the consistency of the filter. A bias estimation will lower the uncertainty of the vehicle pose and of all the landmarks concerned. Indeed, by linking each bias estimation to the previous, it is possible to apply, without further computation or special treatment, the update of the bias to all the landmarks concerned. Of course, landmarks located at the opposite side of the loop closing point will be less affected by the update that those in the vicinity of the closing point. Nevertheless, it will be done naturally depending on how the bias has evolved.

III. Bias integration

A. Architecture

Integrating the bias estimation directly into the SLAM algorithm has many drawbacks especially in a monocular case. Indeed, the bias is an estimation of the divergence of the vehicle pose and consequently of the landmarks. When a landmark is added to the SLAM process, its uncertainty is important because of the lack of information about its depth. Adding the bias will only mix depth and bias uncertainties. The result is that, after a few iterations, the landmark will slowly converge towards its value biased by the divergence of the SLAM and reduce the uncertainty without separating the one caused by the bias from the uncertainty of the depth estimate. At the end, the bias uncertainty will be close to zero, thus losing the filter consistency.

Likewise, mixing the uncertainties will make possible some landmark positions previously unavailable. For instance, with a big enough bias, the landmark uncertainty could include locations behind the observer. Thus, it would become possible to update a landmark just observed behind the camera. It would cause wrong estimates (for landmarks and the vehicle) and would not correct the divergence. This action can only be done thanks to a loop closing (or with absolute localizations). Consequently, integrating the bias inside the estimation process is not possible without causing important damage to the system. Moreover, by tightly linking the bias to the SLAM process, it would prevent any other algorithm (EKF or PF, camera or LRF...) to directly integrate the bias method proposed here, which is very restrictive.

Here, we propose a totally separated estimation process ("high-level SLAM") which collects accurate landmarks out of a classic SLAM (often called in this paper "low-level") and handles the bias. Though applied here to a monocular case, any SLAM algorithm furnishing landmarks and their uncertainty can be used. It means that the sensors used in the SLAM, and the SLAM method itself (EKF, PF...) do not matter. This separation also allows the low-level SLAM to have no memory. Indeed, as collecting the landmarks is relegated with the bias handling in another process, the SLAM filter just needs to transfer the landmarks which have converged to the high-level process which will handle them afterwards. The direct consequence is that the low-level SLAM algorithm comes closer to visual odometry in our case, making it faster because of the lighter maps. Of course, only accurate landmarks are kept in the high-level SLAM because they are the ones relevant when closing a loop (finding a loop with inaccurate landmarks can be complicated). Bringing the modifications made in the low-level to the high-level is not interesting either as it would force us to break the map and re-build it with the updated links which is costly.

When transferring landmarks to the high-level, there is no need to keep the links established in the covariance matrix between the vehicle and the landmarks and between the landmarks themselves. Only the landmark positions and their own uncertainty must be kept. As the landmarks transferred are those which have converged (small uncertainty below a threshold), the links that connect them thanks to the bias is way more important than the ones established during the SLAM process, making them meaningless regarding the estimation process. It is an important asset for applications with communication needs (multi-vehicle). Figure 1 shows how the SLAM process is connected to the algorithm which takes care of the bias. In this Figure, $\mathbf{P}_*$ represents the covariance of $*$.

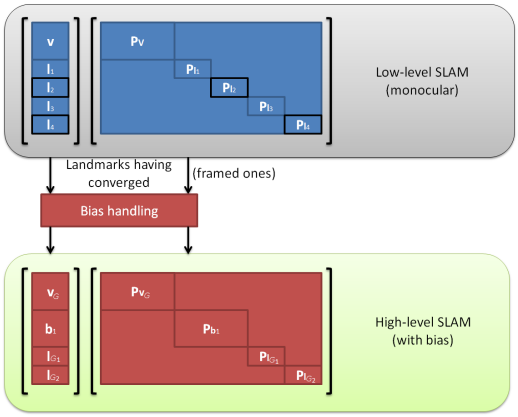

Fig. 1. Architecture around the bias estimation

In Figure 1, we can see that only the landmarks having converged (framed ones) are sent to the high-level SLAM which handles the bias. The vehicle pose can also be sent in order to have an up-to-date estimate. The first bias is initialized at the beginning of the trajectory. Each new arriving landmark is then connected to the whole state vector thanks to the bias. The low-level SLAM could remove the landmarks sent to the higher-level SLAM. However, they are kept until they cannot be seen because they are very accurate and thus help maintaining the new landmarks which are less precise. In order to not distract the monocular SLAM, all the actions performed to handle the bias are done in a separated thread. Landmarks that should be added to the high-level SLAM are placed in a queue in which a timed thread iterates. Figure 2 shows how the whole process is working.

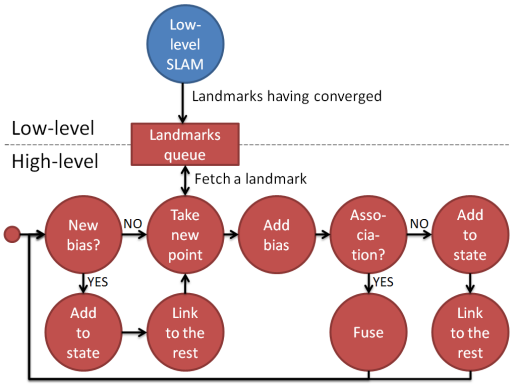

Fig. 2. Algorithm around the bias estimation

It can be seen that the monocular SLAM is interacting with the high-level SLAM only by a FIFO. Each landmark

is taken from the FIFO and an association is sought with the landmarks already in the state vector. If none is found, the landmark is added via a special process which establishes the links with the rest of the state vector. In case of an association, a fusion process is triggered. It will allow to estimate the bias and reduce its uncertainty throughout the whole map. As explained before, each landmark is affected by a unique bias. However, initializing a bias per landmark is not suitable for a real time process. Consequently, a condition (based on the distance traveled) is checked in order to know whether or not a new bias must be initialized and inserted into the state vector.

B. Integration

The integration of the bias is handled by an Extended Kalman Filter. It offers several advantages in our situation. The main one is bound to the fact that each bias is linked to the previous one. One could integrate this information using the non-linear function linking landmarks and biases and then compute the Jacobians associated in order to create this link in the global covariance matrix. However, this task can easily be handled by the EKF. Moreover, the Extended Kalman Filter is well-known to facilitate the integration of various information sources in order to furnish the best estimate possible. With this in mind, the filter in charge of the bias could also be used to integrate information coming from the infrastructure for instance. Geo-localized landmarks would help to reduce the drift and would be easy to add in the EKF. A GPS would also come in handy to provide an absolute localization in order to constrain the drift to the GPS uncertainty. With the EKF, it can easily be done.

The vehicle pose is affected by a bias whose value depends on the curvilinear abscissa. It means that a landmark initialized at an abscissa $s-1$ will be subject to a different bias that another one initialized at s. However, integrating a bias per landmark can be costly as it means that the size of the state vector will grow quickly. In order to limit the load of the filter, we chose to integrate a new bias estimate according to the distance traveled. For example, landmarks initialized between the abscissas $s-1$ and s will be affected by the bias $\mathbf{b}_{s-1}$. This approximation is suitable as long as the distance between each bias estimate is not too important.

In order to create the links between the different biases and landmarks already in the state vector, the initialization of a new bias is performed thanks to the EKF. The new bias is first inserted into the state vector and an infinite variance is added into the covariance matrix. An update step of the EKF is then accomplished to get the true uncertainty and create the links with the rest of the state vector. Based on Equation (3) and after having inserted the new bias with an infinite variance, we are able to refine the uncertainty by considering the difference between the two consecutive biases. Let define h_bn the observation function used in the update step of the EKF. We observe here the evolution of the bias, meaning the difference between two consecutive bias estimates ($\mathbf{b}_s$ and $\mathbf{b}_{s-1}$). We can then compute the associated Jacobian $\mathbf{H_b}$:

$$\mathbf{H_b} = \begin{bmatrix} \mathbf{0}_{6\times\dots} & -\mathbf{I}_{6\times6} & \mathbf{0}_{6\times\dots} & \mathbf{I}_{6\times6} \end{bmatrix} \tag{4}$$

The identity matrices correspond to where the biases $\mathbf{b}_{s-1}$ and $\mathbf{b}_s$ are stored inside the state vector. Similarly, the observation noise is defined as:

$$\mathbf{R_b} = \mathbf{P}_{\mathbf{b}_s} - \mathbf{P}_{\mathbf{b}_{s-1}} \tag{5}$$

where $\mathbf{P}_{\mathbf{b}_s}$ is the current uncertainty associated to the bias and $\mathbf{P}_{\mathbf{b}_{s-1}}$ the previous one. The Kalman gain can then be computed and applied to the covariance uncertainty, thus recreating all the missing links between the new bias and the rest of the state vector.

Landmark initialization is a little bit different though starting the same way. First, the landmark is inserted into the state vector with an infinite uncertainty. The bias is added to the landmark thanks to Equation (2). It is then followed by an update step from the Kalman gain. In our case, the observation $\mathbf{z}_l$ is the value of the landmark out of the low-level SLAM (just after convergence) and which is affected by the drift:

$$\mathbf{z}_l = \mathbf{l}_i$$

$$\mathbf{R}_l = \mathbf{P}_{\mathbf{l}_i}$$

where $\mathbf{R}_l$ is the observation noise meaning the uncertainty of the landmark out of the low-level SLAM ($\mathbf{P}_{\mathbf{l}_i}$). Consequently, the observation function h_l is the non-linear function allowing to pass from a drift-aware landmark to a classic one (it can be seen as the inverse of Equation (2)):

$$h_l(\mathbf{b}_i, \mathbf{l}_{G_i}) = \mathbf{R}_{3D}^T(b_{\alpha_{z_i}}, b_{\alpha_{y_i}}, b_{\alpha_{x_i}}) \left(\mathbf{l}_{G_i} - \begin{pmatrix} b_{x_i} \\ b_{y_i} \\ b_{z_i} \end{pmatrix} \right) \tag{6}$$

$\mathbf{H}_l$, the Jacobian of h_l can then be computed. The details will not be given for space purposes but deriving according to $\mathbf{b}_i$ and $\mathbf{l}_{G_i}$ is easy. Once the EKF update is performed, the new landmark will take into account the bias and be linked with all the previous landmarks and biases.

The main point of this EKF is to perform loop closing, meaning association between two landmarks here, in order to estimate the bias and reduce its uncertainty which impacts all the landmarks. The fusion process is very similar to what have been described before. Let consider a landmark $\mathbf{l}_j$, coming from the low-level SLAM, that has been associated with a landmark $\mathbf{l}_{G_i}$ in the high-level map. $\mathbf{l}_j$ is affected by a bias $\mathbf{b}_j$ already inserted into the state vector. This can be summed up this way: $\mathbf{l}_{G_i}$ and $\mathbf{l}_{G_j}$ represent the same landmark but affected by a different bias. We can thus specify the observation of the Kalman filter and its noise as (f stands here for fusion):

$$\mathbf{z}_f = \mathbf{l}_j$$

$$\mathbf{R}_f = \mathbf{P}_{\mathbf{l}_j}$$

We then define the non-linear function h_f whose objective is to link the landmark already in the state vector $\mathbf{l}_{G_i}$ with the current bias $\mathbf{b}_j$. Similarly to Equation (6):

$$h_f(\mathbf{b}_j, \mathbf{l}_{G_i}) = \mathbf{R}_{3D}^T(b_{\alpha_{z_j}}, b_{\alpha_{y_j}}, b_{\alpha_{x_j}}) \left(\mathbf{l}_{G_i} - \begin{pmatrix} b_{x_j} \\ b_{y_j} \\ b_{z_j} \end{pmatrix} \right) \tag{7}$$

The innovation of the EKF becomes:

$$\mathbf{\Delta} = \mathbf{z}_f - h_f(\mathbf{b}_j, \mathbf{l}_{G_i}) \tag{8}$$

Just as before, the computation of the Jacobian $\mathbf{H}_f$ associated to the function h_f is straightforward and will not be detailed here. The Kalman gain can then be computed and the correction applied. The current bias uncertainty will decrease towards the bias associated to the landmark already inserted. As all the links between the different biases and landmarks have already been established, the whole state vector will be naturally (more or less) affected by this fusion.

IV. Experiments

In order to validate the principle of the bias exposed in this paper, a trajectory with a loop was performed. However, in order to detect loop closures, a data association process must be designed. In the monocular context we are in, many algorithms doing so can be found [9][19].

Nevertheless, they can be costly or difficult to implement. The idea was to validate the way the bias is working over loop closing, not to create a new algorithm able to detect properly all loop closures. This is why we chose to stick to a simple algorithm illustrating the efficiency of our method. A better algorithm could still be implemented once the bias concept was fully validated. Here, individual compatibility between new landmarks and those already in the high-level map is tested. Thanks to the fact that the integrity of the filter is kept by taking into account the bias, individual compatibility is a valid measure. The landmarks that are said compatible with an incoming landmark are then evaluated via a correlation between their patches. The best score above a fixed threshold indicates an association. As a consequence the two landmarks can be fused.

The experiment presented here was accomplished with an electrical vehicle (VIPALAB) on an urban realistic platform called PAVIN. This one is composed of roads, sidewalks, buildings, roundabouts... A single camera was used, combined with odometric information and the steering angle. A Real Time Kinematic (RTK) GPS was also equipped but only for comparison purposes. It served to compare the results obtained after loop closing to an accurate ground truth in order to evaluate the quality of our method. The vehicle was moving at approximately 2 meters per second. The camera was mounted inside the vehicle on the ceiling. It was acquiring images at 15 FPS.

The trajectory is about one-hundred-meter long. It is composed of a long loop and continues on the same path for about 10 meters. The monocular SLAM mapped around 60 landmarks over the trajectory. Those landmarks are those considered accurate enough (uncertainty below a threshold) to be used in the high-level SLAM in order to estimate the bias. The whole application was running in real-time. Figure 3 shows the results of the monocular SLAM algorithm compared to the ground truth.

The black curve is the ground truth and red curve is the trajectory built by the low-level SLAM. There is an important odometry drift. Also, it can be seen at the end of the trajectory that the angle is wrong and so the vehicle is slowly deviating from the true path. More importantly, the red ellipses, representing the uncertainty associated to the vehicle pose, show that very quickly the filter loses the integrity. The blue ellipses illustrate the uncertainty when considering the bias. Here, the process does not look for associations but just considers the bias for each landmark and vehicle pose. It can be seen that the integrity is preserved during the whole trajectory. The high-level SLAM is consistent whereas the low-level one is not.

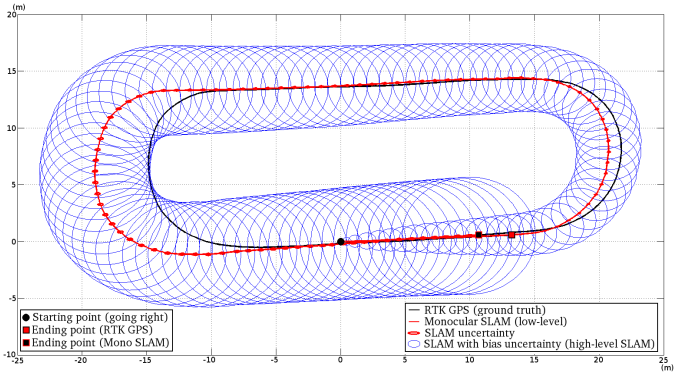

Fig. 3. Results of a drift-aware monocular SLAM

Figure 4 shows the same results but when associations between landmarks are performed.

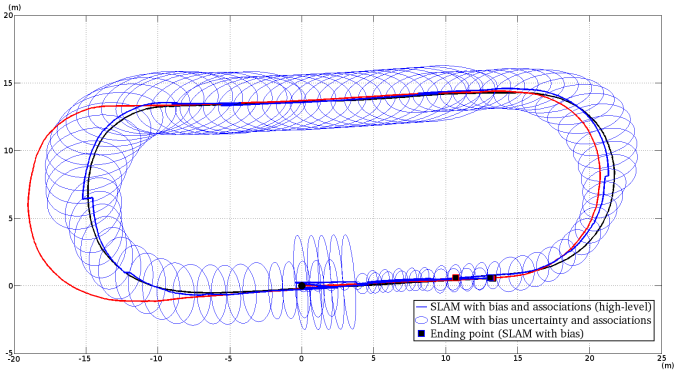

Fig. 4. Results of a drift-aware monocular SLAM with loop closing

The blue curve is the result of the algorithm when associations are accomplished. We can see that an important part of the odometry drift is corrected. Similarly, the angle is properly adjusted. The sudden jumps in the trajectory are due to the fact that we initialize a bias according to the curvilinear abscissa. A new bias is impacted by different values causing those small jumps. More biases would smooth the whole trajectory. The blue ellipses also demonstrate that the consistency is still guaranteed even after having associated landmarks. The error is smaller than previously thanks to the fusions realized. With more associations, the uncertainty would be even more reduced. In a similar manner, by continuing the trajectory and finding associations, the process would be able to refine the

different bias estimates and so come closer to the ground truth.

Figure 5 shows the distance between the vehicle pose taking into account the bias and the RTK GPS all along the trajectory. Only the longitudinal and lateral errors are shown here.

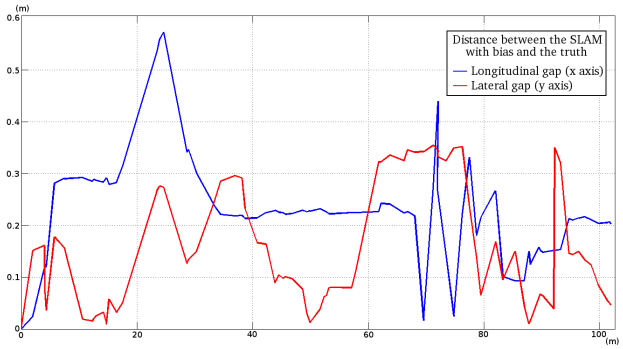

Fig. 5. Distance between the computed vehicle position and the truth all along the trajectory

The average error is around 20 centimeters. Some peaks can be seen. They usually happen when a new bias should be initialized (the bias covers a too-long distance to be accurate all along). Using more biases would help. This error is also bound to the fact that the landmarks used, even if they are accurate, are subject to a small error that can affect the bias estimation. More associations would help to converge towards the truth.

V. Conclusion

A consistent EKF-SLAM process has been presented. It takes into account the natural SLAM drift in order to guarantee the integrity of the estimation. We represent it as a bias of the estimation process and model it thanks to an AR(1) model. Each bias in only bound to the previous one and to the distance traveled in between. This representation allows to guarantee the integrity of the estimation process even when no loop closures are accomplished.

By integrating it into an Extended Kalman Filter, we are able to connect each landmark and vehicle pose to the drift. The direct consequence is that loop closures will be operated naturally without requiring a special treatment.

We chose to separate the bias estimation from the classical SLAM. The main advantage is that any SLAM algorithm can be utilized. Every existing method, able to produce landmark and vehicle estimates (and their uncertainty), can be easily connected to the high-level SLAM. No specific sensor is required as well.

We validated our approach over a trajectory of 100 meters in a monocular context. The results show that the consistency of the filter is preserved even without a loop closure. When associations are found, the drift is corrected and the vehicle pose tends towards its true value. After several associations, the integrity of the estimate is still guaranteed. All these actions were performed in real time.

However, a more robust data association would help when performing loop closures. Indeed, with the simple algorithm described in this paper, some associations are missed. A better data association algorithm could allow to estimate more accurately the drift. We also plan to extend the notion of bias to the multi-vehicle case. The separated architecture presented in this paper could also fuse estimates coming from different vehicles.

References

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

[2] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM Algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, 2006.

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

[4] K. E. Bekris, M. Glick, and L. E. Kavraki. Evaluation of Algorithms for Bearing-Only SLAM. In *IEEE International Conference on Robotics and Automation*, pages 1937–1943, 2006.

[5] G. Bresson, T. Féraud, R. Aufrère, P. Checchin, and 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*, pages 115–120, 2011.

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

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

[8] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse Depth Parametrization for Monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

[9] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-Point RANSAC for EKF Filtering. Application to Real-Time Structure from Motion and Visual Odometry. *Journal of Fields Robotics*, 27(5):609–631, 2010.

[10] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardós. Mapping Large Loops with a Single Hand-Held Camera. In *Robotics: Science and Systems*, 2007.

[11] M. W. M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

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

[13] E. Eade and T. Drummond. Unified Loop Closing and Recovery for Real Time Monocular SLAM. In *British Machine Vision Conference*, 2008.

[14] C. Estrada, J. Neira, and J. D. Tardós. Hierarchical SLAM: real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[15] U. Frese. A Discussion of Simultaneous Localization and Mapping. *Autonomous Robots*, 20(1):25–42, 2006.

[16] S. Huang and G. Dissanayake. Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM. *IEEE Transactions on Robotics*, 23(5):1036–1049, 2007.

[17] S. Julier and J. Uhlmann. Building a Million Beacon Map. In *Sensor Fusion and Decentralized Control in Robotic Systems IV*, volume 4571, pages 1–9, 2001.

[18] A. Martinelli, N. Tomatis, and R. Siegwart. Some Results on SLAM and the Closing the Loop Problem. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2917–2922, 2005.

[19] J. Neira, J. D. Tardós, and J. A. Castellanos. Linear time vehicle relocation in SLAM. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 427–433, 2003.

[20] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale Drift-Aware Large Scale Monocular SLAM. In *Robotics: Science and Systems*, 2010.

[21] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.