Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 676385, 26 pages
doi:10.1155/2012/676385

Research Article

Monocular SLAM for Visual Odometry: A Full Approach to the Delayed Inverse-Depth Feature Initialization Method

Rodrigo Munguía[1, 2] and Antoni Grau[2]

[1] *Department of Computer Science, CUCEI, Universidad de Guadalajara, 44430 Guadalajara, JAL, Mexico*
[2] *Department of Automatic Control, UPC, BarcelonaTech, 08034 Barcelona, Spain*

Correspondence should be addressed to Antoni Grau, antoni.grau@upc.edu

Received 20 March 2012; Revised 6 July 2012; Accepted 8 July 2012

Academic Editor: Piermarco Cannarsa

This paper describes in a detailed manner a method to implement a simultaneous localization and mapping (SLAM) system based on monocular vision for applications of visual odometry, appearance-based sensing, and emulation of range-bearing measurements. SLAM techniques are required to operate mobile robots in *a priori* unknown environments using only on-board sensors to simultaneously build a map of their surroundings; this map will be needed for the robot to track its position. In this context, the 6-DOF (degree of freedom) monocular camera case (monocular SLAM) possibly represents the harder variant of SLAM. In monocular SLAM, a single camera, which is freely moving through its environment, represents the sole sensory input to the system. The method proposed in this paper is based on a technique called delayed inverse-depth feature initialization, which is intended to initialize new visual features on the system. In this work, detailed formulation, extended discussions, and experiments with real data are presented in order to validate and to show the performance of the proposal.

1. Introduction

The online robot estimation position from measurements of self-mapped features is a class of problem, in the robotics community, known as simultaneous localization and mapping (SLAM) problem. This technique consists in increasingly building a consistent map of the environment and, at the same time, localizing the robot's position while it explores its world. SLAM is perhaps the most fundamental problem to solve in robotics in order to build truly autonomous mobile robots.

Robot sensors have a large impact on the algorithm used in SLAM. Early SLAM approaches focused on the use of range sensors as sonar rings and lasers, see [1, 2]. Nevertheless there are some disadvantages with the use of range sensors in SLAM: correspondence or data association becomes difficult, they are expensive, and some of them are limited to 2D maps and computationally inefficient when the number of features is large (see [3, 4] for a complete survey).

The aforementioned issues have propitiated that recent works move towards the use of cameras as the primary sensing mode. Cameras have become more and more interesting for the robotics research community, because they yield a lot of valuable information for data association. A wide variety of algorithms can be obtained from the computer vision research community in order to extract high-level primitives from images. Those primitives are matched with the primitives stored in the map allowing thus the data association. This is still an open problem. A camera is a sensor that yields 3D information. Even for indoor robots whose pose can be represented in 2D, the ability to gather 3D information of the environment is essential. Cameras are also suitable for embedded systems: they are lightweight, cheap, and power saving. Using vision, a robot can locate itself using as landmarks common objects that encounters along its path.

In this context, the 6-DOF (degrees of freedom) monocular camera case (monocular SLAM) possibly represents the most difficult variant of SLAM; in monocular SLAM, a single camera can be freely moving by its environment representing the sole input sensor to the system. On the other hand, while range sensors (e.g., laser) provide range and angular information, a camera is a projective sensor which measures the bearing of image features. Therefore, depth information (range) cannot be obtained in a single frame. This drawback has triggered the emergence of especial techniques for feature initialization approaches in order to allow the use of bearing sensors (such as cameras) in SLAM systems.

As computational power grows, an inexpensive monocular camera can be used to perform simultaneously range and appearance-based sensing replacing typical sensors for range measurement (laser and sonar rings) and for dead reckoning (encoders). Thus, a camera connected to a computer becomes a position sensor which could be applied to different fields such as robotics (motion estimation basically in humanoids robots), wearable robotics (motion estimation for camera-equipped devices worn by humans), tele-presence (head motion estimation using an outward-looking camera), or television (camera motion estimation for live augmented reality) [5].

Monocular SLAM is closely related to the structure-from-motion (SFM) problem for reconstructing scene geometry. SFM techniques, which originally come from the computer vision research community, are sometimes formulated as off-line algorithms that require batch, simultaneous processing for all the images acquired in the sequence. Nevertheless, several recursive solutions to the SFM problem can be found in the literature. In this sense, one of the first works was presented by Matthies et al. in [6] though in this work it is assumed that the camera motion is known. By contrast, Gennery in [7] proposes a method to estimate motion from a known structure. A method for addressing the problem of the two above works, unknown structure as unknown motion, was first introduced by Azarbayejani et al. in [8]. Other SFM-based techniques are presented in works [9, 10]. Some hybrid techniques (SFM-Kalman Filtering) based on stereovision, as stated in [11], have also appeared.

Monocular SLAM has received much attention in the last years. In [12], Deans and Martial proposes a combination of a global optimization BA (bundle adjustment) for feature initialization and a Kalman filter for state estimation. Strelow proposes in [13] a similar method but mixing camera and inertial sensors measurements in an iterated extended Kalman filter (IEKF). In [14], Bailey proposes a variant of constrained initialization for bearing-only SLAM, where past estimates for the vehicle pose are retained in the SLAM state and thus feature initialization can be deferred until its estimates become well-conditioned. Moreover there are some works that use other estimation techniques (apart from EKF) like particle filters (PF) methods. In [15, 16], Kwok and Dissanayake propose a method based in particle filtering techniques. In this case the initial state of features is approximated using a sum of Gaussians, which defines a set of hypothesis for landmark position. In subsequent steps, bad hypotheses of depth are pruned while observations are done. A similar approach is presented in [17] for bearing-only tracking.

Davison et al., [5, 18], proposes a real-time method based on the well-established EKF framework as the main estimation technique. In those works a Bayesian partial initialization scheme for incorporating new landmarks is used. A separate particle filter, which is not correlated with the rest of the map, is used in order to estimate the feature depth prior to its inclusion in the map. It is important to note that prior to Davison, the feasibility of real-time monocular SLAM was first demonstrated by Jin et al. in [19].

In [20], Smith et al. presents a real-time method based on straight lines detection using the EKF estimator. Jensfelt et al. in [21] presents a method where the main idea is to let the SLAM estimation lag behind n frames; those n frames will determine which points are good landmarks to find an estimation of their 3D location. In [22], Solà et al. presents a method based on federate Kalman filtering technique. With an initial probability distribution function (PDF) for the features, a geometric sum of Gaussians is defined. The method is an approximation of the Gaussian sum filter (GSF) that permits undelayed initialization with an additive simple growth of the problem size. In [23] Lemaire et al. present a similar method to [22] but the features are initialized with a delayed method.

In [24], Eade and Drummond proposes a FastSLAM-based [25] approach. In that work the pose of the robot is represented by particles and a set of Kalman filters refines the estimation of the features. When the inverse depth collapses, the feature is converted into a fully initialized standard Euclidean representation. Montiel et al. in [26] present a new approach where the transition from partially to fully initialized features does not need to be explicitly tackled, making it suitable for direct use in EKF framework in sparse mapping. In this approach, the features are initialized in the first frame they are observed with an initial, fixed inverse depth and uncertainty heuristically determined to cover the range from nearby to infinite; therefore distant points can be coded.

In delayed methods, a feature observed in the instant t is added to the map in a subsequent time $t + k$. Usually the delay is used, in this kind of methods, for collecting information that permits to initialize the feature in a robust manner. On the other hand, the undelayed methods take advantage of the observation of the feature from the instant t for updating the system. Conversely the system update step should be computed carefully.

In authors' previous works, [27, 28], a monocular SLAM approach is proposed. In this paper a method, the so-called delayed inverse-depth feature initialization, is used to initializing new features in the system. This method, which is based on the inverse depth parameterization, defines a single hypothesis for the initial depth of features by the use of a stochastic technique of triangulation.

This paper presents an extended and detailed version of the monocular SLAM scheme proposed in [27, 28]. Though the fundamental idea of the original method remains the same, some important and remarkable improvements have been made and included in this paper. Moreover, the implementation of the proposed method is presented in a very detailed manner. New experiments with real data and discussion are also included in order to show the performance of the proposed method and its importance as an open problem in autonomous robotics engineering.

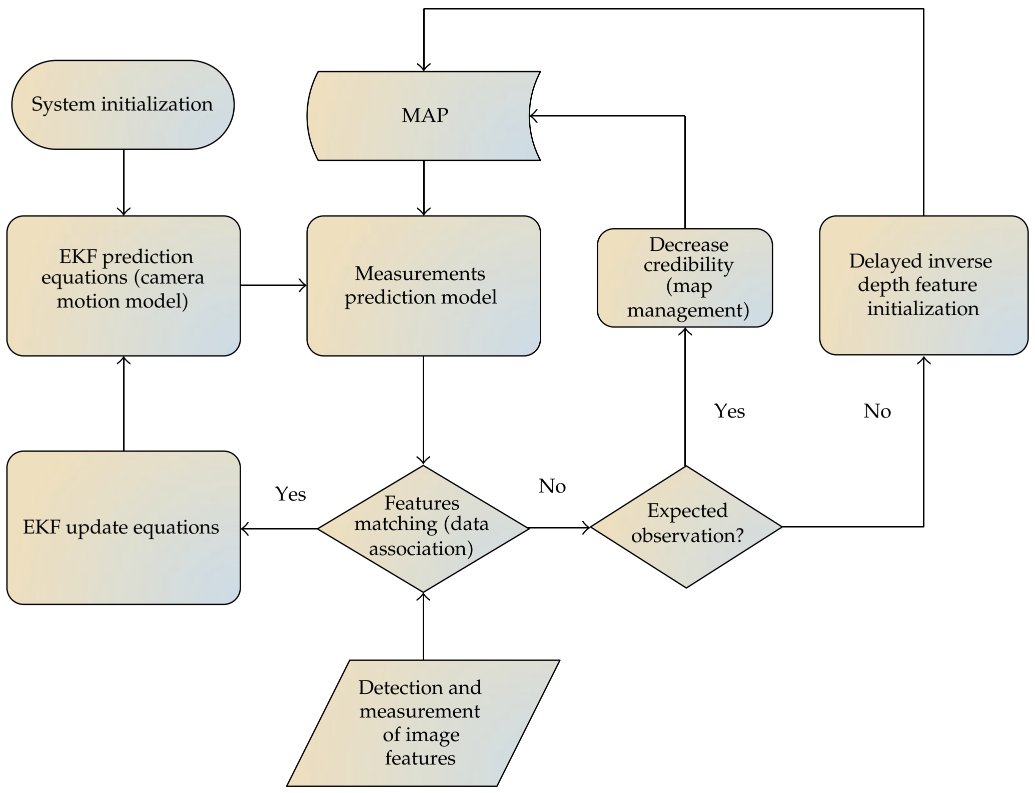

System initialization
MAP
EKF prediction equations (camera motion model)
Measurements prediction model
Decrease credibility (map management)
Delayed inverse depth feature initialization
Yes
No
EKF update equations
Yes
Features matching (data association)
No
Expected observation?
Detection and measurement of image features

Figure 1: Block diagram showing the architecture of the system.

2. Method Description

In this section, the proposed scheme for monocular SLAM is described. In Figure 1 a block diagram indicating the flow information and main sub-processes of the system is shown. All its components are explained in detail in subsequent sub-sections. It is important to remark that once the system is initialized, the detection and measurement of image features represents the sole sensory input of the system.

2.1. System Parameterization

The complete system state $\hat{x}$ consists of:

$$\hat{x} = \left[\hat{x}_v, \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\right]^T, \tag{2.1}$$

where $\hat{x}_v$ represents the state of a free robot camera moving in any direction in $\mathbb{R}^3 \times \mathrm{SO}(3)$, and $\hat{y}_i$ represents a feature point i in the 3D scene. At the same time, $\hat{x}_v$ is composed of:

$$\hat{x}_v = \left[r^{WC} \quad q^{WC} \quad v^W \quad \omega^W\right]^T, \tag{2.2}$$

where

$$r^{WC} = \left[x_v \;\; y_v \;\; z_v\right]^T, \tag{2.3}$$

represents the camera optical center position in Cartesian coordinates, and:

$$q^{WC} = \left[q_1 \;\; q_2 \;\; q_3 \;\; q_4\right]^T \tag{2.4}$$

represents the orientation of the camera respect to the navigation frame by a unit quaternion. Unit quaternions provide a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. Compared with Euler angles they are simpler to compose avoiding the problem of gimbal lock. Compared with rotation matrices they are numerically more stable and in many cases they are more efficient. A good review for attitude representations is given in [29]. The following expressions:

$$\begin{aligned} v^W &= \left[v_x \;\; v_y \;\; v_z\right]^T, \\ \omega^W &= \left[\omega_x \;\; \omega_y \;\; \omega_z\right]^T, \end{aligned} \tag{2.5}$$

denote linear and angular velocities, respectively. The superscripts W and WC denote magnitudes expressed in the world reference, and in the camera reference respectively.

A feature $\hat{y}_i$ is composed of the following 6-dimension state vector:

$$\hat{y}_i = \left[x_i \;\; y_i \;\; z_i \;\; \theta_i \;\; \phi_i \;\; \rho_i\right]^T, \tag{2.6}$$

which models the 3D point located at:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i), \tag{2.7}$$

where x_i, y_i, z_i corresponds to the optical center coordinates of the camera when the feature was observed for the very first time; and θ_i, ϕ_i, represent the azimuth and the elevation respectively (respect to the world reference W) for the directional unitary vector $m(\theta_i, \phi_i)$:

$$m(\theta_i, \phi_i) = \left[\cos\phi \sin\theta \;\; -\sin\phi \;\; \cos\phi\cos\theta\right]^T \tag{2.8}$$

The point depth r_i is coded by its inverse value, $\rho_i = 1/r_i$, as quoted in [26]. Figure 2 illustrates the camera and features parameterization.

2.2. System Initialization

In a robotics context, obtaining the metric scale of the world can be very useful. However, in monocular SLAM the scale of the observed world cannot be obtained using only vision, and therefore another sensor or the observation of a known dimension reference have to be used in order to retrieve the scale of the world.

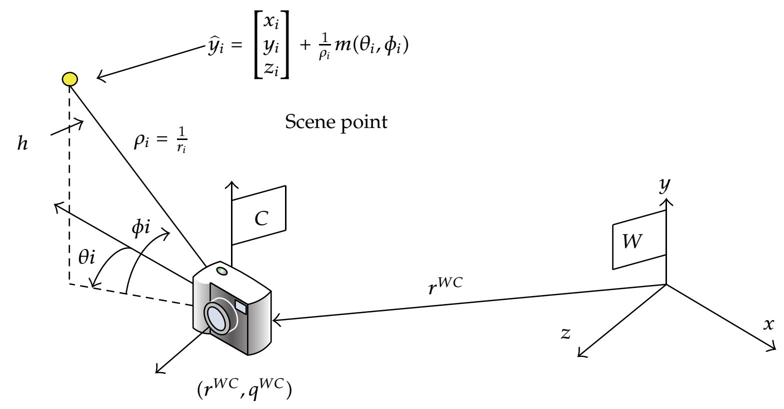

$\hat{y}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i)$
Scene point
h
$\rho_i = \frac{1}{r_i}$
ϕi
θi
C
W
r^{WC}
(r^{WC}, q^{WC})
y
z
x

Figure 2: Camera and features parameterization.

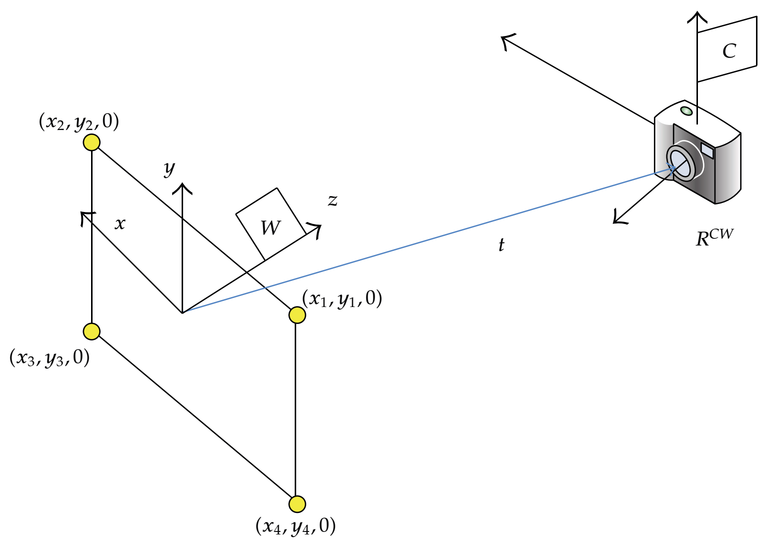


Figure 3: Perspective of 4 coplanar points (P4P) problem.

In this work, the system metric initialization process is analogous to a classical problem in computer vision: the perspective of n-point (PnP) problem [30]. The PnP problem is to find the position and orientation of a camera with respect to a scene object from n correspondent points. In [30] it is demonstrated that, when the control points are coplanar, the perspective on 4-point (P4P) problem has a only solution.

Therefore, in the present work it is assumed that 4 coplanar points are known (for instance, the dimensions of a black paper sheet over a white background). It is also assumed that the intrinsic parameters of the camera are already known: (i) f (focal length); (ii) i_0, j_0 (displacement to the image center), and (iii) $k_1, \ldots, k_n$ (radial lens distortion).

The problem consists in estimating two extrinsic camera parameters: R^{CW} (world to camera rotation matrix for camera orientation) and t (translation vector for the position of the camera center), given 4 coplanar points with spatial coordinates $(x_i, y_i, 0)$, with i in [1..4], and their corresponding 4 undistorted image coordinates (i, j), as shown in Figure 3.

In order to estimate R^{CW} and t, the method is based in the approach proposed in [31]. The following system of linear equations is formed with the vector b as an unknown parameter:

$$\begin{bmatrix} x_1 f & y_1 f & 0 & 0 & -i_1 x_1 & -i_1 y_1 & f & 0 \\ 0 & 0 & x_1 f & y_1 f & -j_1 x_1 & -j_1 y_1 & 0 & f \\ x_2 f & y_2 f & 0 & 0 & -i_2 x_2 & -i_2 y_2 & f & 0 \\ 0 & 0 & x_2 f & y_2 f & -j_2 x_2 & -j_2 y_2 & 0 & f \\ x_3 f & y_3 f & 0 & 0 & -i_3 x_3 & -i_3 y_3 & f & 0 \\ 0 & 0 & x_3 f & y_3 f & -j_3 x_3 & -j_3 y_3 & 0 & f \\ x_4 f & y_4 f & 0 & 0 & -i_4 x_4 & -i_4 y_4 & f & 0 \\ 0 & 0 & x_4 f & y_4 f & -j_4 x_4 & -j_4 y_4 & 0 & f \end{bmatrix} b = \begin{bmatrix} i_1 \\ j_1 \\ i_2 \\ j_2 \\ i_3 \\ j_3 \\ i_4 \\ j_4 \end{bmatrix}, \tag{2.9}$$

where

$$b = \left[\frac{r_{11}}{t_3} \ \frac{r_{12}}{t_3} \ \frac{r_{21}}{t_3} \ \frac{r_{22}}{t_3} \ \frac{r_{31}}{t_3} \ \frac{r_{32}}{t_3} \ \frac{t_1}{t_3} \ \frac{t_2}{t_3} \right]^T. \tag{2.10}$$

The linear system in (2.9) is solved for $b = [b_1 \ b_2 \ b_3 \ b_4 \ b_5 \ b_6 \ b_7 \ b_8]^T$. Then t_3 is obtained as follows:

$$t_3 = \sqrt{\frac{f^2}{b_1^2 + b_3^2 + f^2 b_5^2}}. \tag{2.11}$$

The extrinsic parameters R^{CW} and t are obtained using the following equations:

$$R^{CW} = \begin{bmatrix} t_3 b_1 & t_3 b_2 & (R_{21}R_{32} - R_{31}R_{22}) \\ t_3 b_3 & t_3 b_4 & (R_{31}R_{12} - R_{11}R_{32}) \\ t_3 b_5 & t_3 b_6 & (R_{11}R_{22} - R_{21}R_{12}) \end{bmatrix}, \tag{2.12}$$

$$t = \left[t_3 b_7 \ \ t_3 b_8 \ \ t_3 \right]^T. \tag{2.13}$$

In (2.12) the third column of matrix R^{CW} is formed by the combinations of the values of first and second column of the same matrix. A good review of algorithms for coplanar camera calibration can be found in [29].

At the beginning of the iterative process, the system state $\hat{x}_{\text{ini}}$ is formed by the camera-state $\hat{x}_v$, and the four initial features used for estimating the extrinsic camera parameters:

$$\hat{x}_{\text{ini}} = \left[r_{\text{ini}}^{WC} \ \ q_{\text{ini}}^{WC} \ \ v_{\text{ini}}^{W} \ \ \omega_{\text{ini}}^{W} \ \ \hat{y}_1 \ \ \hat{y}_2 \ \ \hat{y}_3 \ \ \hat{y}_4 \right]^T, \tag{2.14}$$

where

$$r_{\text{ini}}^{WC} = t, \qquad q_{\text{ini}}^{WC} = q\left(\left(R^{CW}\right)^{T}\right), \qquad v_{\text{ini}}^{W} = [0_{3\times 1}],$$
$$\omega_{\text{ini}}^{W} = [0_{3\times 1}], \tag{2.15}$$

r_{ini}^{WR} has the same value that the extrinsic parameter t calculated by (2.13). q_{ini}^{WC} is estimated using the rotation matrix to quaternion transformation $q(R)$, described by (A.2) in the Appendix, from the transpose of the rotation matrix $(R^{CW})^T$ obtained by (2.12).

Each initial feature $\hat{y}_i$ with i in [1..4] is composed by:

$$\hat{y}_{\text{i}} = \left[r_{\text{ini}}^{WC} \;\; \operatorname{atan2}(g_1, g_3) \;\; \operatorname{atan2}\left(-g_2, \sqrt{g_1^2 + g_3^2}\right) \;\; \frac{1}{\|r_{\text{ini}}^{WC}\|} \right]^{T}, \tag{2.16}$$

where

$$\left[g_1 \;\; g_2 \;\; g_3\right] = \left[x_i \;\; y_i \;\; 0\right] - r_{\text{ini}}^{WC}, \tag{2.17}$$

atan2 is a two-argument function that computes the arctangent of y/x for given values of y and x, within the range $[-\pi, \pi]$. If certain confidence is assumed in the estimation process for the initial system state $\hat{x}_{\text{ini}}$, then the initial covariance matrix of the system P_{ini} can be filled with zeros or ε, being this value an arbitrary very small positive value:

$$P_{\text{ini}} = [\varepsilon_{37\times 37}]. \tag{2.18}$$

2.3. Camera Motion Model and Filter Prediction Equations

An unconstrained prediction model for a constant-acceleration camera motion can be defined by the following equation [5, 32]:

$$f_v = \begin{bmatrix} r_{k+1}^{WC} \\ q_{k+1}^{WC} \\ v_{k+1}^{W} \\ \omega_{k+1}^{W} \end{bmatrix} = \begin{bmatrix} r_k^{WC} + (v_k^W + V_k^W)\Delta t \\ q_k^{WC} \times q((\omega_k^W + \Omega^W)\Delta t) \\ v_k^W + V^W \\ \omega_k^W + \Omega^W \end{bmatrix}, \tag{2.19}$$

being $q((\omega_k^W + \Omega^W)\Delta t)$ the quaternion computed from the rotation vector $(\omega_k^W + \Omega^W)\Delta t$. This transformation from rotation vector to quaternion is defined by (A.1) in the Appendix.

No prior assumption can be made about the camera movement. Therefore, the model defined in (2.19) supposes, at every step k, that the changes in linear and angular velocity v^W and ω^W are produced by an input u of unknown linear and angular velocity V^W and Ω^W, with linear and angular acceleration with zero-mean and known Gaussian process covariance, a^W and α^W, respectively:

$$u = \begin{bmatrix} V^W \\ \Omega^W \end{bmatrix} = \begin{bmatrix} a^W \Delta t \\ \alpha^W \Delta t \end{bmatrix}. \tag{2.20}$$

The unknown linear and angular velocity V^W and Ω^W are incorporated into the system by the process noise covariance matrix U:

$$U = \begin{bmatrix} (\sigma_V \Delta t)^2 I_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & (\sigma_\Omega \Delta t)^2 I_{3\times 3} \end{bmatrix}. \tag{2.21}$$

If a static scene is assumed where the landmarks remain in a fixed place, the prediction step for the Extended Kalman Filter (EKF) is defined as follows:

$$\widehat{x}_{k+1} = \begin{bmatrix} f_v(\widehat{x}_v) \\ \widehat{y}_1 \\ \vdots \\ \widehat{y}_n \end{bmatrix}, \tag{2.22}$$

$$P_{k+1} = \nabla F_x P_k F_x^T + \nabla F_u Q \nabla F_u^T, \tag{2.23}$$

where Q and the Jacobians ∇F_x and ∇F_u are defined as follows:

$$\nabla F_x = \begin{bmatrix} \dfrac{\partial f_v}{\partial \widehat{x}_v} & 0_{13\times n} \\ 0_{n\times 13} & I_{n\times n} \end{bmatrix}, \qquad \nabla F_u = \begin{bmatrix} \dfrac{\partial f_v}{\partial u} & 0_{13\times n} \\ 0_{n\times 6} & 0_{n\times n} \end{bmatrix}, \qquad Q = \begin{bmatrix} U & 0_{6\times n} \\ 0_{n\times 6} & 0_{n\times n} \end{bmatrix}. \tag{2.24}$$

$\partial f_v / \partial \widehat{x}_v$ are the derivatives of the equations of the prediction model f_v with respect to the camera state $\widehat{x}_v$. $\partial f_v / \partial u$ are the derivatives of the equations of the prediction model f_v with respect to the parameters of the covariance matrix U. In this case n corresponds to the dimension of the features $\widehat{y}_i$ in the system state $\widehat{x}_v$.

2.4. Measurement Prediction Model

The different locations of the camera, along with the location of the features already mapped, are used by the measurement prediction model h_i to forecast the distorted pixel coordinates u_d and v_d for each feature $\widehat{y}_i$:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = h_i(\widehat{x}_v, \widehat{y}_i), \tag{2.25}$$

$h_i(\hat{x}_v, \hat{y}_i)$ is derived using the following procedure: the model observation of any point $\hat{y}_i$ from a camera location defines a ray expressed in the camera frame as follows, see Figure 2:

$$h^c = \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = R^{CW}\left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i) - r^{WC}\right). \tag{2.26}$$

h^C is observed by the camera through its projection in the image. R^{CW} is the transformation matrix from the global reference frame to the camera reference frame. R^{CW} *is* computed by $R^{CW} = (R^{CW}(q^{WC}))^T$ where $R^{WC}(q^{WC})$ is a rotation matrix depending on the camera orientation quaternion q^{WC} using (A.3) which computes a rotation matrix from a quaternion. The directional unitary vector $m(\theta_i, \phi_i)$ is computed with (2.8).

The vector h^C is projected to the normalized camera retina by the following:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dfrac{h_x}{h_z} \\ \dfrac{h_y}{h_z} \end{bmatrix}. \tag{2.27}$$

The camera calibration model is applied to produce the undistorted pixel coordinates u_u and v_u:

$$\begin{bmatrix} u_u \\ v_u \end{bmatrix} = \begin{bmatrix} u_0 - fu \\ v_0 - fv \end{bmatrix}, \tag{2.28}$$

where u_0, v_0 is the camera center in pixels, and f is the focal length. Finally, a radial distortion model is applied to obtain the distortion pixel coordinates u_d, v_d. In [18] the following distortion model is proposed being K_1 the distortion coefficient:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = \begin{bmatrix} \dfrac{u_u - u_0}{\sqrt{1 + 2K_1 r^2}} + u_0 \\ \dfrac{v_u - v_0}{\sqrt{1 + 2K_1 r^2}} + v_0 \end{bmatrix}, \quad r = \sqrt{(u_u - u_0)^2 + (v_u - v_0)^2}. \tag{2.29}$$

2.5. Features Matching and Filter Update Equations

If the feature $\hat{y}_i$ is predicted to be appeared in the image ($0 < u_d <$ image height & $0 < v_d <$ image width), then an active search technique [33] can be used to constrain the features matching from frame to frame inside elliptical regions around the predicted $h_i = [u_d, v_d]$. The elliptical regions are defined by the innovation covariance matrix S_i:

$$S_i = \nabla H_i P_{k+1} \nabla H_i^T + R_{uv}. \tag{2.30}$$

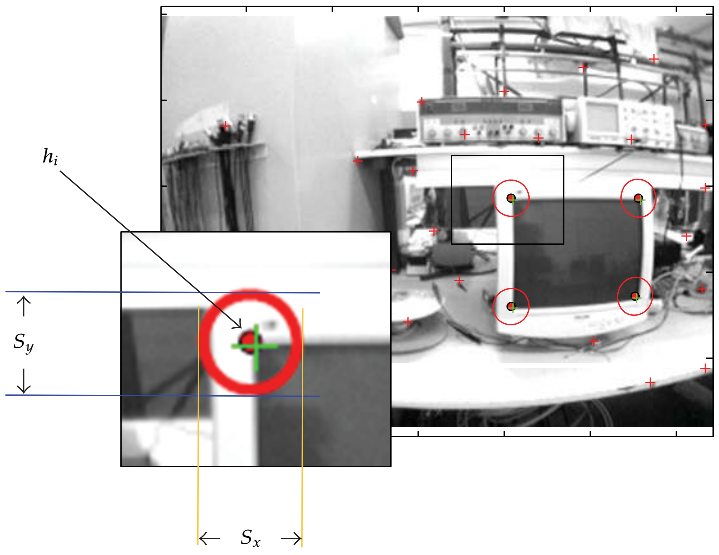


Figure 4: An active search is used to maximize the computational speed and to reduce the mismatch likelihood in the data association process. Feature search is constrained to regions around the predicted h_i. The regions are defined by the innovation covariance S_i as in (2.30).

Measurement noise is incorporated into the system by the noise covariance matrix R_{uv}:

$$R_{uv} = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}. \tag{2.31}$$

Variances σ_u^2 and σ_v^2 are defined in pixel units. P_{k+1} is the prior state covariance estimated by (2.23). ∇H_i is the Jacobian derived from the measurement model h_i:

$$\nabla H_i = \left[\frac{\partial h_i}{\partial \hat{x}_v} \quad \cdots 0_{2\times 6} \cdots \quad \frac{\partial h_i}{\partial \hat{y}_i} \quad \cdots 0_{2\times 6} \cdots \right], \tag{2.32}$$

$\partial h_i / \partial \hat{x}_v$ are the derivatives of the equations of the measurement prediction model h_i with respect to the camera state $\hat{x}_v$. $\partial h_i / \partial \hat{y}_i$ are the derivatives of the measurement prediction model h_i with respect to the parameters of the feature $\hat{y}_i$. Note that $\partial h_i / \partial \hat{y}_i$ has only a nonzero value at the location (indexes) of the observed feature $\hat{y}_i$.

The size of the axis s_x, s_y for the elliptical search region defined by the matrix S_i is determined by (see Figure 4):

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} 2n\sqrt{S_{i(1,1)}} \\ 2n\sqrt{S_{i(2,2)}} \end{bmatrix}, \tag{2.33}$$

where n is the number of standard deviations for the desired region of search.

When a feature $\hat{y}_i$ is added to the map, a unique image patch of n-by-n pixel is stored and linked with the feature. For matching a feature in the current image frame, a patch cross-correlation technique, [33], is applied over all the image locations (u_d, v_d) within the search region defined in (2.33). If the score of a pixel location (u_d, v_d), determined by the best cross correlation between the feature patch and the patches defined by the region of search, is higher than a threshold then this pixel location (u_d, v_d) is considered as the current feature measurement z_i.

If the matching process is successful, then the filter is updated as follows:

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k+1} + Wg, \\ P_k &= P_{k+1} - WS_iW^T, \end{aligned} \tag{2.34}$$

where the innovation g is as follows:

$$g = z_i - h_i, \tag{2.35}$$

and W is the Kalman gain:

$$W = P_{k+1} \nabla H_i^T S_i^{-1}. \tag{2.36}$$

In SLAM, it is well known the injurious effect of incorrect or incompatible matches. In monocular SLAM systems, when delayed feature initialization techniques are used (as in this work), implicitly some weak image features are pruned prior to their addition to the stochastic map, for example, image features produced by fast lighting changes, shines in highly reflective surfaces, or even caused by some dynamic elements in the scene. Nevertheless, the risk of incorrect or incompatible matches could remain due to several factors as: (i) incompatibility due repeated design; (ii) fake composite landmark; (iii) incompatibilities produced by reflections on curved surfaces and materials; (iv) detection of landmarks running along edges. The above drawbacks can be mitigated by the used of batch validation techniques [34, 35] (or variants of them), with the penalty of the increment of computational time.

On the other hand, for the standard EKF-SLAM algorithm, there exists significant empirical evidence which demonstrates that the computed state estimates tend to be inconsistent. Thus, if the covariance matrix is underestimated, then the search regions determined by the active search technique could even collapse, leading the tracking to fail. A feasible solution to this problem could be implemented by fixing a minimum possible size for the axes s_x, s_y in (2.33).

2.6. Delayed Inverse-Depth Feature Initialization

As it is stated before, depth information cannot be obtained in a single measurement when bearing sensors (e.g., a single camera) are used. To infer the depth of a feature, the sensor must observe this feature repeatedly as it freely moves through its environment, estimating the angle from the feature to the sensor center. The difference between those angle measurements is the parallax feature. Actually, parallax is the key that allows estimating features depth. In case of indoor sequences, a displacement of centimeters is enough to produce parallax; on the other hand, the more distant the feature, the more the sensor has to travel to produce parallax. For incorporating new features to the map, special techniques

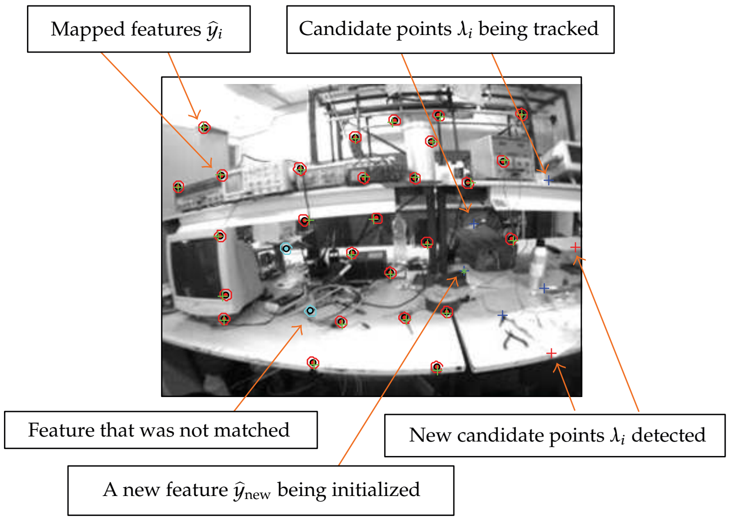

Mapped features $\hat{y}_i$
Candidate points λ_i being tracked
Feature that was not matched
New candidate points λ_i detected
A new feature $\hat{y}_{\text{new}}$ being initialized

Figure 5: New features will be added to the map if the number of predicted features $\hat{y}_i$ to be appearing in the image is lower than a threshold. First, random feature-free areas are detected in the image (green rectangle). Then, a saliency operator is applied in order to detect new candidate points λ_i.

for features initialization are needed in order to enable the use of bearing sensors in SLAM systems.

In this work, authors propose a novel method called delayed inverse-depth feature initialization in order to incorporate new features $\hat{y}_{\text{new}}$ to the stochastic map from the bearing measurements of the camera. The proposed method uses the inverse-depth parameterization and implements an stochastic technique of triangulation, by means of delay, in order to define hypothesis of initial depth for the features. The use of the inverse-depth parameterization, in the context of motion estimation and scene reconstruction using vision, was introduced earlier by Favaro in [36], and extended later in [24, 26].

The proposed method states that a minimum number of features $\hat{y}_i$ is considered to be predicted appearing in the image, otherwise new features should be added to the map. In this latter case, new points are detected in the image with a saliency operator. In this particular case, Harris corner detector is applied, but other detectors should also be used.

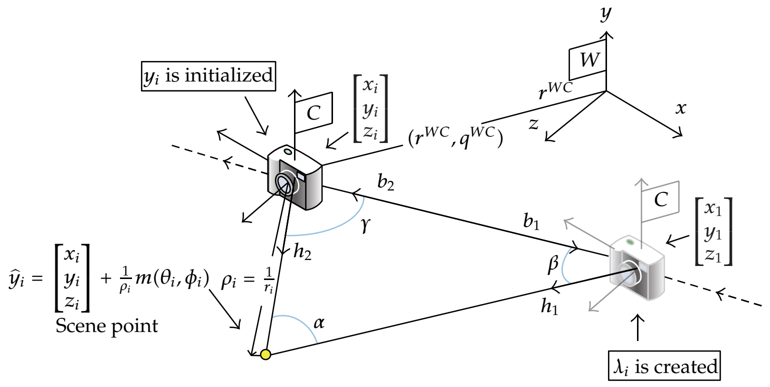
These points in the image, which are not added yet to the map, are called candidate points, λ_i. Only image areas free of both, candidate points λ_i and mapped features $\hat{y}_i$, are considered for detecting new points with the saliency operator, see Figure 5.

When a candidate point λ_i is detected for the first time, some data are stored:

$$\lambda_i = (r_\lambda, \sigma_r, q_\lambda, \sigma_q, u_1, v_1), \tag{2.37}$$

where r_λ is taken from the current camera optical center position r^{WC}:

$$r_\lambda = (x_1, y_1, z_1) = (x_v, y_v, z_v), \tag{2.38}$$



y_i is initialized
$\hat{y}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i)$ $\rho_i = \frac{1}{r_i}$
Scene point
$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$
(r^{WC}, q^{WC})
C
W
r^{WC}
x
y
z
b_2
b_1
h_2
h_1
γ
β
α
$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$
λ_i is created

Figure 6: Representation for the feature initialization process.

σ_r represents the associated variances of r^{WC}, taken from the state covariance matrix P:

$$\sigma_r = (\sigma_{x1}, \sigma_{y1}, \sigma_{z1}) = (P_{(1,1)}, P_{(2,2)}, P_{(3,3)}), \tag{2.39}$$

q_λ is taken from the quaternion q^{WC} representing the current camera orientation:

$$q_\lambda = q^{WC}, \tag{2.40}$$

σ_q represents the associated variances of q^{WC}, taken from the state covariance matrix P:

$$\sigma_q = (\sigma_{1q1}, \sigma_{1q2}, \sigma_{1q3}, \sigma_{1q4}) = (P_{(4,4)}, P_{(5,5)}, P_{(6,6)}, P_{(7,7)}), \tag{2.41}$$

and u_1, v_1 are the initial pixel coordinates where the candidate point λ_i was detected for the first time. In subsequent frames, each candidate point λ_i is intended to be tracked (this approach is independent of the used tracking method) in order to maintain the pixel coordinates u and v updated for such a candidate point. In practice, not all the candidate points λ_i can be tracked, nevertheless the method takes advantage of this fact in order to prune weak image features. For each candidate point λ_i, the tracking process is realized until the point is pruned or initialized in the system as a new feature $\hat{y}_{\text{new}}$. In practice for each frame, the candidate points λ_i could be detected, pruned or considered to be added to the map.

A candidate point λ_i is added as a new feature $\hat{y}_{\text{new}}$ when a minimum threshold of parallax $\alpha_{\min}$ is reached. Parallax α is estimated using (as it can be seen in Figure 6):

(i) displacement base-line b;

(ii) λ_i and associated data;

(iii) the current camera state $\hat{x}_v$ and current measurement z_i.

For each candidate point λ_i, when a new measurement z_i is available then parallax angle α is estimated by the following:

$$\alpha = \pi - (\beta + \gamma). \tag{2.42}$$

The angle β is determined by the directional unitary vector h_1 and the vector b_1 defining the base-line b in the direction of the camera trajectory by the following:

$$\beta = \cos^{-1}\left(\frac{h_1 \cdot b_1}{\|h_1\|\|b_1\|}\right), \tag{2.43}$$

where the directional projection ray vector $h_1 = [h_{1x}\ h_{1y}\ h_{1z}]$, expressed in the absolute frame W, is computed from the camera position and the coordinates of the observed point when it was observed for the first time, using the data stored in λ_i.

$$h_1 = R^{WC}(q_\lambda)h_1^C(u_{1u}, v_{1u}). \tag{2.44}$$

$R^{WC}(q_\lambda)$ is the transformation matrix from the camera reference frame to the global reference frame, depending on the stored quaternion q_λ; (A.3) is used for computing a rotation matrix from a quaternion.

$h_1^C(u_{1u}, v_{1u}) = [h_{1x}\ h_{1y}\ h_{1z}]$ is the directional vector, in the camera frame C, pointing from the position when the candidate point λ_i was observed for the first time to the feature location, and it can be computed by the following:

$$h_1^C(u_{1u}, v_{1u}) = \left[\frac{u_0 - u_{1u}}{f}\quad \frac{v_0 - v_{1u}}{f}\quad 1\right]. \tag{2.45}$$

The undistorted pixel coordinates u_{1u} and v_{1u} are obtained from u_1, v_1 applying the inverse of the distortion model in (2.29):

$$\begin{bmatrix} u_{1u} \\ v_{1u} \end{bmatrix} = \begin{bmatrix} \dfrac{u_1 - u_0}{\sqrt{1 - 2K_1 r^2}} + u_0 \\ \dfrac{v_1 - v_0}{\sqrt{1 - 2K_1 r^2}} + v_0 \end{bmatrix}, \quad r = \sqrt{(u_1 - u_0)^2 + (v_1 - v_0)^2}. \tag{2.46}$$

$b_1 = [b_{1x}\ b_{1y}\ b_{1z}]$ is the vector representing the camera base-line b between the camera optical center position $[x_1\ y_1\ z_1]$, where the point was observed for the first time, and the current camera optical center $r^{WC} = [x_v\ y_v\ z_v]$, taken from the current camera state $\hat{x}_v$:

$$b_1 = \left[(x_v - x_1)\ \ (y_v - y_1)\ \ (z_v - z_1)\right]. \tag{2.47}$$

The angle γ is determined in a similar way as β but using instead the directional projection ray vector h_2 and the vector b_2:

$$\gamma = \cos^{-1}\left(\frac{h_2 \cdot b_2}{\|h_2\|\|b_2\|}\right). \tag{2.48}$$

The directional projection ray vector h_2 expressed in the absolute frame W, is computed in a similar way as h_1 but using the current camera position $\hat{x}_v$ and therefore, the current undistorted points coordinates u_u, v_u:

$$h_2 = R^{WC}\left(q^{WC}\right)h_2^C(u_u, v_u), \tag{2.49}$$

$R^{WC}(q^{WC})$ is the transformation matrix from the camera reference frame to the global reference frame, depending on the current quaternion q^{WC}. Equation (A.3) is used for computing a rotation matrix from a quaternion.

$h_2^C(u_u, v_u) = [h_{2x}\ h_{2y}\ h_{2z}]$ is the directional vector, in the camera frame C, pointing from the current camera position to the feature location, and it can be computed as follows:

$$h_2^C(u_u, v_u) = \begin{bmatrix} \dfrac{u_0 - u_u}{f} & \dfrac{v_0 - v_u}{f} & 1 \end{bmatrix}. \tag{2.50}$$

The undistorted coordinates pixel u_u and v_u are obtained from the current pixel coordinates u, v applying the inverse of the distortion model in (2.29):

$$\begin{bmatrix} u_u \\ v_u \end{bmatrix} = \begin{bmatrix} \dfrac{u - u_0}{\sqrt{1 - 2K_1 r^2}} + u_0 \\ \dfrac{v - v_0}{\sqrt{1 - 2K_1 r^2}} + v_0 \end{bmatrix}, \quad r = \sqrt{(u - u_0)^2 + (v - v_0)^2}. \tag{2.51}$$

$b_2 = [b_{2x}\ b_{2y}\ b_{2z}]$ is similar to b_1 but pointing to the opposite direction:

$$b_2 = \left[(x_1 - x_v)\ \ (y_1 - y_v)\ \ (z_1 - z_v)\right]. \tag{2.52}$$

The base-line b is the module of b_1 or b_2:

$$b = \|b_1\| = \|b_2\|. \tag{2.53}$$

If $\alpha > \alpha_{\min}$ then λ_i is initialized as a new feature map $\hat{y}_{\text{new}}$. The feature $\hat{y}_{\text{new}}$ is made up of:

$$\hat{y}_{\text{new}} = g(\hat{x}, \lambda_i, z_i) = \left[x_i\ \ y_i\ \ z_i\ \ \theta_i\ \ \phi_i\ \ \rho_i\right]^T. \tag{2.54}$$

The first three elements of $\hat{y}_{\text{new}}$ are obtained directly from the current camera optical center position $r^{WC} = [x_v, y_v, z_v]$ taken from $\hat{x}_v$:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix}, \\ \begin{bmatrix} \theta_i \\ \phi_i \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left(-h_{2y}, \sqrt{h_{2x}^2 + h_{2z}^2}\right) \\ \operatorname{atan2}(h_{2x}, h_{2z}) \end{bmatrix}, \tag{2.55}$$

where $[h_{2x}\ h_{2y}\ h_{2z}]$ is computed using (2.49).

The inverse depth ρ_i is derived from the following:

$$[\rho_i] = \frac{\sin(\alpha)}{b\sin(\beta)}. \tag{2.56}$$

The new system state $\hat{x}_{\text{new}}$ is made up simply adding the new feature $\hat{y}_{\text{new}}$ to the final part of the vector state $\hat{x}$. If $\hat{x} = [\hat{x}_v\ \hat{y}_1\ \hat{y}_2]^T$ then $\hat{x}_{\text{new}}$ is composed by the following:

$$\hat{x} = \begin{bmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix}, \qquad \hat{x}_{\text{new}} = \begin{bmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_{\text{new}} \end{bmatrix}. \tag{2.57}$$

The new state covariance matrix P_{new} is derived from the error diagonal measurement covariance matrix R_j and the current state covariance matrix P:

$$R_j = \operatorname{diag}\left[\sigma_u^2\ \ \sigma_v^2\ \ \sigma_u^2\ \ \sigma_v^2\ \ \sigma_{x1}\ \ \sigma_{y1}\ \ \sigma_{z1}\ \ \sigma_{1q1}\ \ \sigma_{1q2}\ \ \sigma_{1q3}\ \ \sigma_{1q4}\right], \tag{2.58}$$

where R_j is composed by the measurement error variances σ_u^2 and σ_v^2 (defined in pixel units), and the variances stored with the candidate point λ_i.

The new state covariance matrix P_{new} after the initialization process is given by the following:

$$P_{\text{new}} = \nabla Y \begin{bmatrix} P_k & 0 \\ 0 & R_j \end{bmatrix} \nabla Y^T. \tag{2.59}$$

The Jacobian ∇Y is composed of the following:

$$\nabla Y = \begin{bmatrix} I_{n\times n} & 0 \\ \dfrac{\partial g}{\partial \hat{x}_v}, 0_{6\times 6}, \ldots, 0_{6\times 6}, & \dfrac{\partial g}{\partial R_j} \end{bmatrix}, \tag{2.60}$$

where I is a n-dimensional identity matrix where n is the dimension of P_k. $\partial g/\partial \hat{x}_v$ are the derivatives of the initializations equations $g(\hat{x}, \lambda_i, z_i)$ with respect to the camera state $\hat{x}_v$. Note that the derivatives with respect to the previously mapped features are filled with zeros. $\partial g/\partial R_j$ are the derivatives of the initializations equations $g(\hat{x}, \lambda_i, z_i)$ with respect to the parameters of the covariance matrix R_j.

For certain environments, those features located far away from the camera will not produce parallax (for small trajectories with respect to the feature depth). To include features in the map with big uncertainty in depth is not very helpful for estimating the camera location r^{WC}, although this kind of features could provide useful information for estimating camera orientation q^{WC}.

A special case is when a near feature does not produce parallax. This case happens when the camera moves in the direction towards the feature position (e.g., $\beta \approx 0°$). In this work it is assumed that there exist the necessary conditions for producing parallax at every time at least for one feature observed by the camera. Furthermore, features located in front of the direction of the camera movement will be discarded (if $\beta < 20°$).

For the proposed method, regarding whether distant features should be included in the map, a minimum base-line $b_{\min}$ can be used as an additional threshold for considering a candidate point λ_i to be initialized as a new feature $\hat{y}_{\text{new}}$. In this case, a candidate point λ_i is considered to be distant from the camera when $b > b_{\min}$ but $\alpha < \alpha_{\min}$. Camera moved from its previous position but no enough parallax has been produced, due to the distant condition of the landmark.

The value of $b_{\min}$ can be heuristically determined depending on the particularities of the application. In this case, for initializing a candidate point λ_i as a new feature $\hat{y}_{\text{new}}$, where $b > b_{\min}$ but $\alpha < \alpha_{\min}$, the simple method for initializing new features proposed in [26] can be used. Nevertheless, in this work the following equations are proposed in order to determine initial values ρ_{ini}, $\sigma_{\rho(\text{ini})}$ depending on $b_{\min}$ and $\alpha_{\min}$:

$$\rho_{\text{ini}} < \frac{2\sin(\alpha_{\min}/2)}{b_{\min}} \qquad \sigma_{\rho(\text{ini})} = \frac{\rho_{\min}}{4}. \tag{2.61}$$

2.7. Map Management

When the number of features in the system state increases then computational cost grows rapidly and consequently it becomes difficult to maintain the frame rate operation. To alleviate this drawback, old features can be removed from the state for maintaining a stable number of features and, therefore, to stabilize the computational cost per frame. Obviously, if old features are removed, then previous mapped areas cannot be recognized in the future. However, in the context of visual odometry, this fact is not considered as a problem. A modified monocular SLAM method that maintains the computational operation stable can be viewed as a complex real-time virtual sensor, which provides appearance-based sensing and emulates typical sensors as laser for range measurements and encoders for dead reckoning (visual odometry).

In authors' related work [37], the *virtual sensor* idea is proposed, developed, and extended so that, using a distributed scheme, it is possible to recognize previous mapped areas for loop closing tasks. In this scheme, the outputs provided by the virtual sensor have been used by an additional SLAM process (decoupled from the camera's frame rate) in order to build and to maintain the global map and final camera pose.

This present work regards the approach of monocular SLAM as a virtual sensor. Therefore, if the number of features exceeds a specified threshold, in order to maintain a stable amount of features in the system the algorithm removes older features that were left behind by camera movement as well as those features that have been predicted to be appearing in the image, but have not been matched in several previous frames.

Removing features from the system state is much easier than adding them. To delete a feature from the state vectorand covariance matrix, the rows and columns which contain it have to be removed. As an example, the remaining matrices after the removal of feature y_2 will be the following:

$$\begin{pmatrix} x_v \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \longrightarrow \begin{pmatrix} x_v \\ y_1 \\ y_3 \end{pmatrix} \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & P_{xy_3} \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & P_{y_1y_3} \\ P_{y_2x} & P_{y_2y_2} & P_{y_2y_2} & P_{y_2y_3} \\ P_{y_3x} & P_{y_3y_1} & P_{y_3y_2} & P_{y_3y_3} \end{bmatrix} \longrightarrow \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_3} \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_3} \\ P_{y_3x} & P_{y_3y_1} & P_{y_3y_3} \end{bmatrix}. \tag{2.62}$$

3. Experimental Results

The proposed method has been implemented using MATLAB in order to test its performance. First, several video sequences were acquired using two different low cost cameras. Later, the algorithm was executed off-line using those video sequences as input signals. In experiments, the following parameter values have been used: variances for linear and angular velocity, respectively, $\sigma_V = 4\,\mathrm{m/s}^2$, $\sigma_\Omega = 4\,\mathrm{m/s}^2$, noise variances $\sigma_u = \sigma_v = 1$ pixel, minimum base-line $b_{\min} = 15\,\mathrm{cm}$ and minimum parallax angle $\alpha_{\min} = 5°$.

In the first series of experiments, a Microsoft LifeCam Studio Webcam was used. This low cost camera has an USB interface and a wide angle lens. It is capable of acquiring HD color video, but in the present experiments gray level video sequences, with a resolution of 424 × 240 pixels at 30 frames per second, were used.

Figure 7 shows the experimental results for a video sequence of 790 frames acquired in a desktop environment with the LifeCam Studio webcam. In this case, the camera was moved from the left to the right following a trajectory similar to the curved front edge of the desktop (see Figure 9, left side). A black paper sheet was used as the initial metric reference (Frame 1, Figure 7). It is important to remark that the initial metric reference determines the origin of the world reference. The initial camera position and orientation respect to the coordinates of the world reference frame are determined according Section 2.2 explanations. Some candidate points were detected from the first frame; however, the first feature was not added to the map until Frame 111. In this frame, two new candidate points were detected (in red), and two candidate points were tracked (in blue). For simplicity, 3σ feature depth uncertainty is indicated by red 3D lines in plots illustrating the outputs of the proposed method (central and right columns in Figure 7). The features are located over these red lines. Of course, full geometric uncertainties should be represented by 3D ellipsoids. In frame 203 the first features, corresponding to the initial metric reference, have been left behind by the camera movement. Other features, corresponding to a painting placed approximately in the same plane that the initial metric reference, have been initialized. In frame 533 several features, corresponding to two different PC screens, have been added to the map. Weak image features, due for example to brightness over reflecting surfaces or changing video in the PC screens, are implicitly

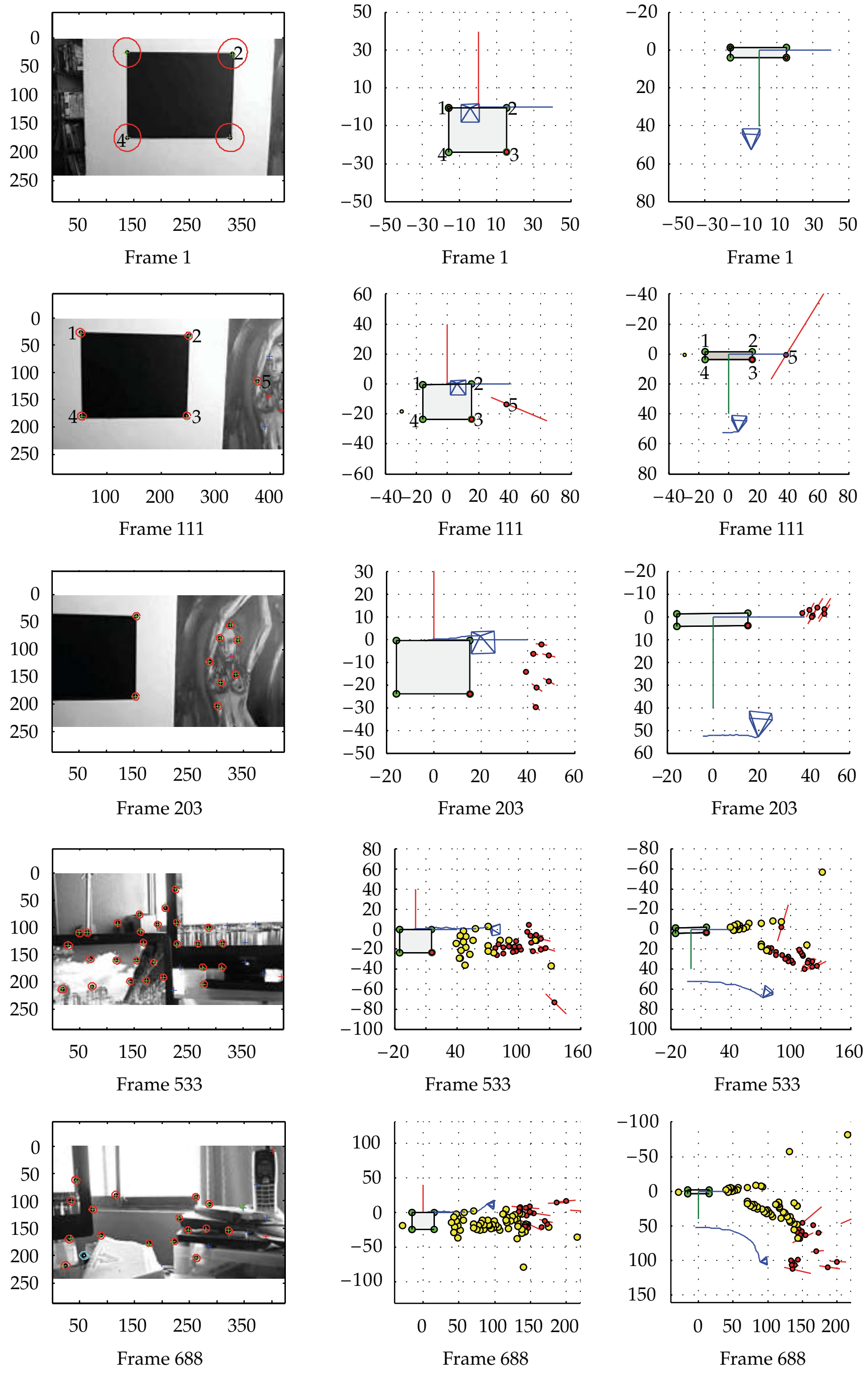


Figure 7: Input images (left column), camera trajectory (X-Y view; central column), and map estimations (X-Z view; right column) for a video sequence of 790 frames acquired in a desktop environment. Frames 1, 111, 203, 533, and 688 are displayed.

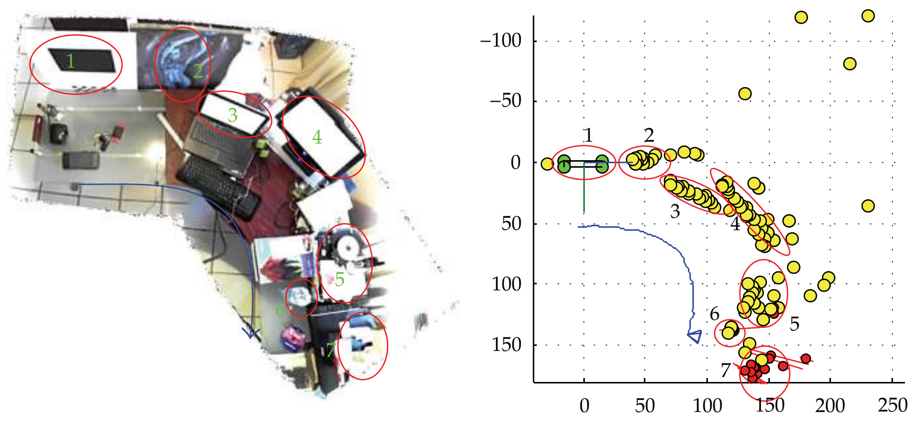


Figure 8: Composite top-view picture for a desktop environment (left plot). Corresponding camera trajectory and map estimations in X-Z view after 790 frames (right plot). Observe that both, camera trajectory and scene structure, have been recovered in a satisfactory manner.

pruned by the image tracking process of candidates points. At frame 688, several features have been left behind by the camera movement and removed from the system (in yellow), while other features close to the range camera view, are maintained in the map (in red).

Figure 8 shows a top view of the experimental environment (left plot), and the final camera trajectory and map estimates, in an X-Z view at frame 790 (right plot). Several objects have been labeled in both plots for illustrating the reconstruction of the scene structure. In this experiment both, camera trajectory and scene structure, have been recovered in a satisfactory manner.

In the second series of experiments, a Fire-i Unibrain monochrome webcam was used. This low-cost camera has an IEEE1394 interface and interchangeable wide angle lens. Video sequences with a resolution of 320×240 pixels acquired at 30 frames per second were used in experiments. Figure 9 shows the experimental results for a video sequence containing about 1750 frames. This sequence has been recorded walking over a predefined cyclical trajectory inside a laboratory environment. At the end of the sequence, 390 features were initialized in the map, but only about 20 or 30 features in average were maintained in the system state allowing that the computational time per frame was stable along the sequence.

In this experiment, the only reference for recovering the metric scale is a computer screen (frame 3), that was left behind around frame 300 by the camera movement. In the frame 565, some features have been clearly removed from the map. Here a feature is removed if it cannot be detected and tracked in the next 20 frames. In the frame 777 the camera has completed its first turn (or cycle) to the defined trajectory. In the frame 1635 the camera has returned next to its initial position (note that the PC screen used as initial metric reference appears again). At this point the discrepancy between the estimated trajectory and the real trajectory is very perceptible (the real path followed is like a long rectangle with rounded edges). The biggest degeneration in the estimated trajectory happens when the camera is turning the curves, mainly in the second and third curve, because there are not far features to obtain enough information for the orientation. Without underestimating the importance of the effects produced by the drift in the trajectory estimations, it can be appreciated in Figure 10 that the 3D structure of the environment has been recovered reasonably well.

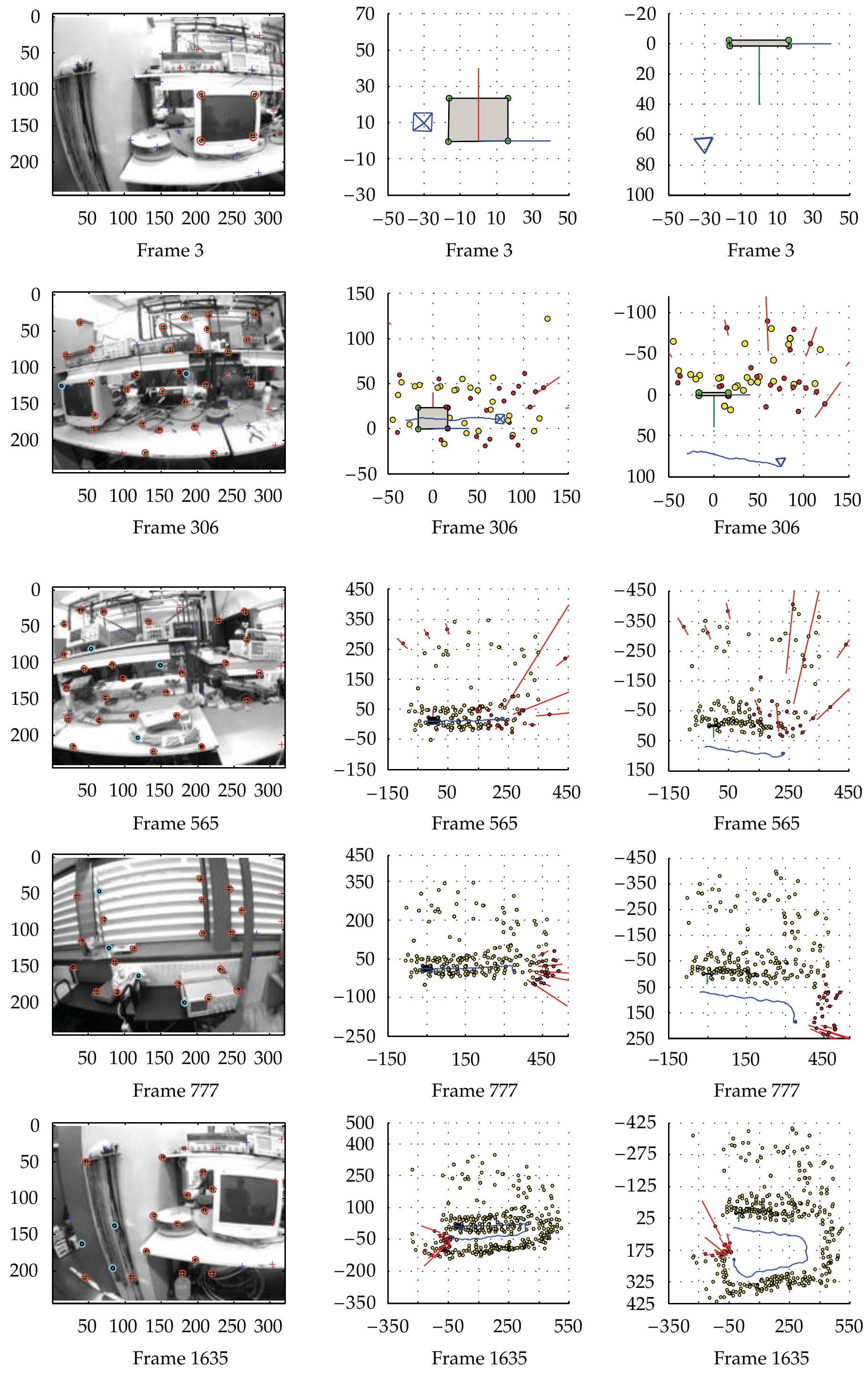


Figure 9: Input images (left column), camera trajectory (X-Y view; central column) and map estimations (X-Z view; right column) for a video sequence containing about 1750 frames recorded in a laboratory environment. Frames 3, 306, 565, 777, and 1635 are displayed.

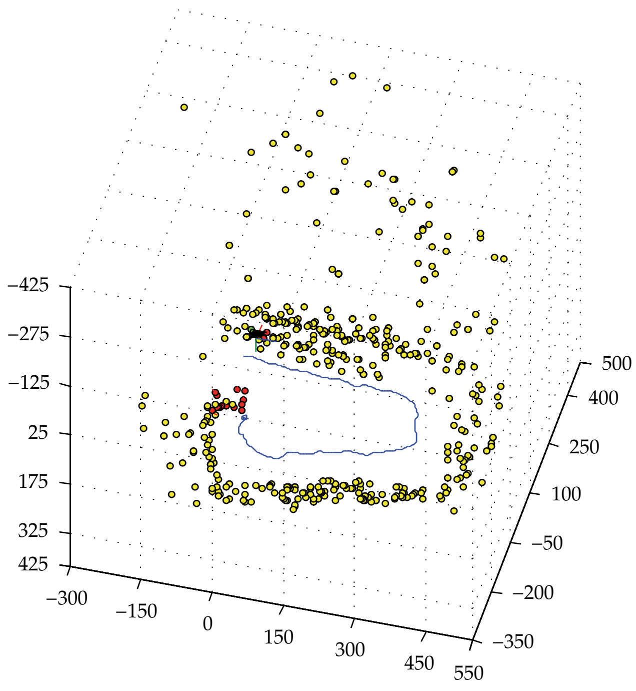


Figure 10: 3D view corresponding to the camera trajectory and map estimates after 1750 frames for the experiment in a laboratory environment shown in Figure 8.

4. Conclusions

In this paper a method for addressing the problem of 3D visual SLAM in a context of visual odometry using a single free-moving camera has been presented.

Depth information cannot be obtained in a single measurement when a bearing sensor (i.e., monocular camera) is used. Therefore, in order to incorporate new features to the map, special techniques for feature system initialization are needed in order to enable the use of bearing sensors in SLAM systems. In this work a method called delayed inverse-depth feature initialization is proposed to incorporate new features to the stochastic map from the bearing measurements of the camera. The proposed method is based in the inverse depth parameterization and implements, by means of a delay, a stochastic technique of triangulation in order to define a hypothesis of initial depth for the features.

In SLAM methods based on the extended Kalman filter is difficult to maintain the frame rate operation when the number of features in the system state increases. Old features have to be removed from the state in order to maintain a stable computational cost per frame. Obviously, if features are removed then previous mapped areas cannot be recognized in the future. However, a modified monocular SLAM method to maintain stable computational operation can be viewed as a complex real-time virtual sensor, which provides appearance-based sensing and emulates typical sensors (i.e., a laser) for range measurements and encoders for dead reckoning (visual odometry).

The experimental results, with real data, show that although a small and single metric reference is used for recovering the world scale, the metric degeneration quantification at the end of the trajectory is low. In the experiments carried out, the camera trajectory and scene structure have been recovered in a satisfactory manner. The results in the trajectory are not error-free though they are more likely to appear in curves when distant features do not exist.

Even though this work is concerned with the approach of monocular SLAM as a virtual sensor (e.g., in the context of visual odometry), the effectiveness of the proposed method has been also verified in a more general SLAM context. A real time implementation of the proposed method [37], fully adapted as a virtual sensor, provides a rich sensory input to a decoupled (Global) SLAM process. This Global SLAM is used for loop closing tasks and therefore for building and maintaining the global map and final camera pose.

Appendix

In this appendix some transformations, repeatedly used along the paper, are included.

A quaternion q can be computed from a rotation vector ω by the following:

$$q(\omega) = \left[\cos\left(\frac{\|\omega\|}{2}\right) \ \sin\left(\frac{\|\omega\|}{2}\right) \ \left[\frac{\omega}{\|\omega\|}\right]\right]^T. \tag{A.1}$$

A quaternion q can be computed form a rotation matrix R by the following:

$$q\left(R_n^b\right) = \begin{bmatrix} q_1 \\ \dfrac{R_n^b(3,2) - R_n^b(2,3)}{4q_1} \\ \dfrac{R_n^b(1,3) - R_n^b(3,1)}{4q_1} \\ \dfrac{R_n^b(2,1) - R_n^b(1,2)}{4q_1} \end{bmatrix}, \quad q_1 = \sqrt{1 + R_n^b(1,1) + R_n^b(2,2) + R_n^b(3,3)}. \tag{A.2}$$

A rotation matrix R can be computed from a quaternion q by the following:

$$R = \begin{bmatrix} \left(q_1^2 + q_2^2 - q_3^2 - q_4^2\right) & 2\left(q_2q_3 - q_1q_4\right) & 2\left(q_1q_3 + q_2q_4\right) \\ 2\left(q_2q_3 + q_1q_4\right) & \left(q_1^2 + q_2^2 - q_3^2 - q_4^2\right) & 2\left(q_3q_4 - q_1q_2\right) \\ 2\left(q_2q_4 - q_1q_3\right) & 2\left(q_1q_2 + q_3q_4\right) & \left(q_1^2 + q_2^2 - q_3^2 - q_4^2\right) \end{bmatrix}. \tag{A.3}$$

Acknowledgments

This work has been funded by the Spanish Ministry of Science Project DPI2010-17112.

References

[1] F. Auat, C. De la Cruz, R. Carelli, and T. Bastos, "Navegación autónoma asistida basada en SLAM para una silla de ruedas robotizada en entornos restringidos," *Revista Iberoamericana de Automática e Informática Industrial*, vol. 8, pp. 81–92, 2011.

[2] R. Vázquez-Martín, P. Núñez, A. Bandera, and F. Sandoval, "Curvature-based environment description for robot navigation using laser range sensors," *Sensors*, vol. 9, no. 8, pp. 5894–5918, 2009.

[3] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[4] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping: part II,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[5] A. J. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *Proceedings of the 19th IEEE International Conference on Computer Vision (ICCV ’03)*, vol. 2, pp. 1403–1410, October 2003.

[6] L. Matthies, T. Kanade, and R. Szeliski, “Kalman filter-based algorithms for estimating depth from image sequences,” *International Journal of Computer Vision*, vol. 3, no. 3, pp. 209–238, 1989.

[7] D. B. Gennery, “Visual tracking of known three-dimensional objects,” *International Journal of Computer Vision*, vol. 7, no. 3, pp. 243–270, 1992.

[8] A. Azarbayejani, B. Horowitz, and A. Pentland, “Recursive estimation of structure and motion using relative orientation constraints,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 294–299, June 1993.

[9] H. Jin, P. Favaro, and S. Soatto, “A semi-direct approach to structure from motion,” *Visual Computer*, vol. 19, no. 6, pp. 377–394, 2003.

[10] A. W. Fitzgibbon and A. Zisserman, “Automatic camera recovery for closed or open image sequences,” in *Proceedings of the European Conference on Computer Vision (ECCV ’98)*, vol. 1, pp. 311–326, 1998.

[11] Y. K. Yu, K. H. Wong, S. H. Or, and M. M. Y. Chang, “Robust 3-D motion tracking from stereo images: a model-less method,” *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 3, pp. 622–630, 2008.

[12] H. Deans and M. Martial, “Experimental comparison of techniques for localization and mapping using a bearing-only sensor,” in *Proceedings of the 7th International Symposium on Experimental Robotic (ISER ’00)*, December 2000.

[13] S. Strelow and D. Sanjiv, “Online motion estimation from image and inertial measurements,” in *Proceedings of the Workshop on Integration of Vision and Inertial Sensors (INERVIS ’03)*, June 2003.

[14] T. Bailey, “Constrained initialisation for bearing-only SLAM,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA ’03)*, pp. 1966–1971, September 2003.

[15] N. M. Kwok and G. Dissanayake, “An efficient multiple hypothesis filter for bearing-only SLAM,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS ’04)*, vol. 1, pp. 736–741, October 2004.

[16] N. M. Kwok, G. Dissanayake, and Q. P. Ha, “Bearing-only SLAM using a SPRT based gaussian sum filter,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1109–1114, April 2005.

[17] N. Peach, “Bearings-only tracking using a set of range-parameterized extended Kalman filters,” *IEE Proceedings*, vol. 142, no. 1, pp. 73–80, 1995.

[18] A. Davison, Y. Gonzalez, and N. Kita, “Real-time 3D SLAM with wide-angle vision,” in *Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV ’04)*, July 2004.

[19] H. Jin, P. Favaro, and S. Soatto, “Real-time 3-D motion and structure of point features: a front-end system for vision-based control and interaction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR ’00)*, pp. 778–779, June 2000.

[20] P. Smith, I. Reid, and A. Davison, “Real-time monocular SLAM with straight lines,” in *Proceedings of the British Machine Vision Conference*, BMVA Press, September 2006.

[21] P. Jensfelt, J. Folkesson, D. Kragic, and H. Christensen, “Exploiting distinguishable image features in robotics mapping and localization,” in *Proceedings of the European Robotics Symposium (EUROS ’06)*, Palermo, Italy, 2006.

[22] J. Solà, A. Monin, M. Devy, and T. Lemaire, “Undelayed initialization in bearing only SLAM,” in *Proceedings of the IEEE IRS/RSJ International Conference on Intelligent Robots and Systems (IROS ’05)*, pp. 2751–2756, August 2005.

[23] T. Lemaire, S. Lacroix, and J. Solà, “A practical 3D bearing-only SLAM algorithm,” in *Proceedings of the IEEE IRS/RSJ International Conference on Intelligent Robots and Systems (IROS ’05)*, pp. 2757–2762, August 2005.

[24] E. Eade and T. Drummond, “Scalable monocular SLAM,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR ’06)*, pp. 469–476, June 2006.

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: a factored solution to the simultaneous localization and mapping problem,” in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI ’02)*, pp. 593–598, Menlo Park, Calif, USA, August 2002.

[26] J. M. M. Montiel, J. Civera, and A. Davison, “Unified inverse depth parametrization for monocular SLAM,” in *Proceedings of the Robotics: Science and Systems Conference*, August 2006.
[27] R. Munguia and A. Grau, “Monocular SLAM for visual odometry,” in *Proceedings of the IEEE International Symposium on Intelligent Signal Processing (WISP ’07)*, Alcala de Henares, Spain, October 2007.
[28] R. Munguia and A. Grau, “Delayed inverse depth monocular SLAM,” in *Proceedings of the 17th World Congress, International Federation of Automatic Control (IFAC ’08)*, Seoul, Korea, July 2008.
[29] M. D. Shuster, “Survey of attitude representations,” *The Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, 1993.
[30] C. Chatterjee and V. P. Roychowdhury, “Algorithms for coplanar camera calibration,” *Machine Vision and Applications*, vol. 12, no. 2, pp. 84–97, 2000.
[31] S. Ganapathy, “Decomposition of Transformation Matrices for Robot Vision,” in *Proceedings of the International Conference on Robotics and Automation*, pp. 130–139, 1984.
[32] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, “MFm: 3-D motion from 2-D motion causally integrated over time,” in *Proceedings of the European Conference on Computer Vision (ECCV ’00)*, 2000.
[33] A. J. Davison and D. W. Murray, “Mobile robot localisation using active vision,” in *Proceedings of the 5th European Conference on Computer Vision (ECCV ’98)*, vol. 2, pp. 809–825, Freiburg, Germany, 1998.
[34] J. Neira and J. D. Tardós, “Data association in stochastic mapping using the joint compatibility test,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, 2001.
[35] T. Baile, *Mobile robot localisation and mapping in extensive outdoor environments [Ph.D. thesis]*, University of Sydney, Australian Centre for Field Robotics, 2002.
[36] P. Favaro, *Stima del moto e della struttura della scena tramite visione dinamica [M.S. thesis]*, University of Padua, 1998.
[37] R. Munguía and A. Grau, “Closing loops with a virtual sensor based on monocular SLAM,” *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 8, pp. 2377–2384, 2009.