Vision-based localization and mapping system for AUV intervention

Narcís Palomeras, Sharad Nagappa, David Ribas, Nuno Gracias, Marc Carreras
University of Girona, Edifici Politecnica IV, Campus Montilivi 17071 Girona, Spain
{*npalomer, snagappa, dribas, ngracias, marcc*}*@eia.udg.es*

***Abstract*— This paper presents a modular localization and mapping system for intervention autonomous underwater vehicles working in semi-structured environments with known landmarks. The system is divided in several modules to make it as generic as possible. Two visual detection algorithms can be used to compute the position of known landmarks by comparing the images taken by the vehicle against an *a priori* known template. Navigation data, provided by standard navigation sensors, is adapted and merged together with landmark positions by means of an extended Kalman filter. This filter is capable of estimating vehicle position and linear velocity as well as the position of detected landmarks in real-time. Experiments performed with the Girona 500 AUV in a water tank demonstrate the proposed method.**

I. Introduction

This paper presents a real-time simultaneous localization and mapping (SLAM) system for an autonomous underwater vehicle (AUV). The system is split into several modules and can be used for any AUV equipped with a basic sensor suite. This work is being developed in the context of the PANDORA FP7 project [1]. The main goal of this project is to improve *persistent autonomy* in semi-structured underwater environments like deep-sea long term observatories or oil and gas industry facilities. Thus, AUVs have to be able to navigate in these facilities, localize key elements like panels, docks and canisters, and perform intervention operations like turn a valve in a panel, communicate/charge through a dock or put objects in a canister. The cost of inspecting and operating offshore structures with AUVs is much smaller than using remotely operated vehicles (ROVs) [2]. However, if instead of deploying these vehicles for each mission they are left in situ, this cost can be further reduced.

A critical issue for this purpose is to create a computational mechanism that enables an AUV to maintain a geometric description of its world and its place in it. This geometric description has to be the base for the semantic description used to plan and execute autonomous actions for the AUV in real-time. A robust approach could be map-based localization. However, it can only be used if a precise map is available and all their elements are static. Another alternative is to navigate relative to a particular object, like a panel or a dock, instead of keeping a global map of the site. Then, prior knowledge of the structure where the vehicle has to operate allows the use of model based pose estimation techniques that rely on a list of detected features [3]. Despite the simplicity and robustness of this approach, it does not provide a geometric map of the environment that may be useful for an on-board planner. A more complete mechanism to complete the task at hand is SLAM. Multiple alternatives have been proposed to simultaneously localize a vehicle with respect to a map while refining this map in the underwater domain [4]–[6]. However, most of these solutions are difficult to implement in real-time.

The purpose of this paper is to present a generic 6 degrees of freedom (DoF) SLAM algorithm based on an extended Kalman filter (EKF). The filter uses a constant velocity model and information about orientation and angular velocity for the predictions. Three different updates are possible: position updates, velocity updates, and landmark updates. Position updates provide information about vehicle position with respect to the world frame. In the experiments presented here, this information is supplied by a depth sensor, and a global positioning system (GPS), when the vehicle is at the surface. Velocity updates are velocities measured with respect to the vehicle's frame by a doppler velocity log (DVL), a visual odometer or any other velocity sensor. Landmark updates contain the pose of an identifiable mark with respect to the vehicle's frame. Two vision-based algorithms are implemented for this purpose. One method uses special markers that can be easily identified and positioned with respect to the vehicle. The other method is based on a template-matching algorithm. It uses images of objects in the environment avoiding the addition of extra markers. The whole system is separated in blocks and programmed using the popular middleware Robot Operating System (ROS) [7]. All the code is open-source and can be download from *https://bitbucket.org/udg_cirs/cola2*.

The paper is organized as follows. After some preliminaries, Section II describes the transformations to be applied to each sensor measurement as well as the EKF-SLAM algorithm. It also presents two vision-based landmark detector algorithms. Section III provides details of the experiments performed in a water tank with Girona 500 AUV to test the whole system. Finally, Section IV concludes the paper and presents the future work.

This work was supported by the EU funded project PANDORA (Persistent Autonomy through learNing, aDaptation, Observation and ReplAnning") FP7-288273, 2012-2014 funded by the european commission and the Spanish project TRITON DPI2011-27977-C03-02 funded by the ministry of science and innovation.

978-1-4799-0002-2/13/$31.00 ©2013 IEEE

This is a DRAFT. As such it may not be cited in other works. The citable Proceedings of the Conference will be published in IEEE Xplore shortly after the conclusion of the conference.

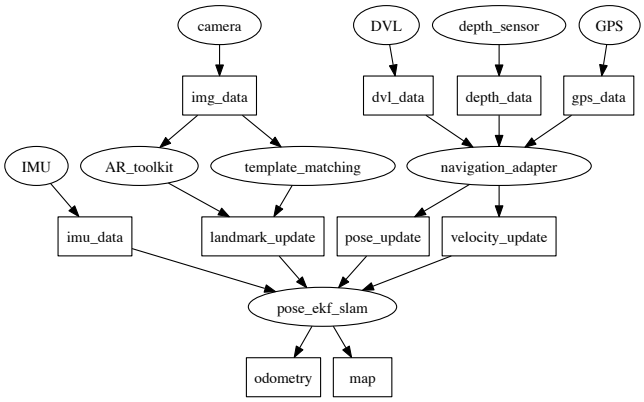


Fig. 1. System's schema for Girona 500 AUV. Ellipse-shaped nodes are processes while rectangular-shaped nodes are messages.

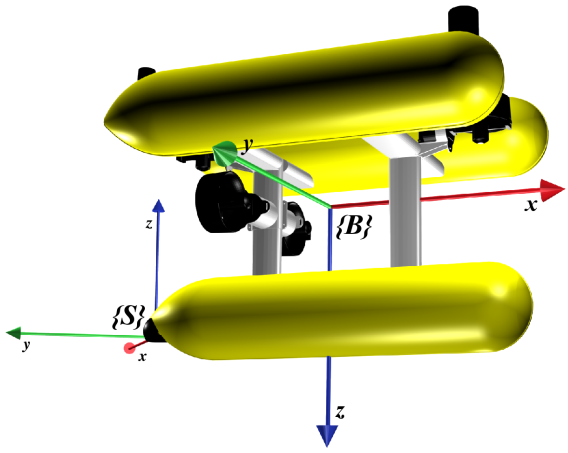


Fig. 2. Girona 500 AUV with its coordinate frame $\{B\}$ and an example of a sensor frame $\{S\}$.

II. Methodology

Fig. 1 shows all the elements involved when the vision-based localization and mapping system runs in the Girona 500 AUV [8]. Depth sensor, GPS, internal motion reference unit (IMU) and DVL are sensor drivers gathering navigation data (i.e. pose, orientation and velocity with its covariance). The *navigation_adapter* node takes these custom sensor messages and transforms them into ROS standard navigation messages (*pose_update* and *velocity_update*). The *template_matching* and the *AR_toolkit* nodes take the images generated by the camera driver and compute the position of an *a priori* known landmark with respect to the vehicle (*landmark_update*) when the landmark is in the vehicle's field of view. Pose, velocity and landmark updates are then merged using an EKF in *pose_ekf_slam* node. Outputs for this node are vehicle position and velocity with its covariance (*odometry*) as well as position and covariance for each landmark (*map*). Drivers for each sensor and *navigator_adapter* node are robot dependent but the rest of blocks can be directly used by a vehicle running ROS. Therefore, if new drivers are provided (e.g. a visual odometers) only the *navigator_adapter* has to be modified. Moreover, to add a new landmark detector compatible with *pose_ekf_slam* node, it is only required to publish the landmark position and covariance using a *landmark_update* message.

A. Preliminaries

Let $\{I\}$ be an inertial coordinate frame and $\{B\}$ a body-fixed coordinate frame, whose origin O_B is located at the center of mass of the vehicle (see Fig. 2). Furthermore, let (x, y, z) be the position of O_B in $\{I\}$ and $(\phi,\ \theta,\ \psi)$ the orientation of O_B in $\{I\}$. Let $\mathbf{v} = (u, v, w)$ be the longitudinal (surge), transverse (sway) and vertical (heave) velocities of O_B with respect to $\{I\}$ expressed in $\{B\}$ and $\mathbf{w} = (r, p, q)$ be the vehicle's angular speed around each axis. Any sensor attached to $\{B\}$ has its own coordinate frame. For the sake of simplicity, all sensor coordinate frames are identified as $\{S\}$ and their origin O_S with respect to $\{B\}$ is indicated by the translation vector $\mathbf{t}_S$ and rotation matrix $\mathbf{rot}_S$.

B. Transformations

The *navigation_adapter* node is in charge of transforming all custom messages coming from sensor drivers to ROS standard geometry messages of type *PoseWithCovarianceStamped* (i.e. *pose_update*) and *TwistWithCovarianceStamped* (i.e. *velocity_update*). Main functions of this node are: transform units to metric system, make data dextrorotatory, and transform sensor measurements from $\{S\}$ to $\{B\}$. To transform a pose measurement $\mathbf{p}_S$ obtained by a sensor fixed at coordinate frame $\{S\}$ to the coordinate frame $\{B\}$, we apply the transformation given by

$$\mathbf{p}_B = \mathbf{p}_S - \mathbf{Rot} \cdot \mathbf{t}_S, \tag{1}$$

where $\mathbf{Rot}$ is a rotation matrix measuring the orientation of O_B with respect to $\{I\}$ and $\mathbf{t}_S$ is the translation vector from O_B to O_S. The transformed covariance matrix $\mathbf{P}_{\mathbf{p}_S}$ corresponding to the measurement $\mathbf{p}_S$ is given by

$$\mathbf{P}_{\mathbf{p}_B} = \mathbf{P}_{\mathbf{p}_S} + \mathbf{J}_{\mathbf{p}_B} \cdot \mathbf{P}_{Rot} \cdot \mathbf{J}_{\mathbf{p}_B}^T, \tag{2}$$

where $\mathbf{J}_{\mathbf{p}_B}$ is the Jacobian of partial derivatives of (1) with respect to vehicle orientation, and $\mathbf{P}_{Rot}$ is the vehicle orientation covariance matrix. To transform a linear velocity measurement $\mathbf{v}_S$ obtained by a sensor fixed at coordinate frame $\{S\}$, the following equation is used:

$$\mathbf{v}_B = \mathbf{rot}_S \cdot \mathbf{v}_S - (\mathbf{w}_B \otimes \mathbf{t}_S), \tag{3}$$

where $\mathbf{rot}_S$ is the rotation matrix of O_S measured from O_B and $(\mathbf{w}_B \otimes \mathbf{t}_S)$ is the cross product of the angular velocity of O_B measured at $\{I\}$ by the translation vector from O_B to O_S. The transformed covariance matrix $\mathbf{P}_{\mathbf{v}_S}$ of measurement $\mathbf{v}_S$ is

$$\mathbf{P}_{\mathbf{v}_B} = \mathbf{rot}_S \cdot \mathbf{P}_{\mathbf{v}_S} \cdot \mathbf{rot}_S^T + \mathbf{J}_{(\mathbf{w}_B \otimes \mathbf{t})} \cdot \mathbf{P}_{\mathbf{w}_B} \cdot \mathbf{J}_{(\mathbf{w}_B \otimes \mathbf{t})}^T, \tag{4}$$

where $\mathbf{J}_{(\mathbf{w}_B \otimes \mathbf{t})}$ is the Jacobian matrix of partial derivatives of $(\mathbf{w}_B \otimes \mathbf{t})$ with respect to $\mathbf{w}_B$ and $\mathbf{P}_{\mathbf{w}_B}$ is the vehicle angular velocity covariance matrix.

Since all these non-linear transformations are performed (if necessary) before the EKF, the updates in the EKF-SLAM algorithm are all linear.

C. EKF SLAM Algorithm

The *pose_ekf_slam* node estimates the vehicle position ($[x\ y\ z]$) and linear velocity ($[u\ v\ w]$) and maps the position of several landmarks ($[lx_i\ ly_i\ lz_i]$) identified by a vision algorithm. The fusion algorithm in charge of this SLAM is an EKF [9]. Vehicle orientation ($[\phi\ \theta\ \psi]$) and angular velocity ($[p\ q\ r]$) are not estimated but directly measured by an IMU. Details of the EKF-SLAM implementation are presented next.

1) State vector: The information to be estimated by the EKF-SLAM algorithm is stored in the following state vector:

$$\mathbf{x}_k = [x\ y\ z\ u\ v\ w\ lx_1\ ly_1\ lz_1\ \dots\ lx_n\ ly_n\ lz_n]^T \quad (5)$$

where ($[x\ y\ z\ u\ v\ w]$) are defined in Section II-A and ($[lx_1\ ly_1\ lz_1]$ ... $[lx_n\ ly_n\ lz_n]$) is the position of each landmark (l_i) with respect to $\{I\}$.

2) System model: A constant velocity kinematics model is used to determine how the vehicle state will evolve from time $k-1$ to k. Landmarks are expected to be static. The predicted state at time k, $\mathbf{x}_k^-$ follows the equations:

$$\mathbf{x}_k^- = f(\mathbf{x}_{k-1}, \mathbf{n}_{k-1}, \mathbf{u}_k, t). \quad (6)$$

$$\mathbf{x}_k^- = \begin{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \end{bmatrix} + \mathbf{R}(\phi_k\theta_k\psi_k) \left(\begin{bmatrix} u_{k-1} \\ v_{k-1} \\ w_{k-1} \end{bmatrix} t + \begin{bmatrix} n_{u_{k-1}} \\ n_{v_{k-1}} \\ n_{w_{k-1}} \end{bmatrix} \frac{t^2}{2} \right) \\ u_{k-1} + n_{u_{k-1}} t \\ v_{k-1} + n_{v_{k-1}} t \\ w_{k-1} + n_{w_{k-1}} t \\ lx_{1_{k-1}} \\ ly_{1_{k-1}} \\ lz_{1_{k-1}} \\ \dots \end{bmatrix} \quad (7)$$

where t is the time period, $\mathbf{u} = [\phi\ \theta\ \psi]$ is the control input determining the current vehicle orientation and $\mathbf{n} = [n_u\ n_v\ n_w]$ is a vector of zero-mean white Gaussian acceleration noise whose covariance, represented by the system noise matrix $\mathbf{Q}$, has been set empirically:

$$\mathbf{Q} = \begin{bmatrix} \sigma_{n_u}^2 & 0 & 0 \\ 0 & \sigma_{n_v}^2 & 0 \\ 0 & 0 & \sigma_{n_w}^2 \end{bmatrix} \quad (8)$$

Associated with the state vector $\mathbf{x}_k$ there is the covariance matrix $\mathbf{P}_k$. Next equation is applied to predict the evolution of this matrix:

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_{k-1} \mathbf{W}_k^T, \quad (9)$$

where $\mathbf{A}_k$ is the Jacobian matrix of partial derivatives of f with respect to the state (5) and $\mathbf{W}_k$ is is the Jacobian matrix of partial derivatives of f with respect to the process noise $\mathbf{n}$.

3) Measurements: Three linear measurement updates are applied in the filter: pose, velocity and landmark updates. A sensors measurement can be modeled as:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{s}_k, \quad (10)$$

where $\mathbf{z}_k$ is the measurement itself, $\mathbf{H}$ is the observation matrix that relates the state vector with the sensor measurement, and $\mathbf{s}_k$ is the sensor noise. Updates can be applied by means of the equations:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1}, \quad (11)$$

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-), \quad (12)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-, \quad (13)$$

where $\mathbf{K}_k$ is the Kalman gain, $\mathbf{R}$ the measurement noise covariance matrix and $\mathbf{I}$ an identity matrix. Below, it is shown how to define $\mathbf{z}_k$ and $\mathbf{H}$ to perform each update applying equations (11)-(13).

Several sensors can provide pose information. For instance a GPS receiver measures vehicle position in the plane $(x,\ y)$ while the vehicle is at the surface, a pressure sensor transforms pressure values into depth (z), or an ultra short base line (USBL) device measures vehicle position $(x,\ y,\ z)$ while submerged. Thus, to integrate a pose sensor is applied:

$$\mathbf{z}_k = [x\ y\ z] \quad (14)$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3n} \end{bmatrix} \quad (15)$$

where $\mathbf{I}_{3\times3}$ denotes the 3×3 identity matrix and $\mathbf{0}_{3\times3n}$ denotes the $3 \times 3n$ zero matrix with n being the number of landmarks. If only $(x,\ y)$ or the (z) is available, $\mathbf{z}_k$ and $\mathbf{H}$ have to be properly arranged.

Velocity updates are provided by sensors like a DVL or a visual odometer. These sensors are able to measure linear velocities with respect to the sea bottom or the water around the vehicle.

$$\mathbf{z}_k = [u\ v\ w] \quad (16)$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3n} \end{bmatrix} \quad (17)$$

If only velocity updates are available, the navigation module behaves as a dead reckoning algorithm that drifts over time. However, if pose updates are present or sufficient landmarks are detected in the environment, the navigation module is able to keep its position error bounded.

The visual detection algorithms, detailed in section II-D, give information about the relative position of a landmark with respect to $\{B\}$. This information not only updates the detected landmark position in the map but also the vehicle pose. Both detection algorithms described later, use an *a priori* known template to identify and compute the relative position of these landmarks.

$$\mathbf{z}_k = [L_x\ L_y\ L_z] \quad (18)$$

$$\mathbf{H} = \begin{bmatrix} -\mathbf{Rot}^T & \mathbf{0}_{3\times3} & \dots & \mathbf{Rot}^T & \dots \end{bmatrix} \quad (19)$$

where $[L_x\ L_y\ L_z]$ is the relative position of the landmark with respect to the vehicle and $\mathbf{Rot}$ is the vehicle orientation rotation matrix.

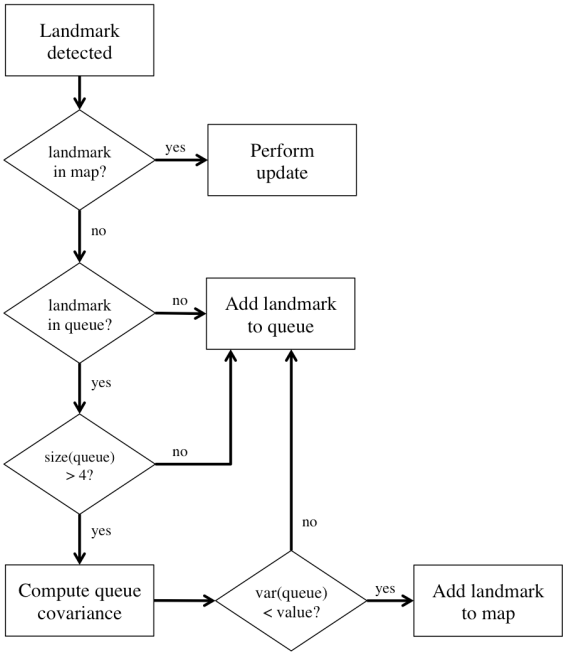


Fig. 3. Diagram to decide if a new landmark has to be introduced in the map or not.

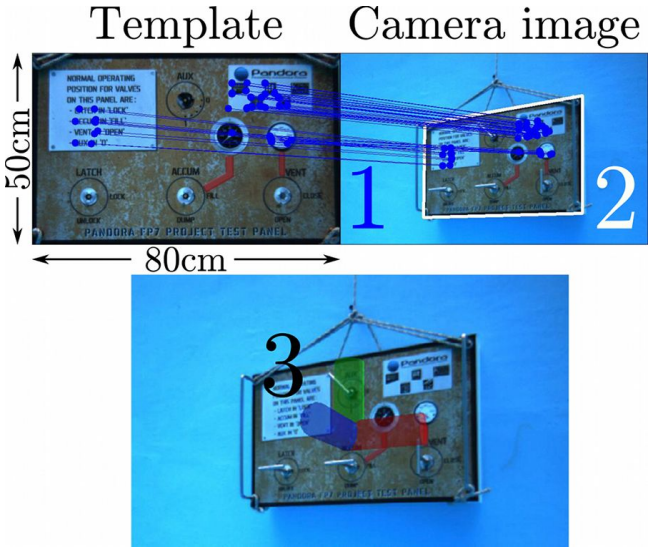


Fig. 4. Detection of the landmark consists of 3 steps: 1) Match keypoints between the template and camera image. 2) Estimate corners of the panel in the camera image. 3) Estimate the translation and rotation of the template in the image by using the known geometry of the landmark.

During the initialization phase, the state vector contains only the vehicle position and linear velocity. There are no landmarks in the state vector. The first time a detection algorithm observes a landmark that it is able to identify, the landmark is introduced in the state vector by composing its position with respect to the vehicle (measured by the visual detector) using the vehicle position with respect to $\{I\}$ (contained in the state vector). To avoid introducing erroneous landmarks in the map, the diagram shown in Fig. 3 is used. To introduce a new landmark in the map, it has to be seen at least five times and the variance of the last five measures has to be lower than a manually defined threshold. A first in first out (FIFO) queue is used for this procedure.

D. Vision-Based landmark detectors

Two algorithms are proposed in this paper to compute the position of a known landmark using vision, by comparing the images from the camera against an *a priori* known template of the landmark itself.

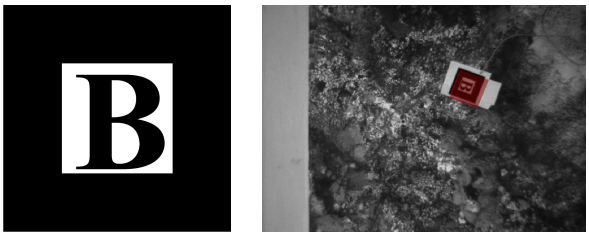
The first one detects and matches features between the camera image and a template. It is possible to detect the presence of the landmark, as well as accurately estimate the pose when a sufficient number of features are matched. In this algorithm, the oriented FAST and rotated BRIEF (ORB) [10] feature extractor has been chosen for its suitability to real-time applications. The ORB feature extractor relies on features from accelerated segment test (FAST) corner detection [11] to detect keypoints in the image. These are obvious features to detect on man-made structures and can be detected very quickly. Moreover, there is a (binary) descriptor vector of the keypoint based on binary robust independent elementary features (BRIEF) [12]. Differences between descriptors can be calculated rapidly, allowing real-time matching of keypoints at higher image frame-rates when compared to other commonly used feature extractors such as scale invariant feature transform (SIFT) [13] and speeded-up robust features (SURF) [14].

Fig. 4 illustrates the matching between a template and an image received from the camera. A minimum number of keypoints must be matched between the template and the camera image to satisfy the landmark detection requirement. A low number of matched keypoints indicates that the landmark is not in the camera field of view. The correspondences between the template and camera image can be used to compute the transformation (or homography) of the template image to the detected landmark in the camera image. This allows to compute the image-coordinates of the corners of the template in the camera image. Then, using the known geometry of the landmark and the camera matrix, the pose of the landmark in the camera coordinate system and consequently in the vehicle coordinate system can be determined.

The second algorithm used to compute the position of a landmark relies on the ARToolkit software library [15] to identify, detect and track marks using a monocular camera (see Fig. 5). To avoid the data association problem in the EKF-SLAM algorithm, each mark is different. The algorithm is robust and works in real-time, however, artificial marks like Fig. 5(a) have to be placed on the environment. To compute the measurement noise covariance the following estimation is used:

$$R = (I_{3\times 3} \cdot d^2) \cdot \mathbf{c}, \tag{20}$$

where d is the estimated distance between the camera and the landmark and $\mathbf{c}$ a vector of fixed coefficients found empirically.



(a) (b)

Fig. 5. (a) ARToolkit landmark and (b) ARToolkit algorithm detecting landmark on the bottom of the water tank.

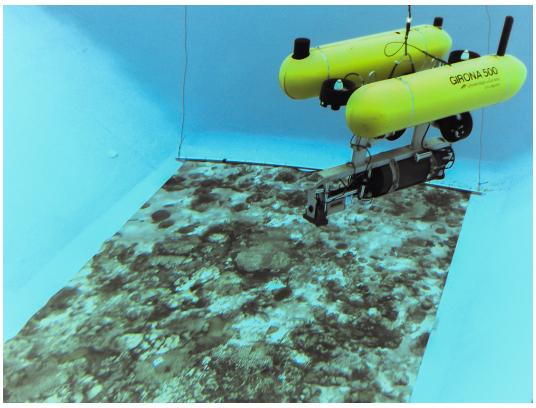


Fig. 6. Girona 500 AUV at the water tank with the poster deployed for ground-truth computation.

III. Experiments

To test the proposed vision-based navigation system, several experiments have been performed with Girona500 AUV [8] in a water tank of $16 \times 8 \times 5$ meters.

A. Setup

The vehicle has been equipped with an IMU that measures orientation and orientation rate at 20Hz. GPS and depth sensors provide pose updates in $(x,\ y,\ z)$ at the surface and (z) when the vehicle is submerged at 2Hz. Velocity updates $(u,\ v,\ w)$ are given by a low cost DVL at 2.7Hz and, finally, a monocular down-looking camera has been used to gather images at 1.875Hz. All the computations have been performed in real-time with the on-board computer, an ultra low voltage dual core processor running at 1.2GHz with 2GB of RAM.

In order to compare the proposed EKF-SLAM with a ground-truth, a 7.1 $\times$ 3.5 meters poster has been placed at the bottom of the water tank (see Fig. 6). It is possible to obtain estimates of absolute camera poses, together with their uncertainty, by registering camera images to the original image of the poster. These estimates then be used as ground-truth. The estimation method uses a parameterization for the pose comprising the 6 DoF in the form

$$\Theta = \left[\begin{array}{cccccc} \alpha & \beta & \gamma & {}^{W}_{C}\mathbf{t}_x & {}^{W}_{C}\mathbf{t}_y & {}^{W}_{C}\mathbf{t}_z \end{array} \right]^T , \qquad (21)$$

where the 3 first elements encode roll, pitch and heading,

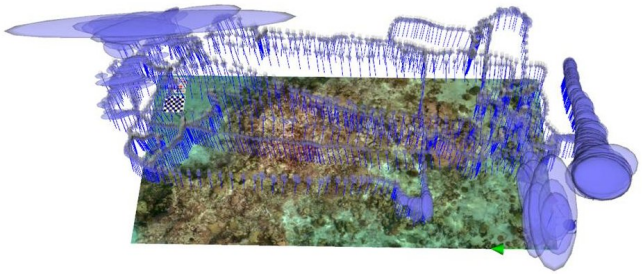


Fig. 7. Representation of the ground-truth trajectory and associated uncertainty. The ellipsoids contain 50% uncertainty and have been enlarged 5$\times$ for visibility reasons. The poses with large uncertainties correspond to locations where the camera was only imaging part of the poster and were not used as ground-truth.

and the last are translations with respect to a local reference frame.

The method relies on the identification of point correspondences using feature-based robust matching [16]. The outcome of the matching are two lists of correspondences $\mathbf{x}^i$ and $\mathbf{x}^m$ of points in the camera image and the poster image respectively. These lists are related by an observation equation in the form

$$\mathbf{x}^i = \mathbf{Q}(\mathbf{x}^m, \Theta) + \varepsilon \ .$$

where $\mathbf{Q}$ is a projection function that takes into account the pose, the camera calibration and the scale of the mosaic poster, and ε is assumed to be Gaussian random noise.

The maximum likelihood estimate of Θ is

$$\Theta_{ML} = \arg\min_{\Theta} \left\| \mathbf{x}^i - \mathbf{Q}\left(\mathbf{x}^m, \Theta\right) \right\|^2 \ . \qquad (22)$$

This minimization is carried out using a non-linear least squares algorithm [17]. The initial value for Θ is provided by an algebraic solution from the elements of the camera to poster homography [18]. The uncertainty in the pose is computed following the approach described in [19]. A representation of the ground-truth trajectory is given in Fig. 7.

B. Results

Girona 500 AUV has been tele-operated at a constant depth of 2.25 meters for about 10 minutes. The continuous line shown in Fig. 8(a) shows the trajectory estimated by the vehicle according the EKF-SLAM algorithm and the two stars point where landmarks have been detected. Only 10% of gathered images contain one of the two landmarks. The ground-truth trajectory computed off-line comparing camera images with the known poster is also plotted in Fig. 8(a) as a dotted line.

Using the logs gathered in this experiment, the EKF-SLAM algorithm has been executed again removing all *landmark_update* messages, thus it has behaved like a dead reckoning algorithm. Fig. 8(b) shows the trajectory estimated in this second execution without landmark updates against the same ground-truth. It is easy to see that, despite there are only two landmarks in the setup and most of the time these landmarks are not present in the imagery, trajectory

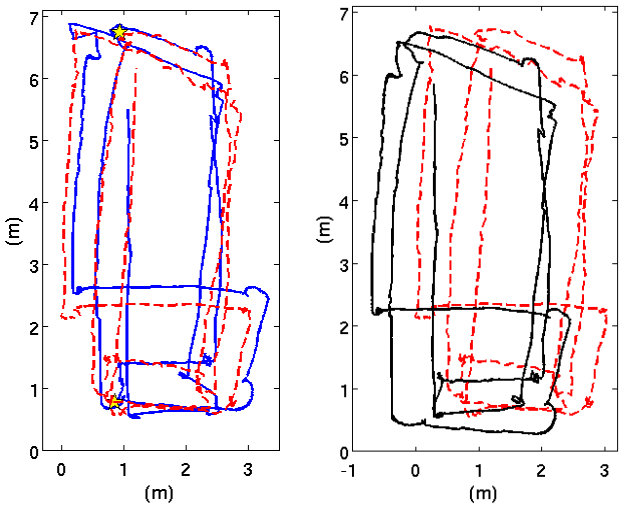


Fig. 8. (a) EKF-SLAM trajectory (continuous line) against ground-truth (dotted line) and (b) dead reckoning trajectory (continuous line) against ground-truth (dotted line). Yellow stars point the position of landmarks.

shown in Fig. 8(b) drifts more than the one presented in Fig. 8(a).

Rather than the Euclidean distance between the estimated trajectories and ground-truth, the Hellinger distance has been used to quantify the error [20]. The Hellinger distance is a general distance on probability distributions. Here, the ground-truth estimated from the poster takes the form of a multivariate Gaussian distribution computed using a maximum likelihood estimator by matching features between the mosaic and camera images.

By computing the distance between the ground-truth distribution and the estimated vehicle pose distribution, we account for the uncertainty inherent in the estimate as well as the ground-truth. The Hellinger distance between two distributions $f(\cdot)$ and $g(\cdot)$ is given by

$$d_H(f,g) = \frac{1}{2}\int \left(\sqrt{f(\mathbf{x})} - \sqrt{g(\mathbf{x})}\right)^2 d\mathbf{x} \in [0,1]. \quad (23)$$

Here, the distributions from the ground-truth, dead reckoning and EKF-SLAM are all multivariate Gaussian. Then, for $f = \mathcal{N}(\mathbf{x}, \Sigma_x)$ and $g = \mathcal{N}(\mathbf{y}, \Sigma_y)$, the Hellinger distance is given by the closed form expression

$$d_H(f,g) = 1 - \sqrt{\frac{\sqrt{\det(\Sigma_x \Sigma_y)}}{\det[\frac{1}{2}(\Sigma_x + \Sigma_y)]}}\exp(\varepsilon), \quad (24)$$

$$\varepsilon = \left[-\frac{1}{4}(\mathbf{x}-\mathbf{y})^T(\Sigma_x + \Sigma_y)^{-1}(\mathbf{x}-\mathbf{y})\right]. \quad (25)$$

Fig. 9 illustrates the error in the estimated trajectories from the dead reckoning and EKF-SLAM. Since the uncertainty of the dead reckoning constantly increases, the distance decreases only slightly if the estimated position is close to the ground-truth. On the other hand, the uncertainty in the estimate from the EKF-SLAM reduces when landmarks are detected leading to reduction in the error at these points in time (see Fig. 10).

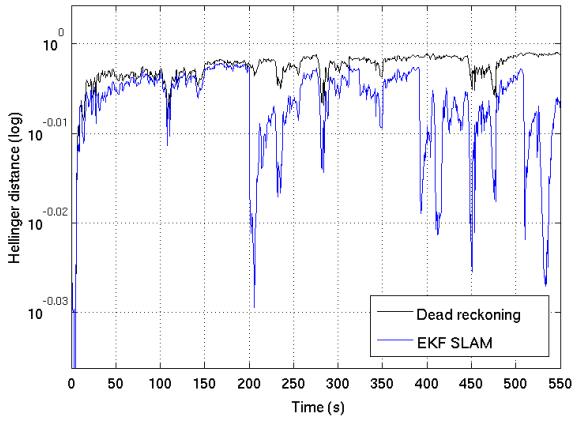


Fig. 9. Error in the estimated trajectories from the dead reckoning and EKF-SLAM.

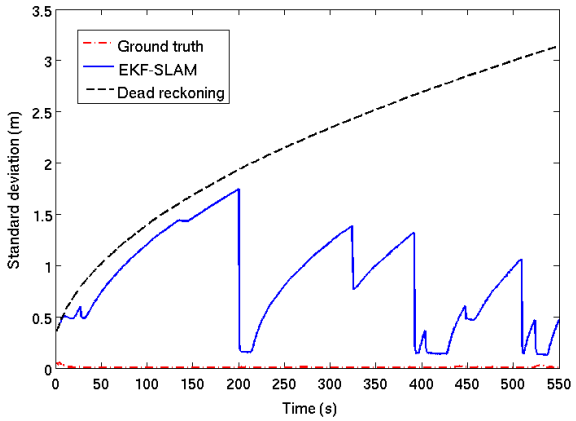


Fig. 10. Comparison of standard deviation for axis X over time between: dead reckoning that always increases, EKF-SLAM that increases at the same rate but decreases when a landmark is detected and the ground-truth that is close to zero.

IV. Conclusions and Future Works

The aim of this paper is to present a modular localization and mapping system for intervention AUVs working in semi-structured environments with known landmarks.

The system has been divided in several modules to make it as generic as possible and to simplify its reutilization for different vehicles and sensor suites. Main modules are: *navigation_adapter*, visual detectors, and the *pose_ekf_slam*. The *navigaton_adapter* transforms navigation sensors custom messages to generic ROS messages. It makes all navigation sensors data be dextrorotatory and follow the metric unit system and, moreover, it transforms these measurements and its covariance from sensor's frame to vehicle's frame. Two visual detectors have been presented to compute the position of a known marks by comparing the images from the camera against an *a priori* known template of the mark itself. Both systems are suitable for real-time applications. Finally, an EKF-SLAM algorithm has been detailed. The algorithm estimates the vehicle position and linear velocity and maps the position of several landmarks identified by the visual detectors.

To test the whole system, real-time experiments have been performed with Girona 500 AUV in a water tank with a known 7.1 $\times$ 3.5 meters poster placed on its bottom. Results prove the improvement on the navigation quality of the EKF-SLAM with respect to a dead reckoning algorithm by comparing them with a ground-truth obtained off-line by matching the gathered images with the known poster.

As a future work, a new landmark detector based on acoustic imagery could be easily adapted to the current setup. A preliminary work has been done identifying the links of an underwater chain using a forward looking sonar [21]. Moreover, the introduction of each landmark orientation in the state vector instead of only its position should be studied.

REFERENCES

[1] "PANDORA FP7 project," http://persistentautonomy.com, 2012, [Online; accessed April-2013]. [Online]. Available: http://persistentautonomy.com/

[2] H. A., "Auv uptake in the offshore industry: maintaining a forward momentum," in *Unmanned Underwater Vehicle Showcase*, 2003.

[3] T. Palmer, D. Ribas, Ridao., and A. Aggelos, "Vision based localization system for AUV docking on subsea intervention panels," in *Oceans*, 2009.

[4] D. Ribas, P. Ridao, and J. Neira, "Underwater SLAM for structured environments using an imaging sonar," *Springer Tracts in Advanced Robotics*, 2010.

[5] P. Newman and J. Leonard, "Pure range-only sub-sea slam," in *IEEE International Conference on Robotics an Automation*, 2003.

[6] J. Salvi, Y. Petillot, S. Thomas, and J. Aulinas, "Visual slam for underwater vehicles using video velocity log and natural landmarks," in *Oceans*, 2008.

[7] WillowGarage, "Robot Operating System," http://www.ros.org, 2007, [Online; accessed April-2013]. [Online]. Available: http://www.ros.org

[8] D. Ribas, N. Palomeras, P. Ridao, M. Carreras, and A. Mallios, "Girona 500 AUV, from survey to intervention," *IEEE/ASME Transactions on Mechatronics*, vol. 17(1), pp. 46–53, February 2012.

[9] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Tech. Rep., 1995.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, nov. 2011, pp. 2564–2571.

[11] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *In European Conference on Computer Vision*, 2006, pp. 430–443.

[12] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: binary robust independent elementary features," in *Proceedings of the 11th European conference on Computer vision: Part IV*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 778–792. [Online]. Available: http://dl.acm.org/citation.cfm?id=1888089.1888148

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, June 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[15] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *2nd International Workshop on Augmented Reality (IWAR 99)*, October 1999, San Francisco, USA.

[16] N. Gracias, "Mosaic–based Visual Navigation for Autonomous Underwater Vehicles," Ph.D. dissertation, Instituto Superior Técnico, Lisbon, Portugal, June 2003. [Online]. Available: vislab.isr.ist.utl.pt

[17] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[18] N. Gracias and J. Santos-Victor, "Trajectory reconstruction with uncertainty estimation using mosaic registration," *Robotics and Autonomous Systems*, vol. 35, pp. 163–177, July 2001.

[19] R. Haralick, "Propagating covariance in computer vision," in *Proc. of the Workshop on Performance Characteristics of Vision Algorithms*, Cambridge, UK, April 1996.

[20] S. Nagappa, D. Clark, and R. Mahler, "Incorporating track uncertainty into the ospa metric," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, 2011, pp. 1–8.

[21] N. Hurtos, N. Palomeras, and J. Salvi, "Automatic detection of underwater chain links using a forward-looking sonar," in *OCEANS'13 MTS/IEEE*, 2013.