The vSLAM Algorithm for Navigation in Natural Environments

Evolution Robotics, Inc.

Niklas Karlsson, Luis Goncalves, Mario E. Munich, and Paolo Pirjanian

Abstract

This article describes the *Visual Simultaneous Localization and Mapping* (vSLAM™) algorithm, a novel algorithm for *simultaneous localization and mapping* (SLAM). The algorithm is vision- and odometry-based, and enables low-cost navigation in cluttered and populated environments. The algorithm creates visual landmarks that are highly distinctive and that can be reliably detected, virtually eliminating the data association problem present in other landmark schemes. No initial map is required, and dynamic changes in the environment, such as lighting changes, moving objects, and/or people are gracefully handled by the algorithm. Typically, vSLAM recovers quickly from dramatic disturbances, such as "kidnapping".

Keywords: SLAM, Vision-based landmarks, SIFT-features, Landmark detection, Pose estimation, Particle filter, Kalman filter, Mixed proposal distribution.

1. Introduction

Simultaneous localization and mapping (SLAM) is one of the most fundamental, yet most challenging problems in mobile robotics. To achieve full autonomy a robot must possess the ability to explore its environment without user-intervention, build a reliable map, and localize itself in the map. In particular, if external beacons, or e.g. *global positioning sensor* (GPS) data, are unavailable, the robot must autonomously determine appropriate natural reference points (or landmarks) on which to build a map.

Only within the last 4-8 years has the research community made significant progress towards solutions to the SLAM problem. Such progress was enabled by the availability of better algorithms, as well as, better sensors. In particular, we should mention the introduction of probabilistic robotics [1] and the usage of the SICK™ *Laser Range Finder* (LRF) and as major milestones in the development of solutions to the SLAM problem.

Initially, the most promising algorithm for solving the SLAM problem was based on *Expectation Maximization* (EM) techniques [2]. The idea behind EM algorithms is to start with an initial guess of robot pose, followed by computing the most likely map. Next, the estimate of where the robot really is located is improved, based on the computed map. Using the improved estimate of robot pose, the previously computed map is made better, and so forth. Unfortunately, since the EM algorithm is not recursive, it is in most practical situations impossible to use in real-time. The excellent survey [3] describes other proposed SLAM solutions, as well as, the state-of-art of SLAM in year 2002.

The recently most promising SLAM algorithm was published in [4], where it was realized that if the localization and mapping problem is described as a problem of path estimation rather than pose estimation, then it can be

separated via a clever factorization.

The SICK LRF provides high resolution 2D distance scans, which makes it possible to reliably detect and localize corners and other edge features in the environment. However, there are at least three problems in using the SICK LRF: 1) It is difficult to reliably associate detected features with previously detected features; 2) the total number of features to maintain in a map is large, potentially causing a prohibitive computational overhead; and 3) it is difficult to extract and recognize corners and edge features in cluttered and highly populated environments.

The above three problems are shared by other traditional sensors, such as, the sonar and IR sensors. But these sensors have the additional problem of lower resolution and accuracy. To use the sonar or IR sensors in SLAM it is impossible to rely on a single sensor reading, but instead it is necessary to integrate sensor data over time to reliably detect e.g. corners or edges.

One way to address the above problems is to use a camera and computer vision based techniques to select appropriate reference points. This was until recently prohibitive because of the price of the camera, the big computational requirement, and the lack of appropriate algorithms. Today, however, cameras have become a commodity and, thanks to Moore's law, high-performing computers are available at low cost. Also, very powerful methods in computer vision [5] and probabilistic robotics now offer the necessary tools for vision-based SLAM algorithms.

In the past, the best vision algorithms were only able to detect and track very simple features (using e.g. the Harris corner detector), or match arbitrary images without providing location data. Tracking corners enabled the creation of rich 3-D structure, but once the corners were no longer viewed, they could never be found again. Object recognition based on correlation, templates, eigen-images, or other similar schemes provide image-based landmark recognition, but these schemes are not robust to lighting or viewpoint changes, and they also do not enable the estimation of a pose with respect to the landmark.

An early attempt of using vision in navigation of mobile robots was presented in [6]. However, this attempt addressed only localization, and not mapping, hence, does not qualify as a SLAM algorithm.

The recent development of SIFT-features [5] has enabled wide baseline tracking, thus enabling the use of virtually unique natural landmarks that can be identified after an arbitrary robot motion. These landmarks also allow relative robot pose estimates to be made. Various SLAM algorithms based on SIFT features have been proposed by the inventor of the SIFT-features, David Lowe, [7,8]. In his systems, a trinocular sensor is used to generate 3-D landmark points from a single robot pose. Each single 3-D point (with associated SIFT features in the three trinocular views) is a landmark, and the landmark generation and the mapping algorithm are tightly coupled.

The vSLAM algorithm, presented in this paper, has a SIFT-feature [5] based visual front-end. But, in contrast to previously proposed algorithm, the vSLAM system generates, detects, and estimates the relative pose to a landmark utilizing a single camera. Furthermore, the vSLAM algorithm utilizes the factorization in [4] to build and maintain the map of virtually unique visual landmarks. By using a localization scheme with a particle filter [9] and an adaptive mixed proposal distribution [10,11], vSLAM enables navigation with good accuracy in a large variety of real-world environments. The adaptive mixed proposal distribution also enables vSLAM to recover from “kidnapping” scenarios; i.e., situations where the robot is lifted up and moved without being notified.

The article is organized as follows. Section 2 formulates the problem solved by vSLAM. We list the sensors that are used in the algorithm, but we also highlight some of the challenges to solve the problem. Thereafter, Section 3 gives an overview of the vSLAM system. The intention is to show

the basic idea of vSLAM, but also to illustrate the flexibility of the system as a result of its modularity. In Section 4 the Visual Front-end is described. The procedure for creating landmarks and computing relative pose estimates is derived. Thereafter, in the short Section 5, the Pre-filter is presented. The Pre-filter provides a convenient way to tune the vSLAM system for use in applications with different desired trade-offs between accuracy and robustness. Section 6 describes how the map of visual landmarks is maintained and how the robot-pose is estimated relative this map. In Section 7 we show some experimental results of using vSLAM for mapping and localization in various environments. Finally, in Section 8 we draw some conclusions and make some final remarks.

2. Problem Formulation

Consider a mobile robot that must localize itself in a previously unknown environment. The objective is to choose a sensor configuration and an algorithm to process the sensor data, which accurately and robustly accomplish the localization in real-world environments. Assume it is not accepted to alter the environment by installing beacons or other external equipment. That is, the design choices only apply to the robot itself, and the robot must determine its location based only on data collected by the selected on-board sensors. Also, realize that since the environment is unknown, the robots must also map the environment.

Now, suppose the mobile robot is equipped with an odometry sensor providing dead reckoning data, and a single camera providing images of the environment. The odometry sensor may consist of standard wheel encoders attached to the driving wheels of a differential drive system.

The objective of the vSLAM system is to fuse image and odometry data in a way that enables robust map-building and localization. Being "robust" is key since the sensor data acquired from the mobile robot (odometry and images) contains plenty of difficult-to-model noise.

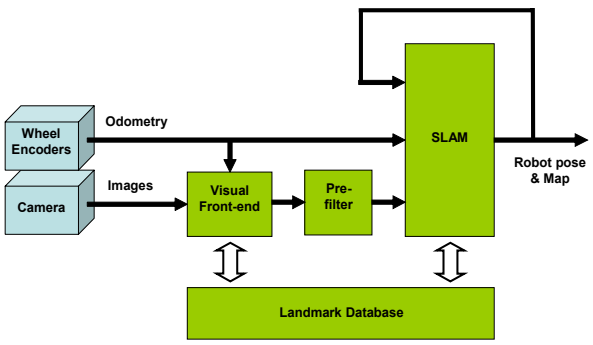
The odometry data is incremental, and therefore it will accumulate error over time. Furthermore, the odometry sensor can never be perfectly calibrated. But, since the robot may slip or may be lifted and moved, the odometry is also exposed to discrete events of dramatic errors.

The image data is challenging to process because of the existence of image blur, occlusions, limited image resolution, imperfect camera calibration, variable lighting conditions, limited processing power, etc. For example, if the robot is operating in a highly populated environment, the field of view of the camera is frequently filled by non-stationary objects, such as moving people that can not be used as reference points in a map. Also, the lighting conditions may change from very dark to light, even saturating the captured image.

The remainder of this article will describe the vSLAM algorithm, which is a system that addresses the above challenges and provides an accurate and robust way of doing localization and mapping in real-world environments.

3. Overview of vSLAM

From a system point of view, vSLAM can be described as in [Fig. 1]. The inputs to the system are odometry data and images, and the output is the robot pose and an abstract vSLAM map.



[Fig. 1] Block diagram of vSLAM

The basic components of the vSLAM map are the visual landmarks that are "created" along the path of the robot. A *visual landmark* is a collection of unique features extracted from an image. Each landmark is associated with a *landmark pose*, defined as the true robot pose (x, y, and θ) when the landmark was created.

In the first module of vSLAM, the *Visual Front-end*, the images are processed and compared with previously created landmarks. If a landmark is recognized, an estimate is computed on where the robot is located relative to where it was located when the landmark was created. Note that these relative estimates do not give information on where in the global map the robot is since that depends on where in the global map the landmark is located, which is unknown to the Visual Front-end. The relative pose estimates are throughout the paper called *visual measurements*. Visual measurements are described in detail in Section 4.2. If a landmark is not recognized, the module attempts to create a new visual landmark. The creation of new landmarks is described in detail in Section 4.1. If a landmark is created, some unique features (the SIFT-features) of the image corresponding to the landmark are saved for later recognition (the features are added to the *Landmark Database*).

If a visual landmark was recognized in the Visual Front-end and a visual measurement is computed, then the visual measurement is passed to the second module, the *Pre-filter*. The *Pre-filter*, assess the reliability of the measurement. If a measurement is considered unreliable, it is rejected and thrown away. It will thereafter not be used in any further processing. However, if it is accepted, it is used as an input to the *SLAM Back-end*.

The SLAM Back-end is a feedback system taking odometry data and relative pose measurements as inputs, to be fused and to generate and maintain a map of many landmarks. First, the robot pose is estimated based on the most up-to-date map and the two inputs. Then, the map is updated to reflect the new information. The SLAM Back-end associates a global landmark pose to the landmark and saves this pose in the Landmark Database. Thereafter, every time the landmark pose is updated, this pose is refined.

4. Visual Front-end

The inputs to the Visual Front-end are images with time stamps plus time-stamped odometry. The output depends on whether

- The front-end created a new landmark
- The front-end computed a visual measurement
- The front-end neither computed a visual measurement, nor created a landmark

If a landmark was created, a landmark ID is provided to be used as a reference for future landmark recognitions. Note that no landmark pose is provided by the front-end. The location of individual landmarks in a global map is estimated in the SLAM module.

If a visual measurement was computed, the output is a relative pose $y = [\Delta x, \Delta y, \Delta\theta]^T$ and a landmark ID n. The output indicates where landmark n is estimated to be located relative the current robot-pose. This relative pose is described in the standard, local robot coordinate system. The origin of this coordinate system is at the floor level, directly below the drive wheels' wheelbase. The positive x-axis extends forward in the direction the robot is facing. The positive y-axis extends to the left assuming you stand straight up and face along the positive x-axis.

To allow the Pre-filter to assess the reliability of the measurement, the following three statistics are also computed and attached to each measurement:

- The *Root Mean Square projection error* (ProjErrRMS) of the estimation. For a perfectly consistent measurement, this error should be zero.
- The *number of feature points* (NPoints) that have been recognized between the current image and the image

corresponding to a landmark. For a reliable estimation this number should be as large as possible.

- The estimated *Slope* of the robot, which under the assumption that the robot is operating on a leveled floor, should be approximately zero.

Remark: If multiple landmarks are recognized in an individual image, they may for simplicity be treated as separate measurements and be treated one after another. This is how they are treated in this paper. However, for improved performance it is possible to process the measurements corresponding to one image simultaneously.

4.1 Landmarks

The creation of a 3-D visual landmark is accomplished as follows:

1. Acquire 3 images while traveling approximately 20 cm between images.
2. Use the SIFT-based object recognition algorithm to find feature correspondences.
3. Solve for pose and structure of the three views.
4. Store the 3-D structure and the associated images and features.

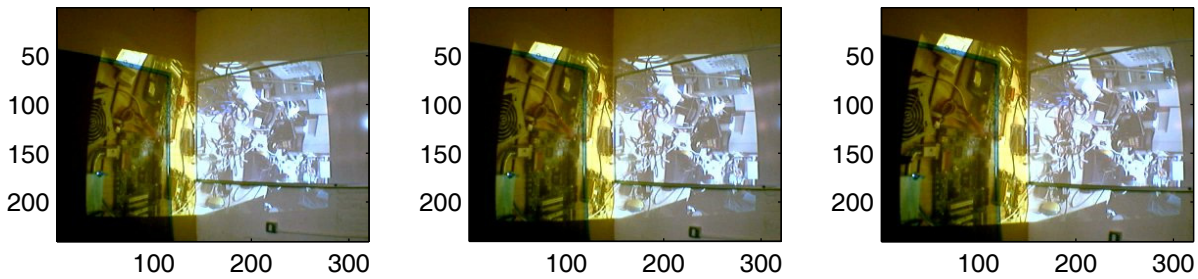
4.1.1 Acquiring Images

In most applications, there is only one camera on the robot (due to cost constraints), and the camera is facing forward (the most convenient orientation for other tasks, such as telepresence, teleoperation, and obstacle avoidance). This means that consecutive views typically consist of nearly pure translation along the optical axis of the camera. This is the most difficult case for which to attempt to do 3-D triangulation, since there is very little disparity between images. For this reason, we utilize three views in order to construct a 3-D landmark. With three views, three-way image correspondences can be detected more reliably, and the 3-D geometry computed more accurately. In typical usage, images taken 20 cm apart while the robot is moving forward are used to build landmarks. A baseline of 20 cm allows estimation of 3-D structure up to 5 m away with sufficient accuracy for subsequent pose estimation. [Fig. 2] shows an example of acquired images for landmark creation.

4.1.2 Finding Image Feature Correspondence

Given three consecutive views of the environment, the SIFT-based object recognition algorithm is used to find three-way correspondences amongst the views. This is done in the following way:

1. Find the correspondences between views 1 and 2, and between views 1 and 3 by training the object recognition algorithm on view 1 and finding all the matching features in view 2 and view 3, respectively.
2. Find the correspondences between views 2 and 3 by training the recognition algorithm on view 2 and

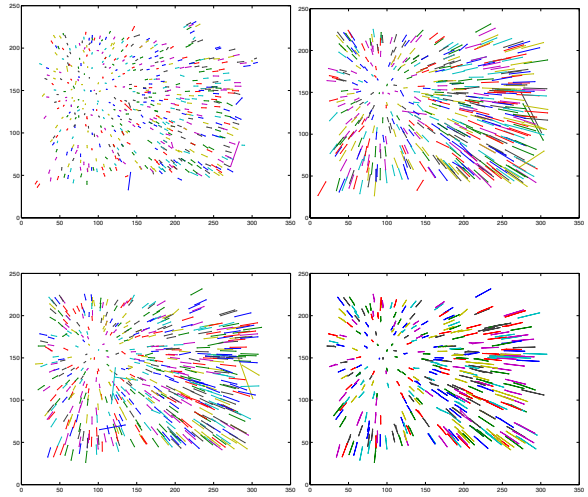


[Fig. 2] Three images used to generate a 3-D landmark. The robot is facing a corner of a room where a feature-rich image has been projected with an LCD projector (to illustrate the method in an easily interpretable structure). The robot moved forward 20 cm between image acquisitions, and the corner was 3.3 m away.

finding all the matching features in view 3.

3. Find the set of consistent three-way matches by, for each matching pair between views 1 and 2,
   - ○ Checking if the feature from view 1 exists in the list of correspondences of views 1 and 3.
   - ○ Checking if the feature from view 2 exists in the list of correspondences of views 2 and 3.
   - ○ Checking that the corresponding feature in view 3 is the same in the above two steps.

A typical *320 x 240* image has approximately 600 features. A typical two-way image matching has on the order of 300 correspondences, and the resulting three-way match has on the order of 100 features. [Fig. 3] shows an example, where there were 303, 285, and 331 matches between views 1-2, 1-3, and 2-3, respectively. This resulted in 185 three-way matches. Besides enabling the use of 3-view based 3-D structure estimation, the three-way matching also helps to eliminate false correspondences. In the Fig., the plots of matches between views 1-2, 1-3, and 2-3 all show spurious correspondences, none of which exist in the plot of three-way matches.



[Fig. 3] Result of three-way matching. Top Fig.s and bottom left are the two-way correspon-dences amongst views 1-2, 1-3, and 2-3, respectively. The bottom right Fig. is the resulting three-way matching.

4.1.3 Solving for Pose and Structure of Three Views

Given a set of three-way feature correspondences in three views, we can now estimate the 3-D structure of the scene. Since the images were acquired by a mobile robot, one could try to use odometry information to define the relative poses of the three views, and use that information to perform geometric triangulation of the correspondences and generate 3-D structure. However, the odometry information is not accurate enough (and there may also be vibrations, slippage, and mis-calibrations), so that poor 3-D structure is generated in this way. Instead, an algorithm that estimates both the pose of the three views as well as the structure must be used to generate accurate 3-D structure. Our algorithm consists of the following steps:

- ● Compute un-warped homogeneous coordinates for the features.

Since we typically use a wide angle lens (>60 degrees horizontal field of view) with significant barrel distortion, the first step is to compute unwarped feature coordinates. This is done by first calibrating the camera with a lens model that includes radial and tangential distortion parameters [12].

- ● Compute an initial estimate of the poses of the views and the 3-D structure using two-view algorithms.

First the algorithm of Longuet-Higgins [13] is applied to obtain a closed form estimate of the pose between view 1 and view 2. This estimate is further refined using a non-linear minimization algorithm to minimize the residual of the epipolar constraint [14]. During each iteration of the non-linear minimization, robust outlier detection is used to remove any false correspondences. The same two algorithms are then applied to view 1 and view 3. This procedure gives us the relative poses of view 2 and view 3 with respect to view 1, up to scale. To estimate the relative scale of the motion between views 1-2 and views

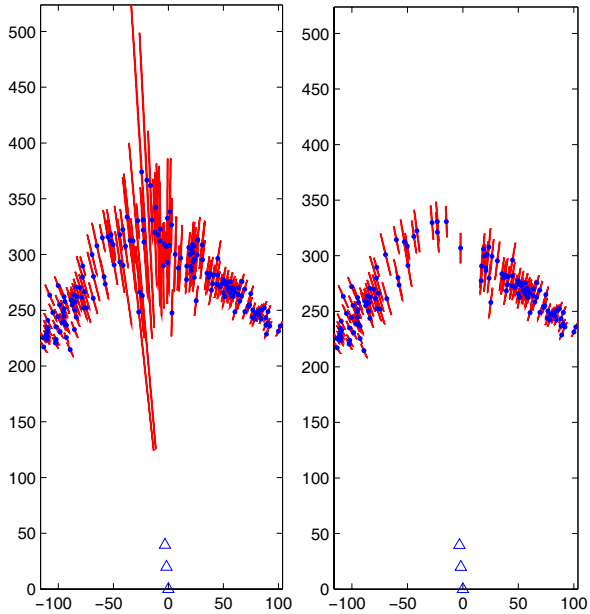
1-3, we compute estimates of the 3-D structure by triangulating views 1-2, and by triangulating views 1-3. Then, the inverse of the median ratio of corresponding triangulated point depths is used as an estimate of the relative scale of the motions.

● Refine the pose and structure estimates.

Given the initial estimates of the motion between views 1-2 and views 1-3, as well as an initial estimate of the 3-D structure from the triangulation of views 1-2, a non-linear minimization algorithm similar to that described in [14] is used to refine the motion and structure solution. In this algorithm, the motion between the three views is represented with the minimal number of parameters, eleven. Since we use a calibrated camera, we can compute Euclidean transformations (true rotations and translations) and estimate metric structure. Thus, the pose between two views is expressed by 6 degrees of freedom — 3 translational degrees of freedom, and 3 rotational degrees of freedom. However, the entire solution is known up to one global scale parameter. Thus, we take the pose between views 1 and 2 to be unit norm translation, resulting in 5 DOFs between view 1 and 2, and 6 DOFs between view 1 and 3.

The algorithm optimizes the motion and the 3-D structure so as to minimize the reprojection error of the 3-D structure onto the three views. At each iteration, robust outlier rejection is performed by detecting those points for which any of the three view reprojections has an error 4 times larger than the robustly estimated standard deviation of all the reprojections. At each iteration, a sparse Jacobian of the reprojection error with respect to both the 11 motion and *3*N* structure parameters (where *N* is the number of points being considered) is computed in order to refine the estimates using a gradient descent step.

The algorithm typically converges in 5 to 10 iterations, which take a total of on the order of 0.06 seconds of computation on a PIII 1GHz processor. Typically, there are 30 to 70 resulting points with good 3-D structure. [Fig. 3] shows an example of the 3-D reconstruction, where two vertical walls with a feature-rich image projected on to them with an LCD projector were imaged. The robot moved 20 cm forward between image captures. Shown is the resulting 3-D structure, along with its uncertainty (computed by multiplying the rms reprojection error with the sensitivity of the location of each 3-D structure point to the reprojection error). Points near the center of expansion of the (almost) purely translational motion have very large depth uncertainty. The right plot of the Fig. shows only those points with depth uncertainty less than 10 cm. Due to the fact that the motion is almost purely translational along the optical axis of the camera, the most difficult case for triangulation, the points on the two walls have depth errors of approximately 10 cm. If two cameras were to be used as a stereo pair, this Fig. could be improved by an order of magnitude, but in this paper we focus on methods that work with a single camera.



[Fig. 4] The computed 3-D structure, with uncertainty, and estimated locations of the three views. Units are centimeters. The right Fig. shows the structure after points with depth uncertainty larger than 10 cm removed.

4.1.4 Storing 3-D Structure and Associated Features

After successfully estimating 3-D structure, it is added as a new entry into the database of visual landmarks. The items that need to be stored in the database are:

- The 3-D structure. This will be used to estimate the robot pose from new viewpoints.
- One of the original three views of the landmark. This will be used to identify landmarks from new viewpoints.
- The list of features in the original view of the landmark, along with a link to the corresponding 3-D structure point of the landmark.

Note that the original images must not be stored.

4.2 Visual Measurements

Given a database of previously generated 3-D visual landmarks, and given a new image acquired by the robot after it has traveled throughout the environment an arbitrary amount, we can attempt to localize the robot with respect to one or more of the landmarks according to the following procedure:

1. Match the new image to the database of landmark images
2. For each match, estimate the relative pose to that landmark given the 3-D structure and the feature correspondences between the new view and the landmark

Matching a newly acquired image to the database of landmark images is done using Lowe's SIFT-based object recognition algorithm. This results in very reliable matches, with virtually no false positives. The matching is also very robust, able to recognize the correct landmark(s) over a wide variation of poses and lighting conditions. The algorithm also provides us with a very reliable list of corresponding features in the new image and the landmark image, which is what we need in order to estimate the relative pose to the landmark.

Since the landmark database associates a 3-D point to the features in the landmark image, the correspondences between images found in the above procedure give us a set of correspondences between the new image's features and 3-D landmark points. These 2-D to 3-D correspondences are used to estimate pose in the following way:

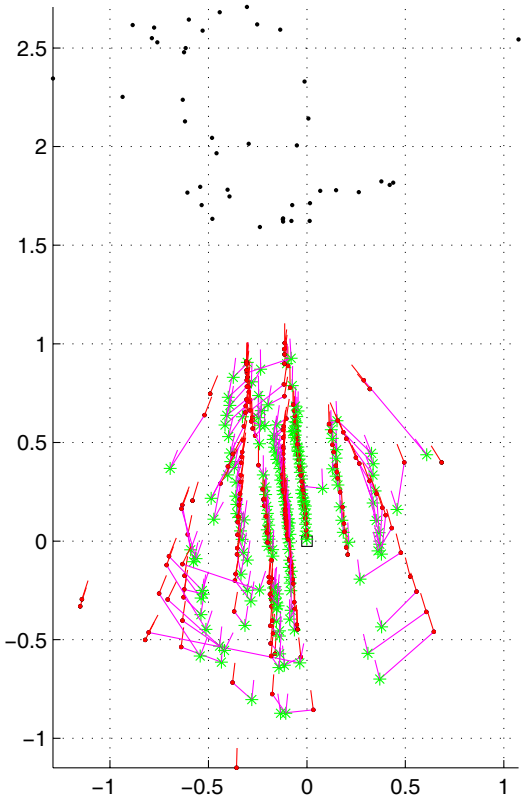
First, the unwarped homogeneous coordinates of the features of the new image are calculated, using the camera calibration parameters. Then, the fairly robust POSIT method [15] of DeMenthon and Davis is used to generate an initial estimate of the pose. The algorithm is called within a loop, where robust outlier rejection is performed. This initial estimate is then used with a non-linear minimization routine, where the 6 DOFs of the pose (3 translational, 3 rotational DOF) are found that minimize the projection error of the rigidly transformed 3-D landmark structure onto the corresponding points in the new image.

Typically, 10 to 40 points of a landmark are seen in a new image and are used to estimate the new pose. The pose is estimated with an accuracy of approximately 10 cm and 2~degrees, and successful estimations occur when the new view is within a 1 m radius of where the landmark was originally seen and created.

[Fig. 5] shows an example of the accuracy of localizations. A landmark was created (in an albeit challenging area, so that only 30 landmark 3-D points exist). The landmark 3-D points had an average depth of 2 m. The robot was driven back and forth around the location of the landmark creation (black square), and at the same time it's ground truth pose was recorded using a NAV 200SICK\TM laser rangefinder with reflective markers placed in the environment. The Fig. shows the ground truth position (solid dots) and orientation, as well as the estimated pose (asterisks connected by a line to the respective ground truth dot). Within a 0.4 m radius of the landmark creation, pose estimation is very frequent and accurate (less than 2 cm error). If the robot is further away from the initial pose, pose estimations are less frequent,

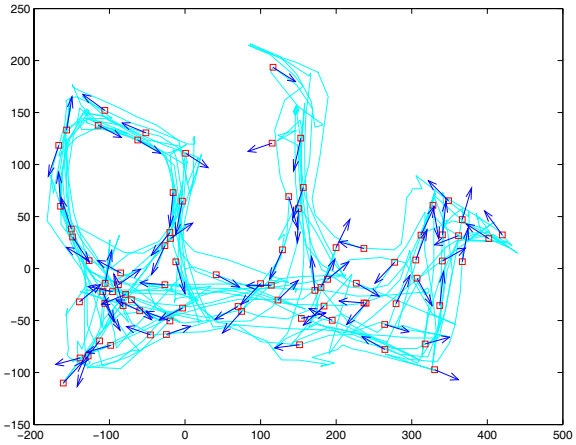
because there are less matching features to the landmark (from the change in viewpoint). The pose estimate also becomes less accurate.

This is mainly due to the error in the depth of the 3-D landmark structure, which cause a noticeable bias in the estimated pose. Beyond 1 m from the landmark creation point, too few features of the landmark are detected, and/or the pose estimation routine fails to converge due to the larger viewable distortion of the landmark structure.



[Fig. 5] Estimated poses with ground truth. Black dots are 3-D structure of landmark. Red dots and emanating dashes are ground truth poses of robot, and connected green asteriks and dashes are the estimated poses. Units are meters.

[Fig. 6] shows the ground truth path of the robot and locations and orientations of the landmarks generated in a larger scale experiment where a robot was driven around a house for 32 minutes. During that time, 82 landmarks were created.



[Fig. 6] The path of the robot during a large-scale experiment of various loops inside a single-floor home. Red squares and blue arrows are the location and orientation of the robot when landmark are generated.

Most of these landmarks were created during the first loop around the house, and then re-observed on subsequent loops. In the Fig., where there appears to be more than one landmark at the same location, it is because the robot was moving in a different direction (and thus viewing a different part of the environment, corresponding to a new landmark). There were a total of 1127 visual measurements made, at an average rate of one measurement every 1.7 seconds.

Applying some selection criteria as to which visual measurements to accept as reliable and which to reject as unreliable (described in more detail in the next section), 966 of the visual measurements (85.7% of them) would be used by the SLAM module. For these measurements, the mean, median and rms distance (distance defined as $\sqrt{x^2 + y^2}$) error of the visual measurements (compared to the ground truth value) are 6.68, 10.19, and 13.87 cm, respectively. The mean, median, and rms values of the magnitude of the orientation error are 1.26, 1.66, and 2.31 degrees, respectively. Table 1 shows a distribution of the accuracy of the visual measurements.

[Table 1] Distribution of visual measurement accuracy.

xy err (cm)	accepted	rejected	Accepted/total	xy err (cm)	accepted	rejected	Accepted/total
<10	627	48	92.89%	0-10	627	48	92.89%
<20	843	99	89.49%	10-20	216	51	80.90%
<30	908	114	88.85%	20-30	65	15	81.25%
<40	946	121	88.66%	30-40	38	7	84.44%
<50	964	127	88.36%	40-50	18	6	75.00%
<70	966	135	87.74%	50-70	2	8	20.00%
<100	966	136	87.66%	70-100	0	1	0.00%
<150	966	140	87.34%	100-150	0	4	0.00%
<200	966	142	87.18%	150-200	0	2	0.00%
>200	0	19	0.00%				

5. Pre-Filter

The Pre-filter provides a convenient way to modify the vSLAM system to accommodate for different desired trade-offs between accuracy and robustness. In particular, different applications may have different criteria for what is a reliable measurement. The basic rule is that ProjErrRMS and Slope should be very small and NPoints should be large for a measurement to be considered reliable. A simple scheme is hence to define a region over these three variables, and to rejject any measurement that falls outside that region. If the boundaries of the region are defined by some threshold values, and these are set conservatively (the region is too small), a big portion of all measurements are rejected resulting in very few updates of the map. On the other hand, if too many of the measurements are accepted (the region is too large), the error variance of the visual measurements may be higher than is acceptable.

Experimentally, we have determined that a good compromise between accurate accepted, yet few rejected measurements for most applications is obtained if the following criteria are used:

- if ProjErrRMS>3.0, then reject the measurement
- if NPoints<10, then reject the measurement
- if Slope>0.1, then reject the measurement

Note that a measurement is rejected as soon as one of the above three inequalities is satisfied.

The concept of a Pre-filter allows the user to select their own acceptance region. E.g., in some applications the environment may be poor on texture, which is needed to create and recognize landmarks. If, in the same application, the over-all requirement on accuracy is fairly low, the robustness of the localization is improved by enlarging the acceptance region. More visual measurements are then accepted making sure that frequent feedback is achieved in the SLAM module for map update and localization. This comes with the price of measurements that on average are somewhat less accurate.

Based on experiments, it is determined that the measurements remaining after the above filtering fairly well can be described by the true relative pose plus a stochastic, uncorrelated error vector $[\varepsilon_x, \varepsilon_x, \varepsilon_\Theta]^T$ belonging to the Gaussian distribution of mean zero and covariance Σ^{vis}, where

$$\Sigma^{vis} = \begin{bmatrix} 81cm^2 & 0 & 0 \\ 0 & 81cm^2 & 0 \\ 0 & 0 & 0.0019\deg^2 \end{bmatrix}.$$

In reality, the error vectors do not belong to a Gaussian distribution, but to some heavier-tailed distribution. But

satisfactory results are obtained when we use a Gaussian approximation.

6. SLAM Back-end

The SLAM Back-end maintains a map of all landmarks and estimates the location of the robot within this map. In particular, it estimates the global location of each landmark and the robot. Note that the global landmark location is completely unknown to the Visual Front-end.

The inputs to this module consist of

- visual measurements passed on from the Pre-filter,
- dead reckoning data from the odometry sensor, and
- the previous estimate of map and robot pose (location and heading).

The outputs consist of a refined estimate of the map and an updated robot pose.

To describe the algorithm in the SLAM Back-end, let us denote the map by Φ. The map consists of references to the landmarks created in the Visual Front-end, each of which will be denoted φ_n. The total number of landmarks at a given time is denoted *N*. This variable is initially set to zero, but will over time increase as landmarks are created and added to the map. The robot pose is denoted s_t, where *t* is a discrete number corresponding to the current time. The *pose* is defined by the *3 x 1* vector $[x_t, y_t, \theta_t]^T$, where x_t and y_t is the location and θ_t is the heading, at time *t*.

To implement vSLAM, one needs to know two stochastic equations. The first equation is the *motion model*, which is a state-space model

$$s_{t+1} = f(s_t, u_t) + w_t \tag{1}$$

where u_t denotes the odometry collected from time *t* to *t+1* and w_t is a noise process representing the error in odometry.

The specific motion model depends on the robot's odometry, which on the other hand depends on the robot's kinematics and of the floor surface. In this paper, we use a motion model derived under the assumption of a non-holonomic drive system and parameterized by two noise parameters; however, the precise model is irrelevant for the development of the algorithm. Our model results in

$$\begin{aligned} E(w_t) &= 0 \\ E(w_t w_t^T) &= g(\theta_t, u_t, \sigma_T, \sigma_T) \end{aligned}$$

where σ_T and σ_R characterize the translational and rotational odometry uncertainty.

The second model is the *measurement model*

$$y_t = h(s_t, \varphi_{n_t}) + v_t \tag{2}$$

where n_t is the observed landmark at time *t* and v_t is a noise process representing the error in the relative visual measurements. If the true robot pose is s_t, then the "error-fre" visual measurement is given by $h(s_t,\varphi_{nt})$, $E(v_t)=0$ and $E(v_t v_t^T)=\Sigma^{vis}$ (see also (2)).

The sequence $s^t=s_1,s_2,\ldots,s_t$ denotes the path of the robot up to time *t*. While s_t denotes a single robot pose, s^t denotes a sequence of poses from time *1* to time *t*.

The ultimate goal is to estimate the robot pose s_t based on

1. an estimate of s_{t-1},
2. measurements u_{t-1} and y_t,
3. the landmark pose φ_{nt}

However, also φ_{nt} is uncertain and must be estimated. Therefore, let $\varphi_{nt,t}$ denote the estimate of φ_{nt} at time *t*. The idea proposed in [4] to estimate s^t and Φ (rather than s_t and Φ) is then adopted. In particular, we consider the posterior distribution $p(s^t,\Phi|n^t,y^t,u^t)$. Using Bayesian calculus and standard assumptions employed in probabilistic robotics, it is straight-forward to show that

$$\begin{aligned} p(s^t, \Phi \mid n^t, y^{t-1}, u^t) &= p(s^t \mid s^{t-1}, u_t) \\ &\quad \cdot p(s^{t-1}, \Phi \mid n^{t-1}, y^{t-1}, u^{t-1}) \end{aligned}$$

Furthermore, the following important factorization is used in the vSLAM algorithm:

$$p(s^t, \Phi \mid n^t, y^t, u^t) = p(s^t \mid n^t, y^t, u^t) \cdot \prod_{i=1}^{N} p(\varphi_i \mid s^t, n^t, y^t) \quad (3)$$

This factorization decomposes the problem of estimating $N+1$ poses, and their cross-correlations, simultaneously; into estimating one robot path and N landmark poses (separately). The factorization is justified by expressing the problem as a Bayesian network. In particular, knowing the robot path *d*-separates [16] the individual landmark estimation problems and makes them independent of each other.

The vSLAM update now proceeds by first updating the robot path, and then the map of visual landmarks.

6.1 Simple Localization

Similarly to in [17], vSLAM implements the robot path update in (3) as a particle filter. Some advantages of this choice for the robot pose update is that the particle filter does not assume the pose distribution to be neither Gaussian, nor uni-modal. The landmark pose updates are implemented as Kalman filters because of the simple dynamic of the landmark poses (they are typically constant).

The basic particle filter is implemented by first computing some number, e.g. 200, of hypothetical paths (so called *particles*) based on the motion model (1), odometry data, and the odometry noise model [18]. These initial particles represent a prior distribution of the robot path. Thereafter, so called *importance factors* are computed based on the prior distribution and the visual measurements. By resampling, with replacement, from the particles with probabilities equal to the importance factors, it can be shown that the distribution of the resulting particles (representing the posterior distribution) indeed approximates the true distribution $p(s^t|n^t,y^t,u^t)$.

6.2 Robust Localization

As recognized by many researchers, the basic particle filter performs poorly if the motion model generates too few hypothetical paths in regions where $p(s^t|n^t,y^t,u^t)$ is large. This flaw of the particle filter is in vSLAM addressed by using a *mixture proposal distribution* instead of the motion model to compute the prior distribution [10,11]. In particular, a subset of the particles, say 180, in the prior distribution are generated based on the motion model, while the remainder of them, say 20, are generated based on the measurement model (2). The particles generated according to the basic filter are called *nominal particles*, while the ones generated based on the measurement model are called *dual particles*.

The importance factors for each nominal particle is in principle computed based on the measurement model, while the importance factors for each dual particle in principle is computed based on the motion model. However, great care must be exercised to ensure that the importance factors of the nominal and dual particles are consistent and can be compared in the resampling process.

A very useful feature in vSLAM enabled by the mixture proposal distribution is that the ratio of nominal versus dual particles can be changed to deal with extreme situation, such as mal-functioning sensors, or kidnapping. More specifically, if a kidnapping scenario is detected or suspected, then one should generate only dual particles, and compute the importance factors from a uniform distribution. Note that simply modifying the odometry noise model to accommodate for the large odometry error typically will not work if a particle filter is employed since, in a normal-sized environment, it would require a very large number of particles to approximate a uniform distribution. Also, if no prior particle is in some neighborhood of the true pose, then all importance factors are likely to become less than the machine precision of the computer; i.e., all importance factors will become zero making it impossible to make any

inference from the measurements.

6.3 Mapping

Now, proceed to the landmark pose estimation. It is interesting to realize that since each Kalman filter update of a landmark pose must obey $p(\varphi_i|s^t,n^t,y^t)$, it is necessary to associate a separate bank of Kalman filters (one filter per landmark) to each robot path hypothesis s^t (each particle). In other words, if there are 200 particles and 50 landmarks, then the total number of Kalman filters is 200 times 50; i.e., 10,000. On the other hand, each Kalman filter contains only three state variables (the recursively estimated landmark pose), and three measurement variables (the current landmark pose estimate); and only the Kalman filters associated to the measured landmark is updated in any given iteration of the vSLAM filter. Consequently, the update scheme becomes computationally very efficient.

7. Experiments

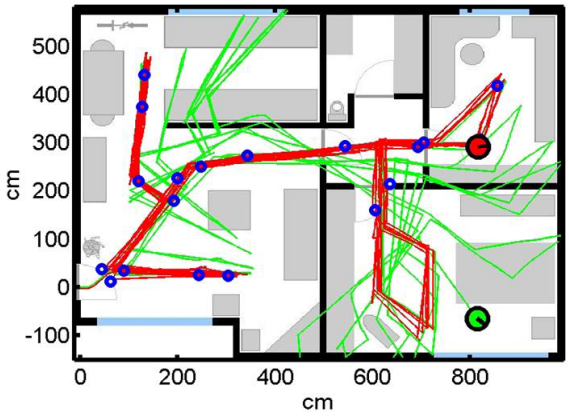
To illustrate the performance of vSLAM, we shall investigate its localization capability as it is integrated on a robot operating in a typical home environment.

7.1 vSLAM in Home Environment (Robot follows a pre-defined path)

First, consider [Fig. 7], which shows the result of vSLAM after the robot has traveled around along a *reference path* in a two-bedroom apartment. The reference path, on which the robot drove, was unknown to the vSLAM algorithm.

To help interpret the result, the layout of the apartment is superimposed in the Fig.. This layout was neither generated, nor used, by vSLAM.

The vSLAM algorithm builds a map consisting of landmarks, which are marked as circles in the Fig.. The green path (odometry *only*) is obviously incorrect, since, according to this path, the robot is traversing through walls and furniture. The red path (the vSLAM corrected path), on the other hand, is consistently following the reference path. The vSLAM path, which uses a combination of visual measurements and odometry, provides a robust and accurate position determination for the robot.



[Fig. 7] Example result of SLAM using vSLAM. Red path: vSLAM estimate of robot trajectory. Green path: odometry estimate of robot trajectory. Blue circles: vSLAM landmarks created during operations.

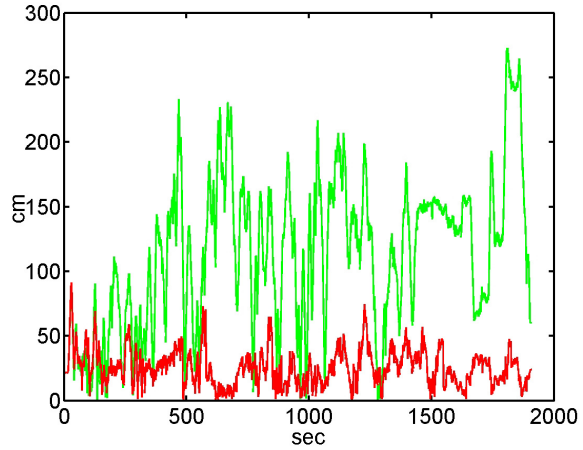
[Fig. 8] The current view from the camera (left), which is successfully matched with the landmark image (right).

[Fig. 8] shows that also a motion blurred image can be used by vSLAM. Indeed, the landmark on the right is recognized in the blurred "current" image on the left in spite it is only partly visible in the current view.

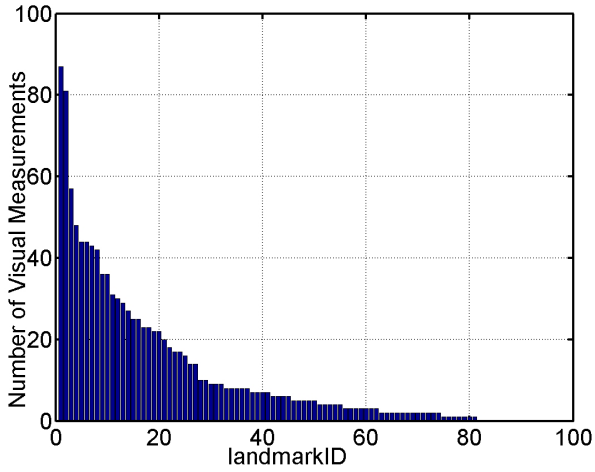
Average localization error using vSLAM is in this example about 10-15 cm, while the median error is 9 cm.

7.2 vSLAM in Home Environment (Robot follows a random path)

In many applications, it is not reasonable to assume that a robot travels on a reference path. Let us therefore now investigate vSLAM localization in a scenario where the robot travels in an arbitrary path (unknown to vSLAM). The experiment took place in a large living room of *7m x 4m* on hardwood floor. As a reference, we acquired ground truth pose data with a Nav200 SICK laser. The speed of travel was about *15 cm/sec*.



[Fig. 9] Position localization error using only odometry (green curve) and using vSLAM (red path).



[Fig. 10] Number of visual measurements per landmark.

The localization error as a function of time is shown in [Fig. 9]. As a comparison, also the pure odometry results are displayed. The localization accuracy using vSLAM is in this particular example somewhat lower than in the previous example. In particular, the average error is about 20 to 25 cm, while the median is about 14 cm.

It is also of interest to investigate the distribution of visual measurements among the landmarks. As seen in [Fig. 10], it turns out that a small portion of all landmarks result in the big majority of all visual measurement.

There are two reasons to this behavior. First of all, some landmarks were created late in the run, and did not have time to result in many measurements. But, a second important reason is that landmarks differ in quality. Some landmarks correspond to scenes with plenty of texture and many unique features. Other landmarks correspond to scenes for which the visual front-end barely managed to create a landmark. Poor landmarks, are difficult to recognize and generate visual measurements from. .

8. Conclusions and Future Work

We have described vSLAM, a novel vision- and odometry-based SLAM algorithm that enables low-cost and robust navigation in cluttered and populated environments. The vSLAM algorithm does not require an initial map, and the algorithm is typically good at detecting and correcting for slippage and collisions.

The key characteristics of vSLAM are

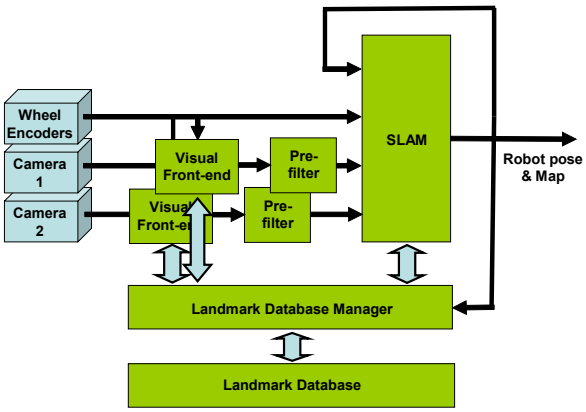
1. The low cost, which is enabled by basing the algorithm on a low-cost camera and an odometer.
2. The visual landmarks that are created and recognized in the front-end. They are essentially unique, which virtually eliminates the data association problem present in other landmark schemes.
3. The update scheme for robot pose and map. It is based on a particle filter and a Kalman filter bank and

enables an efficient real-time implementation.

4. The mixed proposal distribution and the dynamic mixture ratio, which dramatically improves localization recovery from kidnapping and other large disturbances.

While not explored in detail in this article, it can be shown that vSLAM is highly robust to kidnapping and dynamic changes in the environment, for example, changes in lighting, moving objects and/or people. vSLAM recovers quickly from kidnapping once the map has become dense with landmarks. Early in the mapping procedure, is it important to avoid kidnapping and disturbances in general.

A problem with the current implementation of vSLAM is the landmark data base, which in large environments may become prohibitively large. In particular, an individual landmark occupies anywhere between 40 kB and 500 kB depending on the number of visual features associated with the landmark, and the number of particles used in the SLAM module. For most practical implementations, it is therefore essential to integrate a *Landmark Database Manager* with the basic vSLAM system (see Fig. 1). Such a database manager typically includes a *pruning mechanism*, which removes poor landmarks. But it also includes a *segmentation scheme* that partitions the full vSLAM map into sub regions to limit the required primary memory for the map.



[Fig. 11] Block diagram of a proposed extended vSLAM system.

In some applications, it may also be appropriate to use more than one camera (see Fig. 11). For example, one forward-pointing and one backward-pointing camera. Note that these are independent sensors, and images acquired by each one of them are processed separately adding landmarks to a common database, but generating visual measurements independently. The system as described in this paper already accommodates for the use of multiple cameras. An example of when multiple cameras might be appropriate is when the environment is poor on texture and where lowest possible price is not crucial.

It is useful to understand that the use of multiple cameras does not necessarily increase the computational requirements since some alternating image acquisition scheme can be employed to optimize the use of visual information.

Finally, note that if the environment does not contain a sufficient amount of visual features, then landmarks will not be created and visual measurements will not be computed. vSLAM will still produce pose estimates, but the estimates will be exactly what is obtained from wheel odometry. This scenario is rarely experienced, but is most likely to happen in environments that are free from furniture, decorations, and texture.

References

[1] S. Thrun, "Probabilistic algorithms in robotics," Artificial Intelligence, no. 4, pp. 93–109, 2000.

[2] S. Thrun, D. Fox., and W. Burgard, "A probabilitic approach to concurrent mapping and localization for mobile robots," Machine Learning, vol. 31, pp. 29–53, 1998.

[3] S. Thrun, "Robotic mapping: A survey." Carnegie Mellon University, Tech. Rep. CMU-CS-02-111, February 2002.

[4] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous

localization and mapping problem," in 2002 IEEE ICRA, Workshop W4 Notes, 2002.

[5] D. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, 2004.

[6] J. Wolf, W. Burgard, and H. Burkhardt, "Robust vision-based localization for mobile robots using an image retrievel system based on invariant features," in Proc. of Int. Conf. on Robotics and Automation (ICRA), 2002.

[7] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," The International Journal of Robotics Research, vol. 21, no. 8, pp. 735–758, 2002.

[8] S. Se, D. Lowe, and J. Little, "Global localization using distinctive visual features," in Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland, October 2002.

[9] A. Doucet, N. de Freitas, and N. Gordon, Eds., Sequential Monte Carlo Methods in Practice. Springer-Verlag, New York, 2001.

[10] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, "Particle filters for mobile robot localization," in Sequential Monte Carlo Methods in Practice, A. Doucet, N. de Freitas, and N. Gordon, Eds. Springer-Verlag, New York, 2001, ch. 19.

[11] D. Hinkley, Bootstrap Methods and their Application. Cambridge Series in Statistical and Probabilistic Methematics, 1997.

[12] J. Heikkil and O. Silvn, "Calibration procedure for short focal length off the-shelf ccd cameras," Proc. 13th International Conference on Pattern Recognition, pp. 166–170, 1996.

[13] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," Nature, vol. 293, pp. 133–135, 1981.

[14] R. I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521623049, 2000.

[15] D. DeMenthon and L. S. Davis, "Model-based object pose in 25 lines of code," International Journal of Computer Vision, vol. 15, pp. 123–141, June 1995.

[16] G. R. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, Probabilistic Networks and Expert Systems. Springer-Verlag, New York, 1999.

[17] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in Proceedings of the AAAI National Conference on Artificial Intelligence. Edmonton, Canada: AAAI, 2002.

[18] S. Thrun, "Probabilistic algorithms in robotics," Carnegie Mellon University, Tech. Rep. CMU-CS-00-126, April 2000.

Niklas Karlsson, Ph.D.

E-mail : niklas@evolution.com

LuisGoncalves, Ph.D

E-mail : luis@evolution.com


Mario E. Munich, Ph.D.

E-mail : mario@evolution.com


Paolo Pirjanian, Ph.D.

E-mail : paolo@evolution.com