RT-SLAM: a generic and real-time visual SLAM implementation

Cyril Roussillon[1,2,3], Aurélien Gonzalez[1,2],
Joan Solà[1,2,4], Jean-Marie Codol[1,2,5], Nicolas Mansard[1,2],
Simon Lacroix[1,2], and Michel Devy[1,2]

{firstname.name}@laas.fr

[1] CNRS; LAAS; 7 avenue du colonel Roche; F-31077 Toulouse, France
[2] Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
[3] Funded by the Direction Générale de l'Armement (DGA), France
[4] Ictineu Submarins, Industria 12, 08980 St. Feliu de Llobregat, Barcelona, Catalonia
[5] NAV ON TIME, 42 avenue du Général De Croutte, 31100 Toulouse, France

Abstract. This article presents a new open-source C++ implementation to solve the SLAM problem, which is focused on genericity, versatility and high execution speed. It is based on an original object oriented architecture, that allows the combination of numerous sensors and landmark types, and the integration of various approaches proposed in the literature. The system capacities are illustrated by the presentation of an inertial/vision SLAM approach, for which several improvements over existing methods have been introduced, and that copes with very high dynamic motions. Results with a hand-held camera are presented.

1 Motivation

Progresses in image processing, the growth of available computing power and the proposition of approaches to deal with bearings-only observations have made visual SLAM very popular, particularly since the demonstration of a real-time implementation by Davison in 2003 [4]. The Extended Kalman Filter (EKF) is widely used to solve the SLAM estimation problem, but it has recently been challenged by global optimization methods that have shown superior precision for large scale SLAM. Yet EKF still has the advantage of simplicity and faster processing for problems of limited size [14], and can be combined with global optimization methods [6] to take the best of both worlds.

Monocular EKF-SLAM reached maturity in 2006 with solutions for initializing landmarks [7] [12] [9]. Various methods for landmark parametrization have been analyzed [10], and the literature now abounds with contributions to the problem.

This article presents RT-SLAM[1], a software framework aimed at fulfilling two essential requirements. The first one is the need for a generic, efficient and

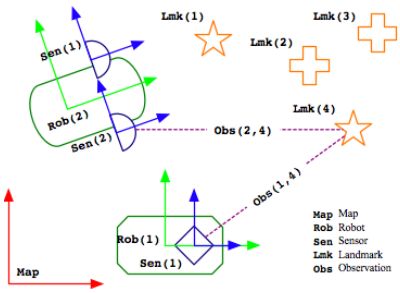
[1] RT-SLAM stands for "Real-Time SLAM"

flexible development tool, that allows to easily develop, evaluate and test various approaches or improvements. The second one is the need for practical solutions for live experiments on robots, for which localization and mapping require real-time execution and robustness. RT-SLAM is a C++ implementation based on Extended Kalman Filter (EKF) that allows to easily change robot, sensor and landmark models. It runs up to 60 Hz with 640×480 images, withstanding highly dynamic motions, which is required for instance on humanoid or high speed all terrain robots. RT-SLAM is available as *open source* software at `http://rtslam.openrobots.org`.

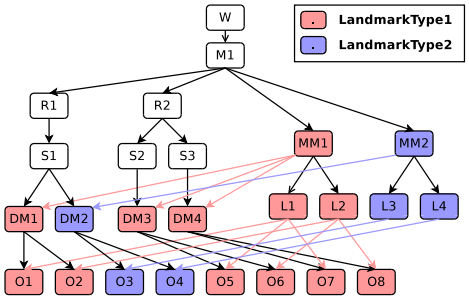
Section 2 details the architecture of RT-SLAM and section 3 presents some of the techniques currently integrated within, to define an efficient inertial/visual SLAM approach. Section 4 analyzes results obtained with a hand-held system assessed thanks to ground truth, and section 5 concludes the article by a presentation of prospects.

2 Overall Architecture

Fig. 1 presents the main objects defined in RT-SLAM. They encompass the basic concepts of a SLAM solution: the *world* or environment contains *maps*; maps contain *robots* and *landmarks*; robots have *sensors*; sensors make *observations* of landmarks. Each of these objects is abstract and can have different implementations. They can also contain other objects that may themselves be generic.



(a) Main objects in a SLAM context. Different robots `Rob` have several sensors `Sen`, providing observations `Obs` of landmarks `Lmk`. States of robots, sensors and landmarks are stored in the stochastic map `Map`. There is one observation per sensor-landmark pair.



(b) Objects hierarchy in RT-SLAM. Each individual map `M` in the world `W` contains robots `R` and landmarks `L`. A robot has sensors `S`, and an observation `O` is created for every pair of sensor and landmark. In order to allow full genericity, map managers `MM` and data managers `DM` are introduced.

Fig. 1. The main objects in RT-SLAM

Map. The maps contain an optimization or estimation engine: for now RT-SLAM uses a standard formulation of EKF-SLAM. Since this solution is very well documented in the literature [5], it is not detailed in depth here. Indirect indexing within Boost's ublas C++ structures is intensively used to exploit the sparsity of the problem and the symmetry of the covariances matrices.

Robot. Robots can be of different type according to the way their state is represented and their prediction model. The latter can be either a simple kinematic model (constant velocity, constant acceleration, ...) or a proprioceptive sensor (odometric, inertial, ...), as illustrated section 3.3. The proprioceptive sensor is an example of generic object contained in robot objects, as different hardware can provide the same function.

Sensor. Similarly to robots, sensors can also have different models (perspective camera, panoramic catadioptric camera, laser, ...), and contain a generic exteroceptive sensor hardware object (firewire camera, USB camera, ...). In addition, as sensors belong to the map, their state can be estimated: this opens the possibility for estimating other parameters such as extrinsic calibration, time delays, biases and gain errors, and the like.

Landmark. Landmarks can be of different type (points, lines, planes, ...), and each type can have different state parametrization (Euclidean point, inverse depth point, ...). Moreover the parametrization of a landmark can change over time, as explained section 3.2. A landmark also contains a descriptor used for data association, which is a dual description to the state representation.

As shown Fig. 1(a), it is worth noticing that landmark objects are common to the different sensors, all of them being able to observe the same landmark (provided they have compatible descriptors for this landmark of course). This allows to greatly improve the observability of landmarks compared to a system where the sensors are strictly independent. In the particular case of two cameras for instance, landmarks can be used even if they are only visible from one camera or if they are too far away for a stereovision process to observe their depth (this process was introduced in [11] as *BiCam* SLAM).

Observation. In RT-SLAM, the notion of observation plays a predominant role. An observation is a real object containing both methods and data. One observation is instantiated for every sensor-landmark pair, regardless of the sensor having actually observed the landmark or not, and has the same lifetime as the associated landmark. The methods it contains are the conventional projection and back-projection models (that depend on the associated sensor and landmark models), while the stored data consist of results and intermediary variables such as Jacobian matrices, predicted and observed appearances, innovations, event counters and others, that allow to greatly simplify and optimize computations.

Managers. In order to achieve full genericity *wrt* landmark types, in particular to allow the concurrent use of different landmark types for one sensor, two

different manager objects are added: *data manager* and *map manager*. Their implementations define a given management strategy, while their instantiations are dedicated to a certain landmark type. The data manager processes the sensors raw data, providing observations of the external landmarks. For this purpose, it exploits some raw data processors (for feature detection and tracking), and decides which observations are to be corrected and in which order, according to the quantity of information they bring and their quality. For example it can apply an active search strategy and try to eliminate outliers as described in section 3.1. The map manager keeps the map clean, with relevant information, and at a manageable size, by removing landmarks according to their quality and the given policy (*e.g.* visual odometry where landmarks are discarded once they are not observed, or multimap slam where maps are "closed" according to given criteria). These managers communicate together: for example, the data manager may ask the map manager if there is enough space in the map to start a new initialization process, and to allocate the appropriate space for the new landmark.

3 Inertial/visual SLAM within RT-SLAM

3.1 Active search and one-point RANSAC

The strategy currently implemented in RT-SLAM's data manager to deal with observations is an astute combination of *active search* [5] and outliers rejection using *one-point RANSAC* [3].

The goal of active search is to minimize the quantity of raw data processing by constraining the search in the area where the landmarks are likely to be found. Observations outside of this 3σ observation uncertainty ellipse would be anyway considered incompatible with the filter and ignored by the *gating* process. In addition active search gives the possibility to decide anytime to stop matching and updating landmarks with the current available data, thus enabling *hard real-time* constraints. We extended the active search strategy to landmark initialization: each sensor strives to maintain features in its whole field of view using a randomly moving grid of fixed size, and feature detection is limited to empty cells of the grid. Furthermore the good repartition of features in the field of view ensures a better observability of the motions.

Outliers can come from matching errors in raw data or mobile objects in the scene. Gating is not always discriminative enough to eliminate them, particularly right after the prediction step when the observation uncertainty ellipses can be quite large – unfortunately at this time the filter is very sensitive to faulty corrections because it can mistakenly make all the following observations incompatible. To prevent faulty observations, outliers are rejected using a one-point RANSAC process. It is a modification of RANSAC, that uses the Kalman filter to obtain a whole model with less points than otherwise needed, and provides a set of *strongly compatible* observations that are then readily corrected. Contrary to [3] where data association is assumed given when applying the algorithm, we do the data association along with the one-point RANSAC process: this allows to look for features in the very small strongly compatible area rather than the

whole observation uncertainty ellipse, and to save additional time for raw data processing.

3.2 Landmark parametrizations and reparametrization

In order to solve the problem of adding to the EKF a point with unknown distance and whose uncertainty cannot be represented by a Gaussian distribution, point landmarks parametrizations and initialization strategies for monocular EKF-SLAM have been well studied [4] [8] [9]. The solutions now widely accepted are undelayed initialization techniques with *inverse depth* parametrization. Anchored Homogeneous Point [10] parametrization is currently used in RT-SLAM.

The drawback of inverse depth parametrizations is that they describe a landmark by at least 6 variables in the stochastic map, compared to only 3 for an Euclidean point $(x\ y\ z)^T$. Memory and temporal complexity being quadratic with the map size for EKF, there is a factor of 4 to save in time and memory by *reparametrizing* landmarks that have converged enough [2]. The map manager uses the linearity criterion proposed in [9] to control this process.

3.3 Motion prediction

The easiest solution for EKF-SLAM is to use a robot kinematic model to do the filter prediction, such as a constant velocity model:

$$\mathcal{R} = (\mathbf{p}\ \mathbf{q}\ \mathbf{v}\ \mathbf{w})^T$$

where $\mathbf{p}$ and $\mathbf{q}$ are respectively the position and quaternion orientation of the robot, and $\mathbf{v}$ and $\mathbf{w}$ are its linear and angular velocities.

The advantage of such a model is that it does not require complicated hardware setup (only a simple camera), but its strong limitation is that the scale factor is not observable. A second camera with a known baseline can provide a proper scale factor, but one can also use a proprioceptive sensor for the prediction step. Furthermore it usually provides a far better prediction with smaller uncertainties than a simple kinematic model, which brings several benefits:

1. search areas for matching are smaller, so processing is faster,
2. linearization points are more accurate, so SLAM precision is better, or one can reduce the framerate to decrease CPU load for equivalent quality,
3. it allows to withstand high motion dynamics.

In the case of an Inertial Measurement Unit (IMU), the robot state is then:

$$\mathcal{R} = (\mathbf{p}\ \mathbf{q}\ \mathbf{v}\ \mathbf{a_b}\ \mathbf{w_b}\ \mathbf{g})^T$$

where $\mathbf{a_b}$ and $\mathbf{w_b}$ are the accelerometers and gyrometers biases, and $\mathbf{g}$ the 3D gravity vector. Indeed it is better for linearity reasons to estimate the direction of $\mathbf{g}$ rather than constraining the robot orientation to enforce $\mathbf{g}$ to be vertical.

A special care has to be taken for the conversion of the noise from continous time (provided by the manufacturer in the sensor's datasheet) to discrete time. As the perturbation is continuous white noise, the variance of the discrete noise grows linearly with the integration period.

3.4 Image processing

Point extraction. Point extraction is based on Harris detector with several optimizations. Some of them are approximations: a minimal derivative mask $[-1, 0, 1]$ is used, as well as a square and constant convolution mask, in order to minimize operations. This allows the use of *integral images* [15] to efficiently compute the convolutions. Additional optimizations are related to active search (section 3.1): only one new feature is searched in a small region of interest, which eliminates the costly steps of thresholding and sub-maxima suppression.

Point matching. Point matching is based on Zero-mean Normalized Cross Correlation (ZNCC), also with several optimizations. Integral images are used to compute means and variances, and a hierarchical search is made (two searches at half and full resolution are sufficient). We also implemented *bounded partial correlation* [13] in order to interrupt the correlation score computation when there is no more hope to obtain a better score than the threshold or the best one up to now. To be robust to viewpoint changes and to track landmarks longer, tracking is made by comparing the initial appearance of the landmark with its current predicted appearance [5].

4 Results

Fig. 2 shows the hardware setup that has been used for the experiments. It is composed of a firewire camera rigidly tied to an IMU, on which four motion capture markers used to measure the ground truth are fixed.

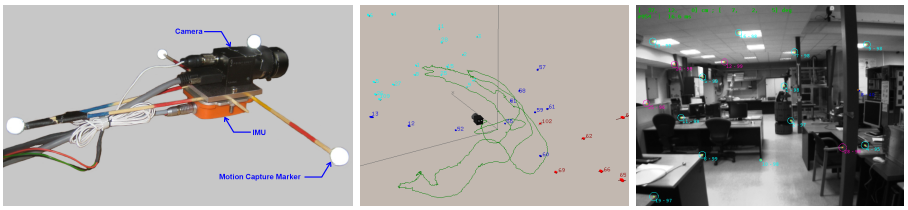


Fig. 2. The experimental setup composed of a Flea2 camera and an XSens MTi IMU, and screen captures of the 3D and 2D display of RT-SLAM.

Two different sequences are used, referred to as *low dynamic* and *high dynamic* sequences. Both were acquired indoor with artificial lights only, with an image framerate of 50 Hz synchronized to the inertial data rate of 100 Hz. The motion capture markers are localized with a precision of approximately 1 mm, so the ground truth has a precision of $\sigma_{xyz} = 1\,\text{mm}$ in position and $\sigma_{wpr} = 0.57°$ in angle (baseline of 20 cm).

Movies illustrating a run of every experiment are provided at:
`http://rtslam.openrobots.org/Material/ICVS2011`.

4.1 Constant velocity model

The inertial data is here not used, and the prediction is made according to a constant velocity model. Fig. 3 presents the estimated trajectory and the ground truth for the *low dynamic* sequence, and Fig. 4 shows the absolute errors for the same run.

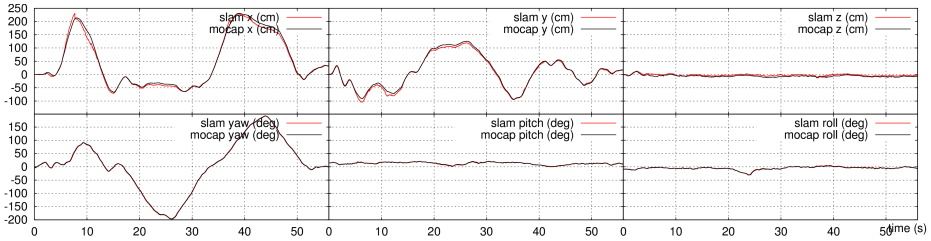


Fig. 3. Illustration of low dynamic trajectory, constant velocity SLAM (with scale factor of 2.05 manually corrected). Estimated trajectory parameters and ground truth, as a function of time (in seconds).

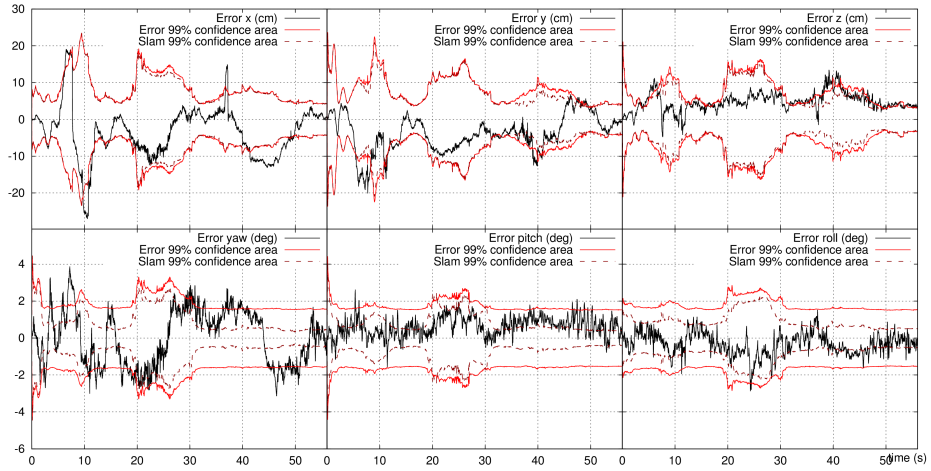


Fig. 4. Errors of the estimated parameters of the trajectory shown Fig. 3. The 99% confidence area corresponds to 2.57σ bounds. The SLAM 99% confidence area, that does not include ground truth uncertainty, is also shown.

Besides the global scale factor which is manually corrected, the camera trajectory is properly estimated. The movie shows that loops are correctly closed, even after a full revolution.

The position error raises up to 10 cm after quick rotations of the camera: this is due to a slight *drift* of the scale factor caused by the integration of newer landmarks (when closing loops, the scale factor is restimated closer to its initial value). Also, uncertainty ellipses remain relatively large throughout the sequence: these issues are solved by the use of an IMU to predict the motions.

4.2 Inertial/visual SLAM

Fig. 5 shows the trajectory estimated with the *high dynamic* sequence, and Fig. 6 and 7 show the behavior of inertial SLAM. Here, all of the 6 degrees of freedom are successively excited, then y and yaw are excited with extreme dynamics: the yaw angular velocity goes up to 400 deg/s, the rate of change of angular velocity exceeds 3,000 deg/s^2, and accelerations reach $3\,g$. It is interesting to note that the time when SLAM diverges (around $t = 65$ s) corresponds to motions for which the angular velocity exceeds the limit of the IMU (300 deg/s) and where its output is saturated.

The IMU now allows to observe the scale factor, and at the same time reduces the observation uncertainty ellipses and thus eases the active search procedure. Conversely, the divergence of the SLAM process at the end of the sequence illustrates what happens when vision stops stabilizing the IMU: it quickly diverges because the biases and the gravity cannot be estimated anymore.

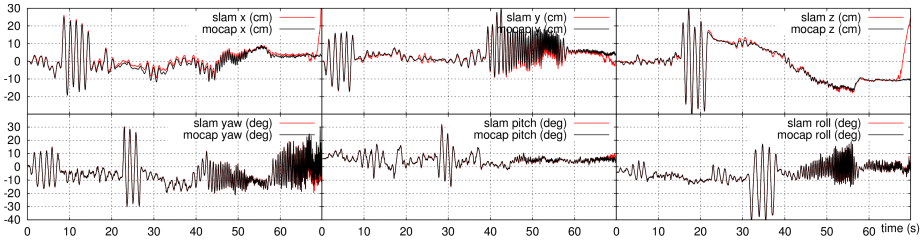


Fig. 5. Illustration of high dynamic trajectory, inertial/visual SLAM. Estimated trajectory parameters and ground truth, as a function of time (in seconds).

5 Outlook

We have presented a complete SLAM architecture whose genericity and performance make it both a useful experimentation tool and efficient solution for robot localization. The following extensions are currently being developed: using a second camera to improve landmarks observability, a multimap approach to cope with large scale trajectories and multirobot collaborative localization, and the use of line landmarks complementarily to points to provide more meaningful maps and to ease map matching for loop closure.

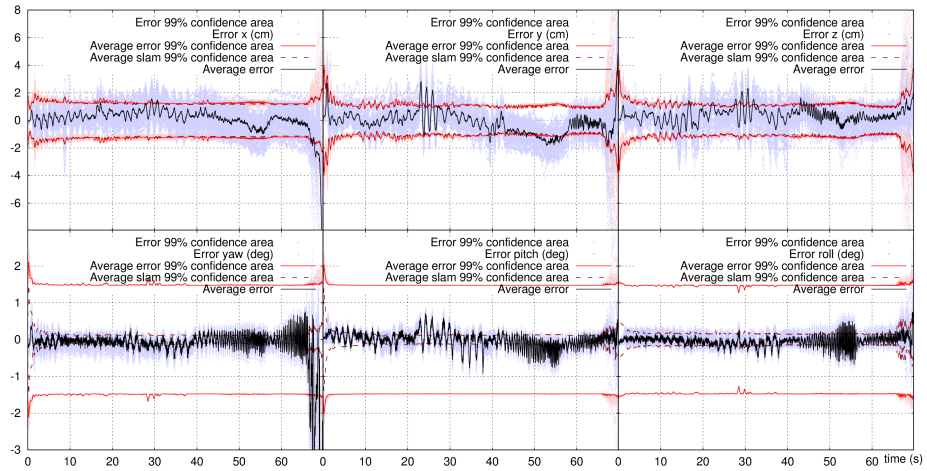


Fig. 6. Errors of inertial/visual SLAM estimated over 100 runs on the dynamic sequence, as a function of time (in seconds) – the difference between each run is due to the randomized landmark selection, see section 3.1. All the runs remain in the neighborhood of the 99% confidence area. The angular ground truth uncertainty is not precisely known: its theoretical value is both overestimated and predominant over SLAM precision in such a small area.

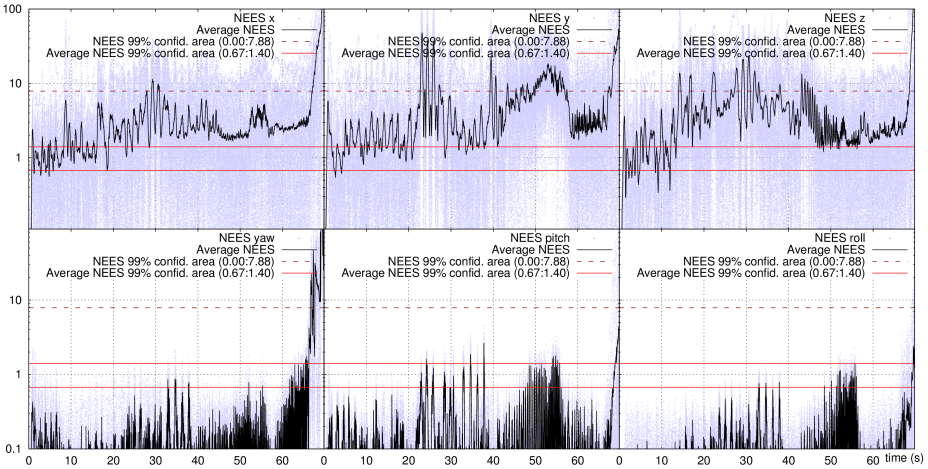


Fig. 7. Inertial/visual SLAM NEES [1] over 100 runs on the dynamic sequence. The average NEES is quickly out of the corresponding 99% confidence area, but as the 100 runs were made on the same sequence they are not independant and the average NEES should rather be compared to the simple NEES confidence area. The filter appears to be very conservative with angles but this is due to the overestimated ground truth uncertainty as explained in Fig. 6.

The architecture of RT-SLAM allows to easily integrate such developments, and also to consider additional motion sensors: to increase the robustness of the system, it is indeed essential to consider the various sensors usually found on board a robot (odometry, gyrometers, GPS). Eventually, it would be interesting to make RT-SLAM completely generic *wrt* the estimation engine as well, in order to be able to use global optimization techniques in place of filtering.

References

1. T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562 –3568, oct. 2006.
2. J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse Depth to Depth Conversion for Monocular SLAM. *Robotics and Automation, 2007 IEEE International Conference on*, pages 2778–2783, April 2007.
3. J. Civera, O.G. Grasa, A.J. Davison, and J.M.M. Montiel. 1-point ransac for ekf-based structure from motion. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3498 –3504, October 2009.
4. Andrew Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *IEEE International Conference on Computer Vision*, pages 1403–1410, October 2003.
5. Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:1052–1067, June 2007.
6. C. Estrada, J. Neira, and J.D. Tardos. Hierarchical slam: Real-time accurate mapping of large environments. *Robotics, IEEE Transactions on*, 21(4):588 – 596, aug. 2005.
7. N.M. Kwok and G. Dissanayake. An efficient multiple hypothesis filter for bearing-only slam. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 736 – 741 vol.1, 2004.
8. T. Lemaire, S. Lacroix, and J. Sola. A practical 3d bearing-only slam algorithm. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2449 – 2454, aug. 2005.
9. J. M. M. Montiel. Unified inverse depth parametrization for monocular slam. In *in Proceedings of Robotics: Science and Systems*, pages 16–19, 2006.
10. J. Sola. Consistency of the monocular ekf-slam algorithm for three different landmark parametrizations. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3513 –3518, May 2010.
11. J. Sola, A. Monin, and M. Devy. Bicamslam: Two times mono is more than stereo. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4795 –4800, 2007.
12. J. Sola, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only slam. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2499 – 2504, 2005.
13. Luigi Di Stefano, Stefano Mattoccia, and Federico Tombari. Zncc-based template matching using bounded partial correlation. *Pattern Recognition Letters*, 26(14):2129 – 2134, 2005.
14. H. Strasdat, J.M.M. Montiel, and A.J. Davison. Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657 –2664, May 2010.

15. Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *in Proc. IEEE Conf. Comput. Vis. Pattern Recog*, pages 511–518, 2001.