Real-Time Monocular SLAM With Low Memory Requirements

Guillaume Bresson, Thomas Féraud, Romuald Aufrère, Paul Checchin, Roland Chapuis

Real Time Monocular SLAM with Low Memory Requirements

Guillaume Bresson*, Thomas Féraud*, Romuald Aufrère*†, Paul Checchin* 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*—The localization of a vehicle in an unknown environment is often solved using Simultaneous Localization And Mapping (SLAM) techniques. Many methods have been developed, each requiring a different amount of landmarks (map size), and so of memory, to work efficiently. Similarly, the required computational time is quite variable from one approach to another. In this paper, we focus on the monocular SLAM problem and propose a new method, called MSLAM, based on an Extended Kalman Filter (EKF). The aim is to provide a solution that has low memory and processing time requirements and that can achieve good localization results while benefiting from the EKF advantages (direct access to the covariance matrix, no conversion required for the measures or the state). To do so, a minimal Cartesian representation (3 parameters for 3 dimensions) is used. However, linearization errors are likely to happen with such a representation. New methods allowing to avoid or hugely decrease the impact of the linearization failures are presented. The first contribution proposed here computes a proper projection of a 3D uncertainty in the image plane, allowing to track landmarks during longer periods of time. A corrective factor of the Kalman gain is also introduced. It allows to detect wrong updates and correct them, thus reducing the impact of the linearization on the whole system. Our approach is compared to a classic SLAM implementation over different data sets and conditions so as to illustrate the efficiency of the proposed contributions. The quality of the map built is tested by using it with another vehicle for localization purposes. Finally, a public data set, presenting a long trajectory (1.3 km) is also used in order to compare MSLAM to a state-of-the-art monocular EKF-SLAM algorithm, both in terms of accuracy and computational needs.**

I. Introduction

Vehicles able to drive in total autonomy are one of the main objectives of the Intelligent Transportation Systems community. One way to fulfill this task is, for a vehicle, to be able to localize itself in unknown environments. This topic, often referred to as Simultaneous Localization And Mapping (SLAM), has been widely studied and is now considered a key part of the mobile robotics field. Over the last twenty years, many solutions have been proposed using different methods and sensors [13]. However, so as to improve accuracy and efficiency, authors tend to combine sensors [2] therefore raising costs. In order to extend the use of autonomous vehicles, it is necessary to build solutions relying on cheap sensors. In this paper, the proposed solution only uses a single camera and an odometer.

Estimating the pose of a vehicle can be accomplished in various ways: particle filters [14], Bundle Adjustment methods (BAs) [17], Extended Kalman Filters (EKFs) [24] and Unscented Kalman Filters [6] being the most common ones. Each method is different and has a strong impact on how the whole system should be designed. However, similar localization results have been achieved with BAs [23] and EKFs [15] for a comparable computational burden. The choice is also a matter of compromise: a comparison of monocular SLAM algorithms [27] has shown that, with important resources, BAs seem more accurate. Nevertheless, with limited resources filtering methods seem to be a smarter alternative. With the emergence of cooperative applications, designing a SLAM algorithm with low memory and computational power requirements can be an advantage. Indeed, it implies that sharing the map built by a team of robots is easier thanks to a low bandwidth need (which means more vehicles in the fleet) and that the remaining computational power can be dedicated to other tasks such as fusing distant information. In this paper, we present an EKF-SLAM algorithm.

Monocular SLAM processes imply that special attention is given to the handling of the landmark depth as it cannot be estimated with a single measurement. Indeed, multiple observations, with enough parallax, are needed to refine the position of a point. However, while the depth of a landmark is uncertain, the whole SLAM process is prone to linearization errors. With highly non-linear models, a depth estimate far from the real value means that the linearization process could easily go wrong. This phenomenon is tightly linked to the representation chosen for the landmarks. With a classic Cartesian representation, each step involving a non-linear function (projection of a landmark uncertainty in the image or update of an estimate) can fail. A different representation usually requires to store more parameters, thus adding to the memory load. It is important to bear in mind that the issues raised in this article also affect other variants of Gaussian filtering (such as Information Filters [28]) and that the solutions proposed here could be adapted to other filters.

In this paper, we introduce a linearization-free process that project 3D uncertainties into images and a corrective factor of the Kalman gain that drastically reduces the possibility of linearization failures. These two contibutions allow to achieve similar localization performance to the state-of-the-art approach of Civera et al. [9], which is based on the Inverse Depth parametrization (ID), while having a more computationally efficient and memory thrifty solution due to the use of a simple Cartesian representation (3 parameters) instead of the ID (6 parameters). We conducted an extensive validation of our approach (called MSLAM), with simulations

and experiments, that demonstrate the quality of the computed localization and the efficiency of the proposed contributions. We also evaluated MSLAM over a public data set and provide a comparison with the ID-SLAM algorithm of Civera et al. [9].

Section II will present the literature. Section III will then expose our approach starting with the tracking step and how linearization errors are avoided by computing a proper uncertainty projection (Subsection III-A). Next, Subsection III-B will introduce the corrective factor of the Kalman gain which limits linearization failures during the state update. Section IV will eventually show the trajectories conducted to validate and compare our algorithm. Section V will conclude and give some insights about the future of MSLAM.

II. State of the art

The unknown depth of points in monocular SLAM algorithms requires careful handling of the landmarks in the state vector. The community has thus focused on how to initialize landmarks. Two main solutions can be found in the literature: delayed initializations and undelayed ones. In delayed approaches, the idea is to keep landmarks out of the state estimation process until they are accurate enough [1][11]. This way, linearization problems are avoided inside the filter. Nevertheless, there is also an important loss of information while landmarks are not inserted into the filter. Indeed, bearing information, which helps estimate the heading of the vehicle, does not need accurate landmarks to be properly computed. Without these landmarks in the state vector, the orientation of the robot cannot be properly recovered.

In contrast to these methods, undelayed initializations aim at integrating landmarks as soon as they are observed even if totally inaccurate. Then, their positions are refined inside the estimation process with new observations. Yet, the linearizations involved by the use of non-linear models are likely to fail with estimates far from their true values. To counter this effect, the authors of [18] and [25] create several hypotheses for each landmark with different depths. With new observations, wrong hypotheses are progressively discarded. However, each new hypothesis implies an extra-cost in terms of both memory and processing time. More convenient representations have been imagined to avoid that. The Inverse Depth parametrization (ID) [8][19] is one of the most popular representations for monocular SLAM. It consists of 6 parameters for each landmark: its 3D position (represented by the azimuth and elevation angles of the line-of-sight on which the landmark lies and the inverse of the depth) and 3 more for the position of the vehicle at the moment of the initialization (anchor). These 6 parameters allow to get a representation that is more suited to large uncertainties along an axis. However, this parametrization is not linearization-error-free [4]. Moreover, it means that a single landmark takes twice as much memory in the state vector as does a classic representation. The storage of the covariance matrix is even more costly as it requires four times the size of the same matrix with a Cartesian representation. Landmarks that have converged are often switched from ID to Cartesian representation [7].

We chose to use a simple Cartesian representation so as to avoid the cost of the ID parametrization. This implies that it is necessary to find ways to avoid or reduce linearization errors [26].

III. MSLAM

First off, the initialization of the 3D point and its uncertainty must be made depending on the 2D feature extracted in the image. In our case, we use a Harris detector for distinguishable features. Let us consider a function h_{ci}, which, from a 3D landmark $\mathbf{l}_c = \begin{pmatrix} x & y & z \end{pmatrix}^T$ in the camera frame $\Re_c$, allows to get a point $\mathbf{l}_i = \begin{pmatrix} u & v \end{pmatrix}^T$ in the image plane $\Re_i$:

$$\begin{cases} u = \dfrac{c_u x + f_u y}{x} \\ v = \dfrac{c_v x + f_v z}{x} \end{cases} \tag{1}$$

with f_u and f_v being the focal distances in pixels and c_u and c_v the coordinates of the optical center in the image frame.

Initializing a landmark requires the inversion of this projection function. The goal is to find, from a feature in $\Re_i$ and its associated noise, a 3D position with its corresponding uncertainty. Let us consider an observation in the image (feature) $\mathbf{l}_i$ with its observation noise $\mathbf{P}_{\mathbf{l}_i}$, a diagonal 2×2 matrix composed of σ_u^2 and σ_v^2 which are respectively the variances along the $\vec{u}$ and $\vec{v}$ axes. We are looking for its 3D representation $\mathbf{l}_c$. As h_{ci} is not invertible, a fixed depth $x = x_d$ is used in order to initialize the landmark. x_d can be chosen experimentally depending on the environment. More open areas will require a higher value than urban zones for instance. However, as will be shown later in the experiments section, the value of x_d does not affect much landmark convergence when linearizations are handled properly. The uncertainty will then be built so as to cover the distance between the vehicle and twice the fixed depth used. Based on the pinhole model, we can infer:

$$\begin{cases} x = x_d \\ y = \dfrac{ux - c_u x}{f_u} \\ z = \dfrac{vx - c_v x}{f_v} \end{cases} \tag{2}$$

Computing the associated covariance is more complex and requires the use of the Jacobians associated to the landmark creation. The idea is to find the uncertainty $\mathbf{P}_{\mathbf{l}_c}$ (a diagonal 3×3 matrix composed of σ_x^2, σ_y^2 and σ_z^2 which are respectively the variances along the $\vec{x}$, $\vec{y}$ and $\vec{z}$ axes) whose projection with the Jacobians is coherent with the observation noise defined in the image. $\mathbf{P}_{\mathbf{l}_i}$ can be expressed as follows:

$$\mathbf{P}_{\mathbf{l}_i} = \mathbf{H}_{ci} \mathbf{P}_{\mathbf{l}_c} \mathbf{H}_{ci}^T \tag{3}$$

where $\mathbf{H}_{ci}$ is the Jacobian associated to h_{ci} (pinhole model):

$$\mathbf{H}_{ci} = \begin{bmatrix} -\dfrac{f_u y}{x^2} & \dfrac{f_u}{x} & 0 \\ -\dfrac{f_v z}{x^2} & 0 & \dfrac{f_v}{x} \end{bmatrix} \tag{4}$$

The uncertainty must be computed along the $\vec{x}$ axis (in front of the camera) in order to reduce the system number of variables. The uncertainty will then be rotated back on

the observation line-of-sight. The projection of the landmark defined in Equation (2) on the $\vec{x}$ axis is computed as follows:

$$\begin{pmatrix}\rho & 0 & 0\end{pmatrix}^{\mathrm{T}} = \begin{pmatrix}\sqrt{x^2+y^2+z^2} & 0 & 0\end{pmatrix}^{\mathrm{T}} \quad (5)$$

It is then possible to identify the values σ_y and σ_z. So as to accurately translate the physical reality of the uncertainty in $\Re_c$, we introduce the variable d_{min} which is the minimal distance from which the camera can see. We thus obtain the following system:

$$\begin{cases} \sigma_x &= \rho - d_{min} \\ \sigma_y &= \dfrac{\rho\sigma_u}{f_u} \\ \sigma_z &= \dfrac{\rho\sigma_v}{f_v} \end{cases} \quad (6)$$

This initialization aims at representing the physical reality of the available knowledge about a landmark. It explains why σ_x is chosen so as to cover the distance from d_{min} (a landmark cannot be any closer otherwise it is not visible by the camera) to twice ρ (which strongly depends of x_d, chosen according to the environment). Without any other indication about it, σ_x is so computed to use at best the physical information we have about the environment we are evolving in and the vehicle itself. It will then allow us to compute a tracking window (see Subsection III-A) which properly translates the ellipsoid aspect in the image based on its geometric appearance instead of relying on one computed at a specific confidence interval.

The uncertainty regarding the camera position (and so more globally, the vehicle) is then taken into account when $\mathbf{P}_{\mathbf{l}_c}$ is passed into the world frame $\Re_w$. To do so, we define h_{cw}, a function allowing to pass a landmark $\mathbf{l}_c$ expressed in $\Re_c$ to the world frame ($\mathbf{l}_w$). h_{cw} requires the 6D pose of the vehicle $\mathbf{v}$ (3D position and the 3 associated angles):

$$h_{cw}(\mathbf{l}_c, \mathbf{R}_w, \mathbf{R}_v, \mathbf{t}_w, \mathbf{t}_v) = \mathbf{R}_w(\mathbf{R}_v\mathbf{l}_c + \mathbf{t}_v) + \mathbf{t}_w \quad (7)$$

where $\mathbf{R}_w$ and $\mathbf{R}_v$ are the rotation matrices computed respectively with the vehicle orientation (vehicle frame $\Re_v$ to $\Re_w$) and extrinsic parameters known from calibration ($\Re_c$ to $\Re_v$). Similarly, $\mathbf{t}_w$ and $\mathbf{t}_v$ are the translation vectors established with the vehicle position ($\Re_v$ to $\Re_w$) and extrinsic parameters ($\Re_c$ to $\Re_v$) which are also known from the calibration step. The Jacobian $\mathbf{H}_{cw}$ corresponding to h_{cw} can then be computed to pass $\mathbf{P}_{\mathbf{l}_c}$ to $\Re_w$. For the sake of clarity, let us break $\mathbf{H}_{cw}$ into two parts: $\mathbf{H}_{\mathbf{v}_{cw}}$ (3×6 matrix) and $\mathbf{H}_{\mathbf{l}_{c_{cw}}}$ (3×3 matrix) corresponding respectively to the Jacobians associated to $\mathbf{v}$ and to $\mathbf{l}_c$. $\mathbf{P}_{\mathbf{v}}$ is the uncertainty associated to $\mathbf{v}$. The uncertainty of the landmark in $\Re_w$ ($\mathbf{P}_{\mathbf{l}_w}$) can be computed as follows:

$$\mathbf{P}_{\mathbf{l}_w} = \mathbf{H}_{\mathbf{v}_{cw}}\mathbf{P}_{\mathbf{v}}\mathbf{H}_{\mathbf{v}_{cw}}^T + \mathbf{H}_{\mathbf{l}_{c_{cw}}}\mathbf{P}_{\mathbf{l}_c}\mathbf{H}_{\mathbf{l}_{c_{cw}}}^T \quad (8)$$

As stated before, linearization failures are likely to happen with this initialization as the true distance of the landmark can be far off the estimate. However, the chosen Cartesian representation has the advantage to be minimal.

A. Tracking window

After its initialization, a landmark is tracked through the next images. These new observations help estimate the landmark depth and thus reduce its uncertainty. The tracking part is usually done with Zero mean Normalized Cross-Correlation (ZNCC) [10][12]. When a feature is first selected, a small patch (11×11 pixels in our case) is extracted around it. This descriptor is then tracked in the next images with ZNCC. However, scanning the whole image looking for a feature is an unreliable, time-consuming process. Consequently, authors tend to define a tracking window in the image plane in order to have a smaller area in which to look for matchings.

Two main approaches can be found in the literature when it comes to computing a tracking window. In both methods, the 3D point is projected into the image. In the first one, the size of the tracking window is similarly set for all the features based on the maximum displacement that can occur between two successive frames [20]. It is suitable as long as the robot does not move too fast or the camera speed is sufficiently high. Indeed, the size of the tracking window could be important thus leading to a costly and unreliable tracking process. Moreover, this approach does not take advantage of the landmark uncertainty that could help to reduce the bounding box size.

The other main trend relies on projecting the uncertainty into the image with the Jacobian of the observation model [12]. It allows to have an area in which a correspondence can then be sought. During the initialization, we compute an ellipsoid whose projection by the Jacobians is guaranteed to be the image uncertainty. However, when defining a tracking window, the linearization is accomplished around the landmark and vehicle estimates, which, after the initialization, might have changed. Consequently, the resulting projection can be wrong as we cannot ensure that the projection of the ellipsoid will respect the initial image uncertainty, thus preventing proper working of the tracking process. Moreover, the correspondence between an uncertainty in the image and in the world is difficult to maintain as the projection of an uncertainty from the image to the camera is not an ellipsoid but a cone. With this method, landmarks are usually initialized and lost very quickly, especially in case of high vehicle velocity or low camera frequency where uncertainties quickly grow. Figure 1 shows how the projection by the Jacobians can be restrictive compared to the shape of the ellipsoid. The tracking issue induced by the Jacobians is also depicted in Figure 3 on a real example. In this last example, the landmark covariance has been correctly initialized and its evolution depends on the noise associated to the odometric measurements which has been properly set.

To avoid the linearization problems, we decided to use a geometrical approach. By using planes, tangent to the ellipsoid of uncertainty, MSLAM is able to compute a proper tracking window. Figure 2 covers the whole process with a 2D example. We start from the 3D uncertainty and we search for a 2D ellipse in the image. Let us consider $\mathbf{P}_{\mathbf{l}_c}$ the covariance of a landmark $\mathbf{l}_c$ in $\Re_c$:

$$\mathbf{P}_{\mathbf{l}_c} = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$$

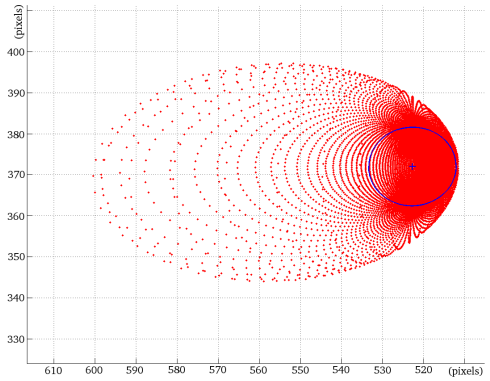


Fig. 1. Simulation example of an ellipsoid projection in an image of resolution 1024×768. The ellipsoid is typical of what is obtained after an initialization and is also slightly off-centered. The blue cross is the projection of a 3D point in the image. The restrictive blue ellipse is the associated uncertainty computed with the Jacobians. The red dots are the projections of the points composing the ellipsoid in $\Re_i$.

We define its inverse as:

$$\mathbf{P}_{\mathbf{l}_c}^{-1} = \frac{1}{\det \mathbf{P}_{\mathbf{l}_c}} \begin{pmatrix} df - e^2 & ce - bf & be - cd \\ ce - bf & af - c^2 & cb - ae \\ be - cd & cb - ae & ad - b^2 \end{pmatrix}$$

$$= \begin{pmatrix} A & B & C \\ B & D & E \\ C & E & F \end{pmatrix}$$

To find the proper ellipse in the image, it is necessary to find the planes tangent to the ellipsoid. We first identify the points on the surface of the ellipsoid $\mathscr{E}$ generated by the covariance $\mathbf{P}_{\mathbf{l}_c}$:

$$\begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}^T \mathbf{P}_{\mathbf{l}_c}^{-1} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} = 1 \tag{9}$$

where $\begin{pmatrix} x_0 & y_0 & z_0 \end{pmatrix}^T$ is the center of $\mathscr{E}$. The planes tangent to the ellipsoid are also orthogonal to the normals of the points on the surface of this ellipsoid. This constraint can be expressed by the gradient:

$$\vec{n} = \nabla \mathscr{E}(x, y, z) = \begin{pmatrix} \frac{\partial \mathscr{E}}{\partial x} & \frac{\partial \mathscr{E}}{\partial y} & \frac{\partial \mathscr{E}}{\partial z} \end{pmatrix}^T$$

$$= \begin{pmatrix} 2(x - x_0)A + 2(y - y_0)B + 2(z - z_0)C \\ 2(x - x_0)B + 2(y - y_0)D + 2(z - z_0)E \\ 2(x - x_0)C + 2(y - y_0)E + 2(z - z_0)F \end{pmatrix} \tag{10}$$

Points that are part of a plane p orthogonal to the normal verify the following relationship:

$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} \in p \Leftrightarrow \begin{pmatrix} x_p - x \\ y_p - y \\ z_p - z \end{pmatrix}^T \cdot \vec{n} = 0 \tag{11}$$

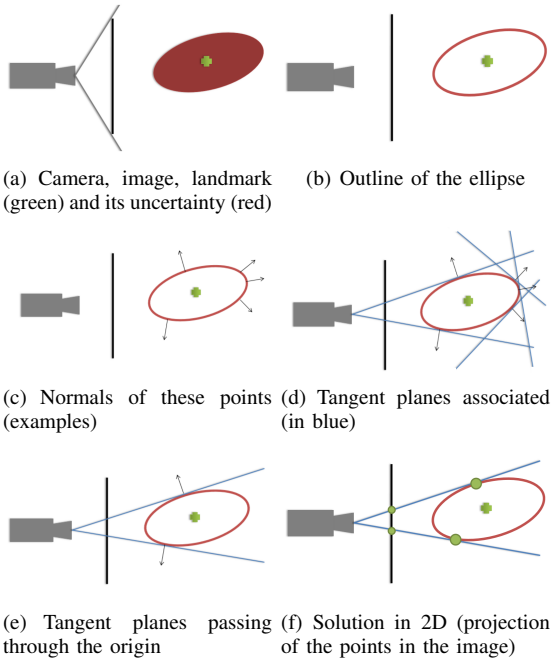
We only keep the points whose tangent planes pass through the origin (camera):

$$\begin{pmatrix} x & y & z \end{pmatrix} \cdot \vec{n} = 0 \tag{12}$$

The points on the surface of the ellipsoid giving these tangent planes can then be projected inside the image to obtain the correct ellipse. However, the uncertainty in the image is defined by an infinite number of points on the ellipsoid. Another constraint must be added to the system. A rectangular window is needed to perform ZNCC. Indeed, it offers better performance than an elliptical window. It allows us to add a constraint about the planes that must be selected. Only four planes are needed for a rectangular window. Furthermore, these planes must intersect the image horizontally or vertically, meaning that they must include the $\vec{y}$ or $\vec{z}$ axis:

$$\vec{y} \cdot \vec{n} = 0 \Leftrightarrow (x - x_0)B + (y - y_0)D + (z - z_0)E = 0 \tag{13}$$

$$\vec{z} \cdot \vec{n} = 0 \Leftrightarrow (x - x_0)C + (y - y_0)E + (z - z_0)F = 0 \tag{14}$$



(a) Camera, image, landmark (green) and its uncertainty (red)

(b) Outline of the ellipse

(c) Normals of these points (examples)

(d) Tangent planes associated (in blue)

(e) Tangent planes passing through the origin

(f) Solution in 2D (projection of the points in the image)

Fig. 2. 2D example of the constraints used to compute a bounding box in the 3D case.

It gives us the following system to solve for the points tangent to horizontal planes:

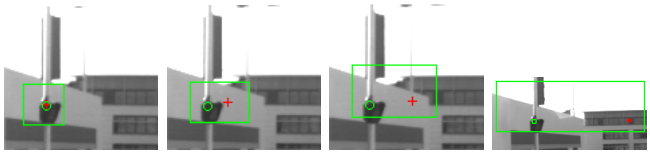
$$\begin{cases} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}^T \mathbf{P}_{\mathbf{l}_c}^{-1} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} = 1 \\ \begin{pmatrix} x & y & z \end{pmatrix} \cdot \vec{n} = 0 \\ \vec{y} \cdot \vec{n} = 0 \end{cases} \tag{15}$$

And the following one for the points on the vertical planes:

$$\begin{cases} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}^T \mathbf{P}_{\mathbf{l}_c}^{-1} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix} = 1 \\ \begin{pmatrix} x & y & z \end{pmatrix} \cdot \vec{n} = 0 \\ \vec{z} \cdot \vec{n} = 0 \end{cases} \tag{16}$$

The resolution of these systems is straightforward and will not be detailed here for lack of space. After resolution, each system provides a couple of points $\begin{pmatrix} x & y & z \end{pmatrix}^T$ which, once projected inside the image, allows the computation of a proper bounding box. Thanks to this method, linearization errors are avoided. More details can be found about the bounding box method in [5]. The results of a tracking example can be observed in Figure 3.

(a) Frame 1: tracking okay (b) Frame 2: tracking okay (c) Frame 3: tracking failure (d) Frame 7: tracking failure



(e) Frame 1: tracking okay (f) Frame 3: tracking okay (g) Frame 7: tracking okay (h) After 18 frames

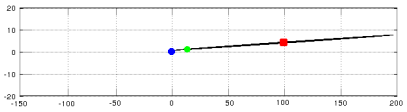
Fig. 3. (a)-(d): tracking failure due to significant linearization errors (projection through the Jacobians). (e)-(h): tracking with the method proposed in this paper. In both cases, the green rectangle is the bounding box of the covariance projection in the image (area where the point is supposed to be). The red cross is the estimate of the landmark position. The green circle is the observation. In this example, the vehicle is moving forward while trying to track a landmark in the images.

By improving the tracking, we are able to enhance two major aspects of our SLAM algorithm. Indeed, the most demanding step in terms of processing time is feature extraction. As fewer landmarks are initialized, our algorithm becomes much faster and requires less memory. The other consequence is that more landmarks are accurate and thus kept in the state vector. It means that the vehicle pose can reach a better accuracy. A validation of the tracking window is proposed in Section IV.

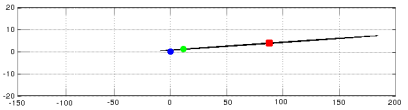
B. Corrective factor of the update step

After the tracking step, it is necessary to update the state vector in order to take advantage of the new observation. However, the update stage relies on the Kalman gain which itself depends on the linearization of the observation function. As stated before, the linearization process can go wrong because it is made around the fictitious 3D point which can be located far from its true position. The consequence of this wrong update is that the landmark can pass behind the camera while it has just been observed. An example of failure during the update is shown in Figure 4. The data set used is the same as the one in Figure 3. The point is initialized at 100 meters. After the second update, the landmark is estimated behind the observer which is impossible.

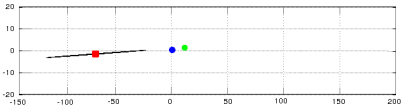
This problem has already been noticed in [21] and is also affecting the Inverse Depth parametrization. However, the solution proposed by the authors simply consists in discarding



(a) Initialization of the point at 100 meters on the line-of-sight



(b) First update: integrity is preserved



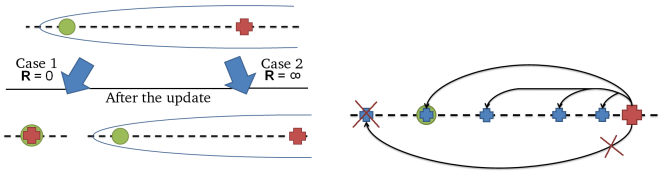
(c) Second update: landmark updated behind the observer

Fig. 4. Update failure due to linearization errors. Top view of a point updated. The blue circle is the position of the vehicle. The red square is the landmark. The green circle is the real position of the landmark. The black ellipse is the uncertainty associated to the landmark after its initialization.

an update if the landmark new position is behind the camera. Some authors rely on a fast camera in order to have several small updates instead of a big one [11]. It has the advantage to avoid most of the linearization errors. However, it only works if the camera frequency is high and the vehicle speed low.

The perturbations generated by a non-linear observation function can be detected as long as this function can be expressed as a ratio between two linear functions [15]. It is the case of the function allowing to project landmarks in $\Re_i$. As soon as a linearization error is detected, a corrective can be applied so as to reduce the linearization influence. This corrective is used as a multiplicative factor for the Kalman gain. Indeed, the Kalman gain quantifies the impact that an observation will have on the current estimate. By modulating it, it becomes possible to ponder its effect during the update step of the Kalman filter, thus reducing linearization errors. To know when the update must be corrected, it is necessary to understand when a linearization has failed. To do so, the results expected from an update must be studied. We have already stated that linearization issues will appear more frequently shortly after the initialization when the uncertainty is important. Let us consider a landmark estimate with a big uncertainty and an observation of this estimate. We analyze two cases: an observation with no uncertainty (perfect observation) and the same observation but with an infinite uncertainty (see Fig. 5(a)). With a perfect observation, the updated landmark should have converged on the observation. On the other hand, an observation with infinite uncertainty is useless and should not improve the landmark uncertainty or its position. As a consequence, we can determine a range in the image where the updated landmark should be located. Without any new information (observation with infinite uncertainty), the projection of the updated landmark must be the same as before. With the best possible correction (perfect observation), the projection of the updated landmark must be on the observation. Considering these extreme cases, if the projection of

the updated landmark is not between the landmark projection before the update and the observation, a linearization error has occurred (see Fig. 5(b)).



(a) Extreme cases for the update in the image frame: perfect observation and infinite uncertainty observation.

(b) Update range for the landmark projection.

Fig. 5. Update of a landmark. The red cross is the projection of the landmark estimate. The green circle is the observation. The blue crosses represent the updated projection of the landmark estimate. $\mathbf{R}$ is the observation noise (uncertainty).

When the failure is noticed, the role of the corrective factor is to bring back the updated projection of the landmark on the observation. Indeed, the observation noise is very small compared to the landmark uncertainty. As a consequence, we can rely on the observation when a linearization error occurs. It can be mathematically expressed by the following relationship:

$$\mathbf{z}_k = h(\mathbf{x}_{k|k}) \tag{17}$$

where $\mathbf{z}_k = \begin{pmatrix} z_{u_k} & z_{v_k} \end{pmatrix}^T$ is an observation in the image frame, h is the observation function and $\mathbf{x}_{k|k}$ is the estimated state after the update.

So as to make this equation true, we define $\mathbf{\Omega}_k$ the Kalman gain corrected by the factor r:

$$\mathbf{\Omega}_k = r \cdot \mathbf{K}_k \tag{18}$$

where $\mathbf{K}_k$ is the Kalman gain.

It is important to observe that this corrective does not change the ratio between the uncertainties from the observation and from the projection of the estimate into the image:

$$\begin{aligned} \mathbf{\Omega}_k &= r \cdot \mathbf{P}_k \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k \right)^{-1} \\ &= \mathbf{P}_k \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{H}_k \frac{\mathbf{P}_k}{r} \mathbf{H}_k^{\mathrm{T}} + \frac{\mathbf{R}_k}{r} \right)^{-1} \end{aligned} \tag{19}$$

where $\mathbf{P}_k$ is the uncertainty estimate, $\mathbf{H}_k$ the Jacobian associated to h and $\mathbf{R}_k$ the observation uncertainty.

Both uncertainties are similarly affected by r. Moreover, it is possible to deduce the range of values that r can take. Indeed, $\mathbf{R}_k$ is the best a priori available on the observation noise. It means that it could not be lower than the value chosen. Therefore, after applying r, $\mathbf{R}_k$ cannot be smaller than it was, so r cannot be greater than 1. A correction greater than 1 would mean that the projection of the updated landmark is already in the authorized range and so that the correction should not be considered. The impact of r on $\mathbf{P}_k$ allows to infer that r must be greater than 0. $\mathbf{P}_k$ is a covariance matrix and so is positive-semidefinite by nature. A negative value for r would change that. As a consequence, the corrective factor r must only be applied if between the range $(0\ 1]$.

The projection of the landmark estimate $\begin{pmatrix} u_{k-1} & v_{k-1} \end{pmatrix}^T$ is defined through the observation function h as follows (for a 3D landmark $\mathbf{l}_w$ in the world frame $\Re_w$):

$$\begin{cases} u_{k-1} = \dfrac{\mathbf{F}_1 \mathbf{R}_{cw}^T (\mathbf{l}_w - \mathbf{t}_{cw})}{\mathbf{F}_3 \mathbf{R}_{cw}^T (\mathbf{l}_w - \mathbf{t}_{cw})} \\ v_{k-1} = \dfrac{\mathbf{F}_2 \mathbf{R}_{cw}^T (\mathbf{l}_w - \mathbf{t}_{cw})}{\mathbf{F}_3 \mathbf{R}_{cw}^T (\mathbf{l}_w - \mathbf{t}_{cw})} \end{cases} \tag{20}$$

with $\mathbf{F}_i$ being the i^{th} line of the intrinsic parameters matrix and $\mathbf{R}_{cw}$ the rotation matrix allowing to pass landmarks from $\Re_c$ to $\Re_w$ ($\mathbf{t}_{cw}$ is the associated translation).

Considering the Kalman state update equation and the observation function (20), expressing the observation as the updated landmark projection can be done as follows:

$$\begin{cases} z_{u_k} = \dfrac{\mathbf{F}_1 \mathbf{R}_{cw}^T (\mathbf{l}_w + \mathbf{\Omega}_k \mathbf{\Delta}_k - \mathbf{t}_{cw})}{\mathbf{F}_3 \mathbf{R}_{cw}^T (\mathbf{l}_w + \mathbf{\Omega}_k \mathbf{\Delta}_k - \mathbf{t}_{cw})} \\ z_{v_k} = \dfrac{\mathbf{F}_2 \mathbf{R}_{cw}^T (\mathbf{l}_w + \mathbf{\Omega}_k \mathbf{\Delta}_k - \mathbf{t}_{cw})}{\mathbf{F}_3 \mathbf{R}_{cw}^T (\mathbf{l}_w + \mathbf{\Omega}_k \mathbf{\Delta}_k - \mathbf{t}_{cw})} \end{cases} \tag{21}$$

where $\mathbf{\Delta}_k$ is the innovation defined as: $\mathbf{\Delta}_k = \mathbf{z}_k - h(\mathbf{x}_{k|k-1})$.

From there, we can extract the corrective factor by getting $\mathbf{\Omega}_k$ out of the equation:

$$\begin{cases} r_u = \dfrac{(z_{u_k} - u_{k-1}) \mathbf{F}_3 \mathbf{R}_{cw}^T (\mathbf{l}_w - \mathbf{t}_{cw})}{(\mathbf{F}_1 - z_{u_k} \mathbf{F}_3) \mathbf{R}_{cw}^T \mathbf{K}_k \mathbf{\Delta}_k} \\ r_v = \dfrac{(z_{v_k} - v_{k-1}) \mathbf{F}_3 \mathbf{R}_{cw}^T (\mathbf{l}_w - \mathbf{t}_{cw})}{(\mathbf{F}_2 - z_{v_k} \mathbf{F}_3) \mathbf{R}_{cw}^T \mathbf{K}_k \mathbf{\Delta}_k} \end{cases} \tag{22}$$

The lowest corrective factor between r_u and r_v will be kept to avoid producing overconfident estimates. With this factor, linearization failures will be avoided most of the time. This can be observed through the same example as Figure 4 in Figure 6 where the landmark position now properly converges.

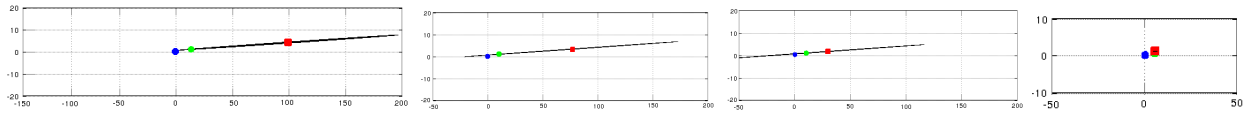
The efficiency of this corrective factor will be exposed in the next section.

IV. Experiments

We will first demonstrate the benefits of each new aspect of our method (Subsection IV-A). Then, other experiments will illustrate the efficiency of the whole algorithm (Subsections IV-B and IV-C). We also compared MSLAM to the state-of-the-art monocular SLAM algorithm of Civera et al. [9] over a public data set so as to highlight the advantages of our method (Subsection IV-D).

A. Individual validations

First off, the impact of x_d on the number of landmarks tracked and kept (accurate landmarks whose uncertainty falls below a threshold, fixed here at 50 centimeters accumulated on the 3 axes) will be demonstrated in the following experiment. A vehicle performed a 30-meter trajectory in a static urban environment. The vehicle was moving at approximately 1 $m.s^{-1}$ and the camera was running at 10 Hz. In each image, the algorithm was tracking 10 landmarks (if fewer are visible, new ones are initialized). Results are visible in Figure 7 with different initialization distances. It can be seen that the number



(a) Initialization of the point at 100 meters on the line-of-sight (b) First update: integrity is preserved (c) Second update: integrity still preserved (d) After 10 updates, the landmark has converged

Fig. 6. Proper update thanks to our corrective factor. Top view of a point updated. The colors are the same as in Fig. 4.

of landmarks initialized is approximately constant (between 125 and 150) with a similar convergence rate (one out of two landmark converges) whatever the distance used for the initialization.

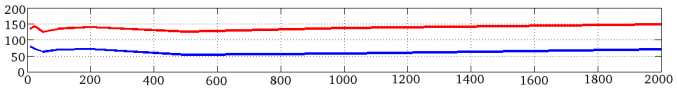


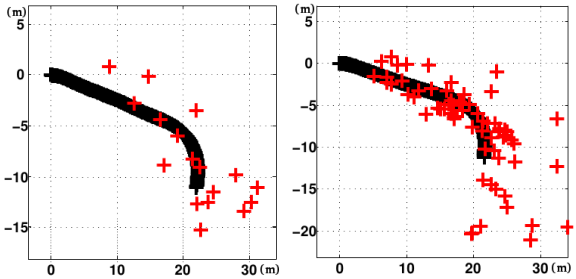
Fig. 7. Number of landmarks initialized (in red) and kept (in blue) according the initialization distance x_d in meters.

In order to illustrate the benefits of this tracking window, we conducted an experiment based on the same data set as previously and with the same conditions. The trajectory was run twice. The first time, an EKF-SLAM using the Jacobians method for the tracking process was used. The second time, the same algorithm was applied, this time using our new geometrical approach. Both algorithms are identical except for this very specific aspect that is the computation of the tracking window. The results are summarized in Table I. We can notice that far fewer landmarks are initialized with our method. Indeed, as our algorithm is able to track them during longer periods of time with the proper bounding boxes (1.96 s in the mean, approximately 20 frames, vs. 1.53 s with the Jacobians, around 15 images, almost a 30% increase), there is no need to initialize new ones. The convergence rate is also in favor of the geometrical approach (27% vs. less than 20% for the Jacobians method). It is necessary to keep in mind that, in both algorithms, the linearization issues coming from the update step are not corrected, making the convergence of landmarks more difficult to achieve.

	EKF-SLAM using Jacobians for tracking	EKF-SLAM with our geometrical approach
Landmarks initialized	249	199
Landmarks conserved	49	53
Mean tracking time	1.53 s	1.96 s
Max. tracking time	17.6 s	22.9 s

TABLE I
BENEFITS OF THE TRACKING PROCESS.

Finally, to show the efficiency of the corrective factor, we used the same 30-meter trajectory as before. During the first execution of the trajectory, a monocular SLAM without the proposed correction was applied. The second time, MSLAM (with the corrective factor proposed here) was used. The algorithms are, once more, identical except for the corrective factor. The trajectory, and the landmarks which have converged in both cases, can be seen in Figure 8 while Table II provides more detailed results. We can notice the same effect as in the bounding box experiment: fewer initializations are made while more landmarks have converged.



(a) Trajectory performed without the corrected Kalman gain. (b) Trajectory performed with the corrected Kalman gain.

Fig. 8. 30-meter trajectory illustrating landmarks convergence with and without the corrected Kalman gain. The trajectory is in black. The red crosses are the conserved landmarks.

	EKF-SLAM without corrective factor	MSLAM (our approach)
Landmarks initialized	384	134
Landmarks conserved	32	75
Divergences	439	0

TABLE II
BENEFITS OF THE KALMAN GAIN CORRECTIVE FACTOR.

Each time a landmark is updated out of the previously defined proper bounds, we cancel the update and keep the landmark in the state vector. It explains why the number of divergences given in Table II is greater than the initializations as a landmark position can diverge several times in different updates. Of course, initializing fewer landmarks allows the algorithm to be faster.

B. SLAM with a low frequency camera

The two experiments that will be presented here share the same trajectory (of approximately 170 meters). The vehicle used, an electrical vehicle called VipaLAB (see Fig. 9), was equipped with a single camera, an odometer (feeding a kinematic model) and a RTK GPS (ground truth for comparison purposes) and was driven manually at approximately 2 $m.s^{-1}$ throughout the trajectory. This speed corresponds to the platooning applications aimed where vehicles are used as proximity transportation systems in city centers. However, the methods proposed here could be applied to faster vehicle as long as they use a higher camera frequency than here. Indeed, the problem is similar between high speed and low frequency: successive observations of landmarks are more difficult to find due to the great displacement occurring between two images.

Fig. 9. VipaLAB: electrical vehicle used in the experiments here. It is equipped with a Marlin F-1318 camera (circled in green) and an odometer.

These experiments took place in an urban platform called PAVIN which is composed of crosswalks, curbs, roads and roundabouts. Several pictures, illustrating the conditions of the experiments as well as the platform, can be seen in Figure 10.

The aim of these experiments is to test the robustness of the proposed solution by using a slow-running camera. With a low camera speed, linearization failures are more likely to happen as each new observation will be located far from the previous one. It means that the bounding box needs to be properly computed in order to be sure that the matching will be correct. Moreover, it also means that the Kalman update will be more subject to linearization errors due to the gap between the observations and the estimates. In the first experiment, the camera was acquiring images at 3.75 Hz and only 10 landmarks per image were used (with new ones initialized when less than 5 are visible). In all the experiments of this section, landmarks were initialized at 200 meters on the line-of-sight of their first observation.

In each of these experiments, another implementation of MSLAM, without the corrected Kalman gain and with the projection through the Jacobians for the tracking process, was used in order to compare with MSLAM. Both algorithms are identical except for the two aspects mentioned. It means that the same constraints are applied: the bounding boxes computed have a minimum and maximum size. These constraints limit the impact of linearization errors on the vehicle pose. The goal is to show the efficiency of the proposed solutions to counter linearization errors, in terms of resources and quality of the computed localization. The results of the first experiment are visible in Figure 11. It is important to note that loop closures are not applied here and in any other experiment.

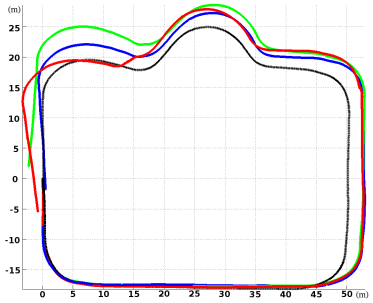


Fig. 11. Localization of the vehicle with a camera running at 3.75 Hz. **Black:** ground truth (RTK GPS). **Green:** odometry-only trajectory. **Red:** EKF-SLAM without the proposed corrections. **Blue:** MSLAM.

The first thing to notice here is that the localization given by MSLAM is better than the one based on an EKF-SLAM without the correctives. In this example, the divergence is not very important and this is mostly due to the fact that our kinematic model is already quite good and prevents important errors. Moreover, the controls mentioned before avoid most of the linearization errors. Still, it can be seen that near the end of the trajectory, the orientation of the vehicle is not well estimated which would certainly lead to a more important divergence if the trajectory was not finished.

As for the previous experiments, Table III indicates the number of landmarks initialized, having converged and the mean tracking time.

	SLAM without the corrections	MSLAM
Landmarks initialized	472	408
Landmarks conserved	267	354
Mean tracking time	2.8 s	3.8 s

TABLE III
BENEFITS OF MSLAM WITH A REAL TRAJECTORY.

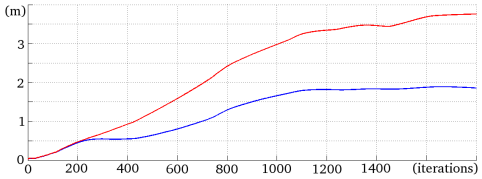
Once again, more landmarks have been initialized by the SLAM not using the corrections introduced in this paper as it is not able to track points during long periods of time (2.8 s in the mean vs. 3.8 s with MSLAM) . Similarly, a higher convergence rate is achieved with MSLAM as most linearization errors are avoided. Even though the frequency of the camera is low, many landmarks have been kept with MSLAM. It can be explained by the fact that with fewer images, the updates are more important and make the landmarks converge faster. Without a proper bounding box or Kalman gain, it is difficult to track and update the landmarks without risking linearization failures. An interesting point here is the size of the map at the end of the trajectory. Our goal is to provide a solution with low memory requirements. Here, with 354 landmarks for 170 meters, less than 34 KB are needed to keep the whole map (state vector and covariance matrix). With an Inverse Depth parametrization, the same map would have required 119 KB. Over long distances, a lighter map is an important asset. Concerning the computational time required, MSLAM was running in real time and took on average 15 ms per image of resolution 1024×768 pixels. The two corrections proposed in this article (computation of the bounding box and corrective factor of the Kalman gain) do not slow down the process as they only require around 1 μs each.

The quality of the computed localization is given, for both MSLAM and the SLAM without corrections, in Figures 12(a) and 12(b) with the Root-Mean-Square Error (RMSE) and the Consistency Index (CI) as defined in [22] regarding this trajectory. This last quality index is based on the Normalized Estimation Error Squared (NEES) and the Chi-square test (the desired significance level is set here to 0.05). When CI is lower than 1, the estimation is consistent with the ground truth and when the CI is greater than 1, the estimated pose is optimistic (inconsistent) thus measuring the accuracy of the estimated covariance.

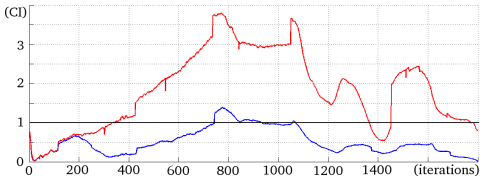
MSLAM is able to achieve a smaller RMSE in comparison with a classic SLAM. Also, MSLAM provides consistent vehicle poses most of the time as opposed to the SLAM

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

Fig. 10. Some examples of typical camera outputs in PAVIN. Some pictures show the important reflections caused by the sun during these experiments.



(a) Root-Mean-Square Error



(b) Consistency Index

Fig. 12. In red are the results of the SLAM without correctives and in blue, those of MSLAM for a camera running at 3.75 Hz.

without correctives. However, it is worth noting that SLAM algorithms tend to drift over time because of non-linear models [3][16] thus requiring special means to counter this effect, such as loop closing which is not performed here.

For the second experiment, we used our previous trajectory and slowed down the camera to less than 2 Hz. With such a setting, the linearization problems are intensified. Figure 13 shows the trajectories performed by both MSLAM and the SLAM without the corrections introduced in this article.

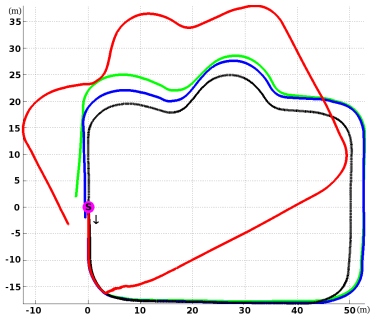
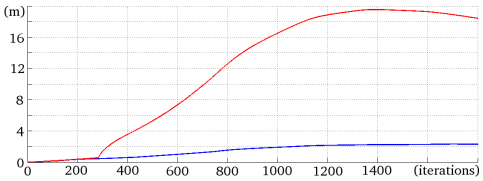


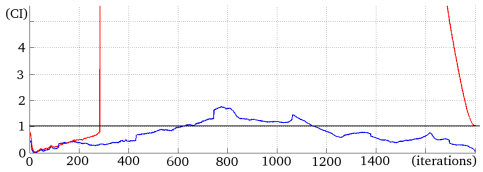
Fig. 13. Localization of the vehicle with a camera running at less than 2 Hz. **Black:** ground truth (RTK GPS). **Green:** odometry-only trajectory. **Red:** EKF-SLAM without the proposed corrections. **Blue:** MSLAM.

The trajectory calculated by MSLAM is very close to the one in Figure 11 which confirms the robustness of our approach. Indeed, even with fewer images, it is able to achieve similar localization results. On the other hand, the approach which does not use the solutions proposed here is affected by this low frequency and quickly diverges. Right after the first bend, the orientation changes a lot and a jump in the position can also be noticed. These are the consequences of linearization failures. However, after the first bend most of the linearization errors are avoided as no sudden hops can be seen in the trajectory. The different controls added to the algorithm helped a lot in preventing linearization errors from being too important.

As for the previous experiment, Figures 14(a) and 14(b) present the RMSE and the CI for this trajectory.



(a) Root-Mean-Square Error



(b) Consistency Index

Fig. 14. Same color code as previoulsy. This time the camera is running at less than 2 Hz.

The effect observed before is amplified here by linearization errors. MSLAM performs similarly despite the lower speed whereas the SLAM without the correctives has a much higher RMSE. The CI of MSLAM is almost always lower than 1 meaning that consistency is ensured most of the time. On the other hand, the SLAM without correctives has a CI which goes above 100 (not displayed here for the sake of clarity).

C. Localization using a map built by MSLAM

To demonstrate the quality of the maps built, we conducted another trajectory in which the map of a vehicle was given to another one. The scenario is simple: a first vehicle builds a reference map which is then used by a second vehicle. The task of the second robot is to localize itself inside this map while following a trajectory similar to the first one. The second algorithm does not map more landmarks but only uses the ones in the reference map in order to localize itself. The principle

is the same: landmarks are projected in the image where observations are sought with ZNCC. We used a simulator in order to perform similar trajectories more easily. The simulator presents realistic physics as well as environments mapped from real locations. It must be noted that the simulator is only used to generate sensor data. Indeed, our algorithm is running on a separate computer as would be the case with a real vehicle. For this trajectory, the environment is an urban city center (place de Jaude). All the typical urban features can be found: roads, building, curbs, crosswalks and so on. Several pictures of the environment along with some typical camera outputs from the simulator are exposed in Figure 15.

By measuring the true gap between the paths followed by both vehicles (with RTK GPS's for instance) and comparing it to the gap obtained with MSLAM, we are able to have a clear indicator of the quality of the localization. Figure 16 shows the two trajectories performed and their respective ground truth. Both trajectories were approximately 170-meter long.

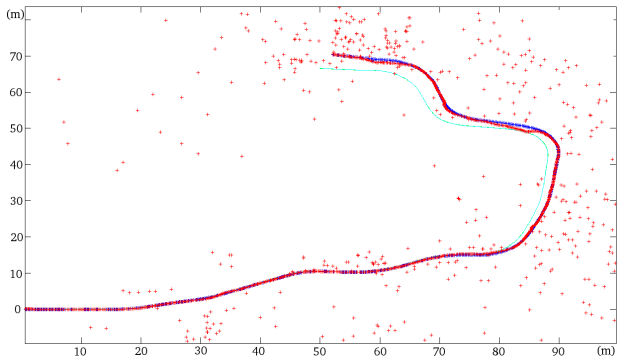


Fig. 16. Localization of the vehicles. **Green:** ground truth for the first vehicle. **Cyan:** ground truth for the second vehicle. The two trajectories are superimposed because very similar, even though not identical. **Blue:** MSLAM for the first vehicle. **Red:** MSLAM for the second vehicle using the map built by the first. **Red crosses:** landmarks mapped during the first passage and tracked in the second.

We can see that the localizations computed by our algorithm are very close to the ground truth. More landmarks have been initialized in order to have more points in the reference map and thus ease the localization in this map for the second vehicle. Here, around 600 landmarks have been mapped, which remains quite a low quantity to handle. Figure 16 also shows that the trajectories computed by MSLAM are rather close together, only diverging a little bit near the end. A closer look is given at the localization error in Figure 17.

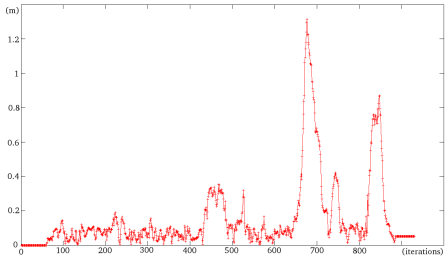


Fig. 17. Localization error when using the reference map built by the first vehicle (difference between RTK GPS positions and MSLAM ones).

The error is on average around 10 centimeters which is suitable for automatic driving. However, some important peaks can be noticed near the end of the trajectory. It can be explained by the fact that fewer landmarks are visible at this point of the trajectory (in bends). It consequently makes localization using the reference map much more difficult. Mapping more landmarks in bends could help to reduce this problem.

These results show that MSLAM can be used in cooperative applications where the vehicles share their maps. Indeed, thanks to the use of a classic Cartesian representation, maps are light and so network requirements are low. The results here illustrate that good localization results can still be achieved.

D. Comparison with a state-of-the-art SLAM algorithm

The last experiment exposed in this paper aims at comparing MSLAM with the state-of-the-art approach of Civera et al. [9] over a public data set. In [9], a monocular EKF-SLAM using the Inverse Depth parametrization and an odometer is used. This representation makes it possible to avoid most linearization errors. Furthermore, a complex data association method is applied: 1-point RANSAC. This algorithm guides the data association process by constraining the filter to perform the first update with the observation allowing to keep most associations after the fact. The data set used was the one applied by Civera et al. and comes from the RAWSEEDS public database (www.rawseeds.org).

The trajectory was recorded on Milan's campus and is a mix of classic urban environment and more open areas with trees, grass or gravel. The small robot used (Robocom) evolves at a low speed (less than 1 $m.s^{-1}$) and is equipped with an odometer and a camera furnishing images of resolution 320×240 at 30 Hz. The trajectory is about 1360-meter long. Figure 18 shows the estimated path of the robot when only using proprioceptive sensors.

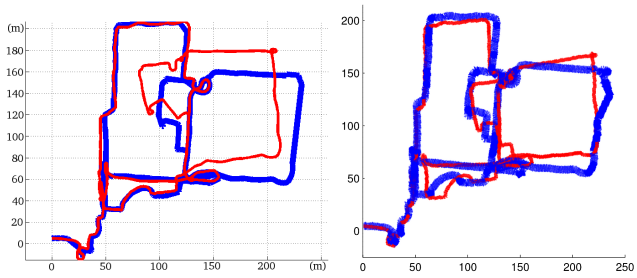
Fig. 18. In green, the trajectory computed using only odometric sensors. The blue trajectory is one given by the GPS and used as a ground truth.

We can notice that there is an important odometric drift. It is also important to note that these results are identical to the ones obtained by Civera et al. and so that only the implication of the vision algorithm will be measured. Figure 19 exposes the trajectory computed by our algorithm next to the one of Civera et al. published in [9].

We can see that most of the odometric drift has been corrected. A part of the angular error could not be corrected in the second half of the trajectory, thus creating this gap with

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

Fig. 15. Some examples of pictures taken from the simulator. (a)-(b): overview of the environment used for the simulated scenarios. (c)-(d): some typical camera outputs taken with the simulated camera.



(a) Localization results of MSLAM (b) Localization results from [9]

Fig. 19. Localization results of monocular SLAM algorithms. In both cases, the blue curve is the ground truth and the red one the trajectory computed by the SLAM algorithm.

the GPS. The measure of the gap between the GPS and the position computed by MSLAM gives a good indication of the localization quality. On the whole trajectory, an average 0.84% drift has been measured. The approach of Civera et al. is able to reach a 0.7% drift. The results are thus close. The use of a more efficient data association algorithm (like 1-point RANSAC) in MSLAM could further improve our results. Figure 20 shows the localization error throughout the trajectory (difference between MSLAM vehicle pose and the RTK GPS).

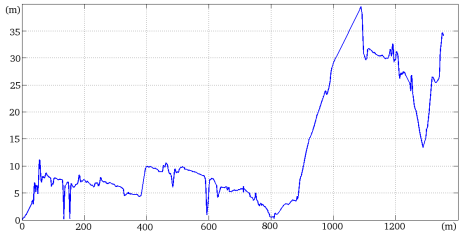


Fig. 20. Distance between the computed position and the ground truth according to the distance traveled.

Concerning the processing time, we measured the time required per image. As for the comparison to be fair, both algorithms were running on hardware equipped with the same processor: a i7 at 2.67 GHz. MSLAM took in the mean 7 ms per image (ranging from 3 ms to 13 ms) whereas the one of Civera et al. is closer to 18 ms per image (ranging from 14 ms to more than 40 ms). Similarly, both algorithms tracked 25 landmarks per image but the ID parametrization requires twice as much memory to store the state vector and four times as much for the covariance matrix compared to MSLAM.

V. Conclusion

An EKF-SLAM solution (MSLAM) relying only on a camera and an odometer has been presented. A minimal Cartesian representation is used, allowing to have a memory-thrifty algorithm thus making MSLAM suited for all kinds of cooperative approaches. Linearization issues, involved by the light Cartesian representation that we have chosen, are corrected. During the tracking step, a new method to compute a proper bounding box has been introduced and validated. A corrective factor in the update step, reducing the impact of linearization problems on the Extended Kalman filter, has been presented and tested. The benefits of MSLAM over a classic approach have been extensively evaluated with several experiments.

Satisfying localization results are achieved within a small computing time. MSLAM was tested with several trajectories under different conditions and camera frequencies. The localization inside a reference map built by our algorithm has also been presented. It shows that the maps produced by MSLAM are suitable for automatic driving scenarios as the localization accuracy obtained is sufficient. Finally, a comparison with the state-of-the-art approach of Civera et al. has been made. It illustrates that MSLAM behaves similarly while being faster and needing less memory.

However, data association should be investigated to reinforce our algorithm. It would avoid losing track of a landmark when its aspect undergoes too much changes (illumination or slow running camera). Another interesting perspective is to develop new multi-vehicle approaches using MSLAM favorable characteristics.

References

[1] T. Bailey. Constrained Initialisation for Bearing-Only SLAM. In *ICRA*, volume 2, pages 1966–1971, Taipei, China, 2003.

[2] T. Bailey and H. Durrant-Whyte. Simultaneous Localization and Mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117, 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 *ICRA*, pages 1937–1943, Orlando, USA, 2006.

[5] G. Bresson, T. Féraud, R. Aufrère, P. Checchin, and R. Chapuis. Parsimonious Real Time Monocular SLAM. In *IV*, pages 511–516, Alcal de Henares, Spain, 2012.

[6] D. Checklov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-Time and Robust Monocular SLAM using Predictive Multi-Resolution Descriptors. *Advances in Visual Computing*, 4292:276–285, 2006.

[7] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse Depth to Depth Conversion for Monocular SLAM. In *ICRA*, pages 2778–2783, Rome, Italy, 2007.

[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 *RSS*, Atlanta, USA, 2007.

[11] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *IEEE International Conference on Computer Vision*, pages 1403–1410, Nice, France, 2003.

[12] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

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

[14] E. Eade and T. Drummond. Scalable Monocular SLAM. In *CVPR*, volume 1, pages 469–476, New York, USA, 2006.

[15] T. Féraud, R. Chapuis, R. Aufrère, and P. Checchin. Improving Results of Non-Linear Observation Function Using a Kalman Filter Correction. In *International Conference on Information Fusion*, Chicago, USA, 2011.

[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] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, Santa Barbara, USA, 2007.

[18] N. M. Kwok and G. Dissanayake. An Efficient Multiple Hypothesis Filter for Bearing-Only SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 736–741, Sendai, Japan, 2004.

[19] J. Montiel, J. Civera, and A. J. Davison. Unified Inverse Depth Parametrization for Monocular SLAM. In *RSS*, Philadelphia, USA, 2006.

[20] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real Time Localization and 3D Reconstruction. In *CVPR*, pages 363–370, New York, USA, 2006.

[21] M. P. Parsley and S. J. Julier. Avoiding Negative Depth in Inverse Depth Bearing-Only SLAM. In *IROS*, pages 2066–2071, Nice, France, 2008.

[22] L. M. Paz, J. D. Tardós, and J. Neira. Divide and Conquer: EKF SLAM in O(n). *IEEE Transactions on Robotics*, 24(5):1107–1120, 2008.

[23] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau. Localization in Urban Environments: Monocular Vision Compared to a Differential GPS Sensor. In *CVPR*, pages 114–121, San Diego, USA, 2005.

[24] R. Smith, M. Self, and P. Cheeseman. A Stochastic Map for Uncertain Spatial Relationships. In *4th International Symposium on Robotics Research*, pages 467–474, 1988.

[25] J. Solà, A. Monin, M. Devy, and T. Lemaire. Undelayed Initialization in Bearing Only SLAM. In *IROS*, pages 2499–2504, Edmonton, Canada, 2005.

[26] J. Solà, T. A. Vidal-Calleja, J. Civera, and J. Montiel. Impact of landmark parametrization on monocular EKF-SLAM with points and lines. *International Journal of Computer Vision*, 97(3):339–368, 2012.

[27] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: Why Filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[28] M. R. Walter, R. M. Eustice, and J. J. Leonard. Exactly Sparse Extended Information Filters for Feature-Based SLAM. *The International Journal of Robotics Research*, 26(4):335–359, 2007.

Guillaume Bresson received his PhD in mobile robotics from Université Blaise Pascal (UBP), France in 2014. He received his Dipl. Ing. degree in computer science in 2010 from ISIMA, UBP, France and his MSc degree in robotics from UBP, France, in 2010. He is currently a postdoctoral fellow at Inria Paris-Rocquencourt in the RITS team. His current research interests revolve around Intelligent Transportation Systems, ranging from localization and perception for a single vehicle to the multi-vehicle problematic.

Thomas Féraud received his engineer degree in electronics, control, signal processing and embedded systems from Polytech'Clermont-Ferrand, France, in 2008, his MSc. degree in Robotic Vision from the Université Blaise Pascal, Clermont-Ferrand, France, in 2008, and his Ph.D. degree in Mobile Robotics from the Université Blaise Pascal, Clermont-Ferrand, France, in 2011, where he was hosted by the LASMEA, CNRS. He was a Postdoctoral Fellow at Kumamoto University, Japan. He is currently at Institut Pascal, CNRS, where he is working on visual localization and mapping. His current research interests include estimation and data fusion applied to autonomous vehicle, mainly on non-linear processes as visual odometry or Extended Kalman Filter.

Romuald Aufrère is an Associate Professor in the Pascal Insitute (Ex. Lasmea) and LIMOS Laboratories of the Blaise Pascal University (UBP), France. He obtained his Ph.D. degree in Computer Vision in 2001. In 2002, he carried out a Postdoctoral training in the Robotics Institute of Carnegie Mellon University (CMU), USA. Since september 2003, he teaches Automatics, Robotics and Signal Processing in an engineering school of computer science (ISIMA). His research interests are in intelligent Vehicles. He is currently working on monocular and decentralized SLAM, multi-vehicle localization and active perception.

Paul Checchin is Associate Professor (HDR) in the Department of Electrical Engineering and Industrial Data Processing at Blaise Pascal University. He is a member of the ISPR (ImageS, Perception systems and Robotics) department carrying on research in Computer Vision and Robotics, within the Pascal Institute (UMR6602 CNRS/UBP), a joint research unit of CNRS (The National Center for Scientific Research) and Blaise Pascal University. His main research interests focus on robot autonomy, localization, mapping, scene understanding and perception applied to Simultaneous Localization And Mapping (SLAM) of mobile robots, moving object tracking in extensive outdoor environments based on radar, LiDAR and visual sensors.

Roland Chapuis received the PhD degree in 1991, Blaise Pascal Univ. France. From 1991 to 2002 he was an assistant professor, he became a full-Professor in 2002 in the Electrical department of Polytech Clermont-Ferrand engineering school of Blaise Pascal University in Clermont-Ferrand France. He is the head of Perception Systems team in Pascal Institute laboratory. His research interests include intelligent vehicles, dynamic objects recognition and tracking, multisensorial perception systems, multimodal data fusion, simultaneous localization and mapping.