**2011 IEEE/RSJ International Conference on Intelligent Robots and Systems**
**September 25-30, 2011. San Francisco, CA, USA**

# State Estimation of an Underwater Robot Using Visual and Inertial Information

Florian Shkurti, Ioannis Rekleitis, Milena Scaccia and Gregory Dudek

*Abstract*— **This paper presents an adaptation of a vision and inertial-based state estimation algorithm for use in an underwater robot. The proposed approach combines information from an Inertial Measurement Unit (IMU) in the form of linear accelerations and angular velocities, depth data from a pressure sensor, and feature tracking from a monocular downward facing camera to estimate the 6DOF pose of the vehicle. To validate the approach, we present extensive experimental results from field trials conducted in underwater environments with varying lighting and visibility conditions, and we demonstrate successful application of the technique underwater.**

## I. Introduction

Algorithms for estimating the position and orientation (pose) of a mobile robot are considered significant enablers for robot autonomy, and thus, a large spectrum of robotics research has focused on them. In particular, vision-based pose estimation techniques have become quite popular in recent years, not only due to the wide availability of cheap camera sensors, but mainly due to a series of technical advances and successful demonstrations of Structure from Motion and Visual Odometry systems. Illustrative examples in the former category include Davison et al.'s early [1] and more recent work [2] on real time accurate 3D structure reconstruction and motion estimation of a monocular camera, moving in a constrained indoor space. In the latter category Nister [3], Konolige [4], Furgale and Barfoot [5] have shown online visual odometry systems that are capable of accurately localizing terrestrial robots over tens-of-kilometers-long trajectories. In addition, vision-aided localization techniques also include appearance-based localization algorithms, which do not attempt to estimate the 6DOF pose of the robot, but rather to identify the place at which the robot is located. The seminal work of Cummins and Newman [6], which performed place recognition over a thousand-kilometer-long trajectory, exemplifies this line of research.

Our goal is to perform 6DOF pose estimation for the Aqua family of amphibious robots [7]. These hexapod robots are equipped with three IEEE-1394 IIDC cameras, a low-cost IMU, and a pressure sensor which is used to provide noisy measurements of depth from the surface of the water. Their swimming motion is a product of six paddles, which oscillate synchronously in two groups of three.

The majority of the localization work mentioned previously does not deal with the underwater domain, and often assumes the existence of a mathematical motion model of the vehicle at hand. In our case, such a model is highly nonlinear, hard to justify, and susceptible to deviations due to currents and other factors. In general, underwater environments are more challenging than most of their indoor counterparts for the following reasons:

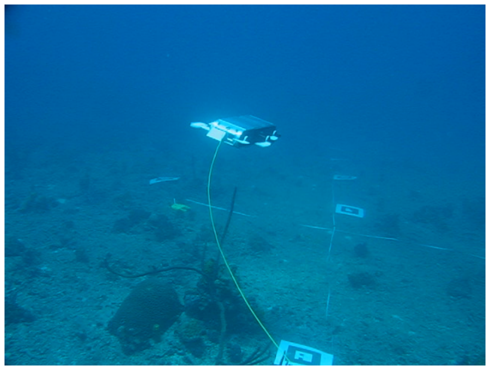


Fig. 1. The Aqua vehicle starting a dataset collection run

(a) *They are prone to rapid changes in lighting conditions*. The canonical example that illustrates this is the presence of caustic patterns, which are due to refraction, non-uniform reflection, and penetration of light when it transitions from air to the rippling surface of the water [8].

(b) *They often have limited visibility*. This is due to many factors, some of which include light scattering from suspended plankton and other matter, which cause blurring and "snow effects". Another reason involves the incident angle at which light rays hit the surface of the water. Smaller angles lead to less visibility.

(c) *They impose loss of contrast and colour information with depth*. As light travels at increasing depths, different parts of its spectrum are absorbed. Red is the first color that is seen as black, and eventually orange, yellow, green and blue follow [9]. This absorption sequence refers to clear water, and is not necessarily true for other types.

Given the above constraints, pose estimation algorithms that rely on IMU and visual data are promising, because the IMU provides a noisy estimate of our vehicle's dynamics even in the presence of strong currents, while the camera imposes corrective motion constraints on the drifting IMU pose estimate, which is obtained by integration. In this paper we present an adaptation of the algorithm presented by Mourikis and Roumeliotis [10], [11] for the underwater domain, and we validate the approach through offline experiments on underwater datasets that capture varying scenes and lighting conditions.

The authors would also like to thank NSERC for its generous support.
F. Shkurti, I. Rekleitis, M. Scaccia and G. Dudek are with the School of Computer Science, McGill University, Montreal, QC, Canada
{florian, yiannis, mscacc, dudek}@cim.mcgill.ca

## II. Related Work

The work of Mourikis and Roumeliotis uses an Extended Kalman Filter (EKF) estimator that integrates the incoming IMU linear acceleration and rotational velocity measurements in order to propagate the state of the robot. Integration of this data also propagates the noise, which quickly causes the IMU estimate alone to drift. To correct this, the authors rely on the constraints imposed by the tracking of visual features from frame to frame, using a monocular camera. Once a feature ceases being observed, its 3D position is estimated via bundle adjustment and regarded to be 'true'. The residual between the expected 3D feature position and the 'true' position determines the correction applied to the state and the covariance. One of the technically appealing attributes of this system is that the landmarks are not part of the state, which caused dimensionality explosion in the early formulations of the SLAM (Simultaneous Localization and Mapping) problem. Furthermore, the filter was shown to perform well even with a fixed, small number of past camera poses, in essence making the state vector and the covariance of fixed size. The experimental results of this work demonstrated highly accurate estimated trajectories with error of just 0.31% of a 3.2km trajectory.

One of the differences between their configuration and ours is that we are using a low-cost IMU, which is very susceptible to magnetic interference, as it is placed very close to moving motor parts, housed in a densely-assembled robot interior, which is characterized by high temperatures when the robot is operating. All these factors significantly impact our IMU's accuracy.

More recent work by Jones and Soatto [12] reports even more large-scale results, with 0.5% error of a 30km trajectory, which was traversed in an urban environment. Aside from the significance of the low drift given the length of the trajectory, another novelty of this work is the fact that it autocalibrates the transformation between the camera and the IMU, as well as the gravity vector, which are two major sources of systematic bias in this class of algorithms.

Furthermore, Kelly et al. [13] presented a visual and inertial pose estimation algorithm for an autonomous helicopter, which uses images from a stereo camera and IMU measurements to perform inertial-aided visual odometry. The authors report errors of less than 0.5% over a 400m long trajectory.

In the underwater domain, Corke et. al [14] presented experimental results of an underwater robot performing stereo visual odometry (without inertial data) using the Harris feature detector and the normalized cross-correlation similarity measure (ZNCC). They reported that, after outlier rejection, 10 to 50 features were tracked from frame to frame, and those were sufficient for the stereo reconstruction of the robot's 3D trajectory, which was a square of area 30m by 30m. Other vision-based SLAM systems that have used visual odometry underwater include [15]–[17].

Eustice et. al [18] showed successful use of the sparse information filter to combine visual and inertial data, while performing a robotic survey mapping of the surface of the RMS Titanic. They also presented a technique for maintaining the consistency bounds of the filter's covariance.

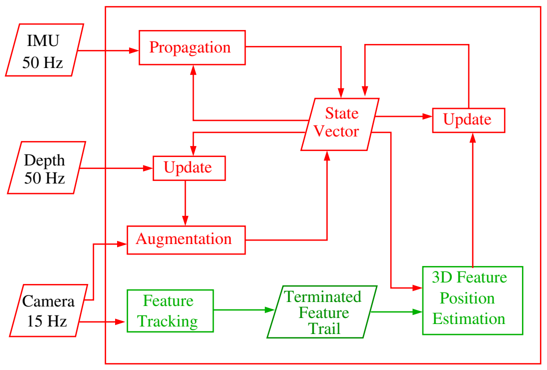


Fig. 2. An overview of the 6DOF state estimation algorithm. The state vector appears in Eq. (1)

There are, of course, other technologies for pose estimation underwater including the use of beacons or Doppler-based motion estimation [19]. In this paper we examine the extent of what is feasible using only passive sensing, and specifically vision and proprioception alone. One advantage, incidentally, of purely passive sensing is that it is both energy efficient and discreet towards marine ecosystems.

## III. 6DOF State Estimation of an Autonomous Underwater Vehicle

The state vector at time $t_k$ that we are interested in estimating has the following form:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{X}_{IMU_k} & {}^{C_1}_{G}q & {}^{G}\mathbf{p}_{C_1} & \ldots & {}^{C_N}_{G}q & {}^{G}\mathbf{p}_{C_N} \end{bmatrix}^T \quad (1)$$

where $\mathbf{X}_{IMU_k}$ is a $16 \times 1$ vector expressing the state of the IMU, and the pairs $\{{}^{C_i}_{G}q \quad {}^{G}\mathbf{p}_{C_i}\}$ represent the transformations between the global frame and the $i^{th}$ camera frame. In particular, ${}^{C_i}_{G}q$ is a unit quaternion [1], that expresses the rotation from the global frame to the frame of camera $C_i$, and ${}^{G}\mathbf{p}_{C_i}$ is the origin of camera frame $C_i$ given in coordinates of the global reference frame $G$. In total, the state vector $\mathbf{X}_k$ is of length $16 + 7N$, where $N$ is the number of camera frames that are being tracked. As will be explained in more detail below, camera frames are appended to the state vector every time an image is recorded. Assuming the vehicle is not standing still, most features that are tracked will exit the field of view of the robot's camera after some number of frames, which is the main reason why we can bound $N$ above by $N_{max}$ (we set this to 32). The IMU state vector, consists of the following:

$$\mathbf{X}_{IMU_k} = \begin{bmatrix} {}^{I}_{G}q & \mathbf{b}_g & {}^{G}\mathbf{v}_I & \mathbf{b}_a & {}^{G}\mathbf{p}_I \end{bmatrix} \quad (2)$$

where ${}^{I}_{G}q$ is a quaternion that expresses the rotation from the global frame to the IMU frame, $\mathbf{b}_g$ is the bias of the gyroscope, ${}^{G}\mathbf{v}_I$ is the velocity of the IMU frame in coordinates of the global frame, $\mathbf{b}_a$ is the bias of the accelerometer, and ${}^{G}\mathbf{p}_I$ is the origin of the IMU frame in global coordinates.

[1]Recall that quaternions provide a well-behaved parameterization of rotation. We are following the JPL formulation of quaternions; see [20], [21]

Since the state vector contains quaternions and the covariance matrix is an expectation of a product of vectors, a linearized representation of error quaternions is required. To that end, we follow the small-angle approximation used by Mourikis and Roumeliotis, whereby a quaternion correction $\delta q = [\mathbf{k} sin(\delta\theta/2) \quad cos(\delta\theta/2)] \simeq [\mathbf{k}\delta\theta/2 \quad 1] = [\delta\boldsymbol{\theta}/2 \quad 1]$ determines the approximated difference in rotation between two frames. So, we take the vector representation of the error quaternion to be $\delta\boldsymbol{\theta}$. [2]

The algorithm consists of four steps (see Fig. 2):

(a) In the propagation step the linear acceleration and angular velocity measurements from the IMU are integrated, in order to provide a short-term, noisy estimate of the robot's motion. Only the IMU state vector, $\mathbf{X}_{IMU_k}$ and the corresponding IMU covariance matrix are modified during this phase; the state and covariance of the camera frames remains intact.

(b) Depth measurements from a pressure sensor are available at the same rate as the IMU, however, we sample them at the same rate as the camera and we perform a depth-based update.

(c) When the camera records an image, the camera frame is appended to the state vector and the covariance matrix size grows accordingly. Also, current features are matched to previous features. If the maximum number of camera frames in the state vector has been reached, then old frames are pruned, provided no feature trail is visible from said frames.

(d) If there exist feature tracks that ceased to appear in the current image, the 3D position of said features is estimated and is regarded as 'true'. Assuming said features are not deemed to be outliers, the residual between the 'true' feature positions and the estimated feature positions gives rise to the vision-based update, which corrects the entire state vector, and in particular, the IMU integration drift.

## A. Propagation

Given IMU measurements $\boldsymbol{\omega}_m$ and $\boldsymbol{\alpha}_m$ for the angular velocity and linear acceleration of the IMU frame, respectively, we model the errors affecting the measurements as follows:

$$\begin{aligned} \boldsymbol{\omega}_m &= {}^I\boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \\ \boldsymbol{\alpha}_m &= \mathbf{R}({}^I_G q)({}^G\boldsymbol{\alpha} - {}^G\mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a \end{aligned} \tag{3}$$

where ${}^G\boldsymbol{\alpha}$ and ${}^I\boldsymbol{\omega}$ are the true values, the noise is white Gaussian, and the biases are modeled as Brownian motion processes with $\dot{\mathbf{b}}_g = \mathbf{0}$, $\dot{\mathbf{b}}_a = \mathbf{0}$, and $\mathbf{R}()$ denotes the rotation matrix resulting from the corresponding quaternion. The linear acceleration and angular velocity estimates used in the IMU state integration are $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}_m - \hat{\boldsymbol{b}}_a$ and $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g$. With that in mind, we perform the integration of the IMU state using 4th order Runge-Kutta integrator, modified to include a zeroth-order quaternion integrator [21] for the rotation of the global frame to the IMU frame:

[2]There exist alternative ways of representing quaternion error; see for instance [22]–[24]

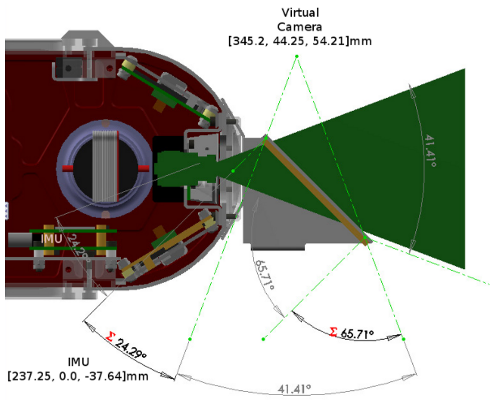


Fig. 3. A schematic of the relationship between the IMU coordinate frame and the mounted camera coordinate frame.

$$\begin{aligned} {}^I_G\hat{q}_{t_{k+1}} &= \begin{bmatrix} \frac{\hat{\boldsymbol{\omega}}}{\|\hat{\boldsymbol{\omega}}\|} sin(\|\hat{\boldsymbol{\omega}}\| \frac{(t_{k+1}-t_k)}{2}) \\ cos(\|\hat{\boldsymbol{\omega}}\| \frac{(t_{k+1}-t_k)}{2}) \end{bmatrix} \otimes {}^I_G\hat{q}_{t_k} \\ \dot{\hat{\mathbf{b}}}_g &= \mathbf{0}_{3\times 1}\ , \quad {}^G\dot{\hat{\mathbf{v}}}_I = \mathbf{R}^T({}^I_G\hat{q})\hat{\boldsymbol{\alpha}} + {}^G\mathbf{g} \\ \dot{\hat{\mathbf{b}}}_a &= \mathbf{0}_{3\times 1}\ , \quad {}^G\dot{\hat{\mathbf{p}}}_I = {}^G\hat{\mathbf{v}}_I \end{aligned} \tag{4}$$

We have observed that 200-500 iterations of the Runge-Kutta integrator are sufficient, while less than 50 iterations produce noticeable numerical inaccuracies in terms of integration residual. The propagation of the covariance via integration is fully presented in [10]. At this point it is worth mentioning that ${}^G\mathbf{g}$ is estimated offline by averaging sensor readings while the robot is still. Initialization of ${}^I_G\hat{q}$ can become involved in cases where the algorithm starts when the robot is in motion and accelerating, since the gravity vector might not be observable from the acceleration values.

## B. State Augmentation

Our vehicle is equipped with two cameras facing forward and one camera facing backwards. As the robot swims over the seafloor, most of the structures are below it. Therefore, in order to observe the seafloor a mirror is placed at a $45^\circ$ angle in front of the back camera. This configuration results in a virtual downward-looking camera located behind the robot; see Fig. 3. As a first step, the robot's design specifications have been used to estimate the coordinate transformation $\{{}^C_I q, {}^I\mathbf{p}_C\}$ from the IMU frame to the virtual camera coordinate frame. However, small errors in this transformation are sources of systematic biases in the estimation process. That is why the use of a IMU-camera transformation calibration process is recommended, such as [24], [25].

When a new image is recorded, the transformation of the global frame to the new camera frame, ${}^C_G\hat{q} = {}^C_I q \otimes {}^I_G\hat{q}$, and ${}^G\hat{\mathbf{p}}_C = {}^G\hat{\mathbf{p}}_I + \mathbf{R}^T({}^I_G\hat{q})^I\mathbf{p}_C$ is appended to the state vector, and the covariance matrix is modified accordingly.

## C. Feature Tracking

In previous work [26] we experimented with a large array of potential feature detectors, such as SURF [27], SIFT [28], FAST [29], CenSurE [30], Shi-Tomasi [31], but also with matching strategies that include Approximate Nearest Neighbors [32] and normalized cross-correlation, as well as with the Kanade-Lucas tracker [33]. We have found SURF with Approximate Nearest Neighbor matching to be very precise, however, when the number of detected features is high (e.g. above 1000), feature tracking cannot be done in real-time. On the other hand, Shi-Tomasi features matched using ZNCC scores of image patches searched along the epipolar lines is more susceptible to false matches, but can be done in real-time. At the moment, we are using SURF features with Approximate Nearest neighbor matching. Our outlier rejection is aggressive: we use the fundamental matrix criterion to eliminate matches that do not satisfy epipolar constraints; we eliminate features that give rise to unusually high residuals in the vision-based update (as described in [11]); and finally, we eliminate features whose position is estimated to be closer than $0.2m$ and farther than $10m$ from the robot. The latter is an environment-specific restriction, because underwater visibility is lost outside that range.

### *D. 3D Feature Position Estimation*

Consider a single feature, $f$, that has been tracked in $n$ consecutive camera frames, $C_1, C_2, ..., C_n$. Let us denote ${}^{C_i}\mathbf{p}_f = \begin{bmatrix} {}^{C_i}X_f & {}^{C_i}Y_f & {}^{C_i}Z_f \end{bmatrix}$ to be the 3D position of feature $f$ expressed in camera frame $C_i$ coordinates. We are interested in estimating this as accurately as possible, because the vision-based update will depend on it. Then, we can write the following:

$$\begin{aligned} {}^{C_i}\mathbf{p}_f &= \mathbf{R}({}^{C_i}_{C_1}q)\,{}^{C_1}\mathbf{p}_f + {}^{C_i}\mathbf{p}_{C_1} \\ &= {}^{C_1}Z_f \left( \mathbf{R}({}^{C_i}_{C_1}q) \begin{bmatrix} \frac{{}^{C_1}X_f}{{}^{C_1}Z_f} & \frac{{}^{C_1}Y_f}{{}^{C_1}Z_f} & 1 \end{bmatrix}^T + \frac{1}{{}^{C_1}Z_f}\,{}^{C_i}\mathbf{p}_{C_1} \right) \end{aligned}$$

If we let $\alpha_f = \frac{{}^{C_1}X_f}{{}^{C_1}Z_f}$, $\beta_f = \frac{{}^{C_1}Y_f}{{}^{C_1}Z_f}$, and $\gamma_f = \frac{1}{{}^{C_1}Z_f}$ then

$$\begin{aligned} {}^{C_i}\mathbf{p}_f &= {}^{C_1}Z_f \left( \mathbf{R}({}^{C_i}_{C_1}q) \begin{bmatrix} \alpha_f & \beta_f & 1 \end{bmatrix}^T + \gamma_f\,{}^{C_i}\mathbf{p}_{C_1} \right) \\ &= {}^{C_1}Z_f \begin{bmatrix} h_{i1}(\alpha_f, \beta_f, \gamma_f) \\ h_{i2}(\alpha_f, \beta_f, \gamma_f) \\ h_{i3}(\alpha_f, \beta_f, \gamma_f) \end{bmatrix} \end{aligned} \quad (5)$$

Now, assuming a simple pinhole camera model, and disregarding the effects of the camera's calibration matrix, we can model the projection of feature $f$ on the projection plane ${}^{C_i}Z = 1$ as follows:

$$\mathbf{z}_f^{C_i} = \frac{1}{h_{i3}(\alpha_f, \beta_f, \gamma_f)} \begin{bmatrix} h_{i1}(\alpha_f, \beta_f, \gamma_f) \\ h_{i2}(\alpha_f, \beta_f, \gamma_f) \end{bmatrix} + \mathbf{n}_f^{C_i} \quad (6)$$

where $\mathbf{z}_f^{C_i}$ is the $2 \times 1$ measurement, and $\mathbf{n}_f^{C_i}$ is noise associated with the process, due to miscalibration of the camera, motion blur, and other factors. If we stack the measurements from all cameras into a single $2n \times 1$ vector $\mathbf{z}_f$ and similarly for the projection functions $\mathbf{h}_i$ into a $2n \times 1$ vector $\mathbf{h}_f$, then we will have expressed the problem of estimating the 3D position of a feature as a nonlinear least-squares problem with 3 unknowns:

$$\underset{\alpha_f, \beta_f, \gamma_f}{\operatorname{argmin}} \|\mathbf{z}_f - \mathbf{h}_f(\alpha_f, \beta_f, \gamma_f)\| \quad (7)$$

Provided we have at least 2 measurements of feature $f$, i.e. provided we track it in at least 2 frames, we use the Levenberg-Marquardt nonlinear optimization algorithm in order to get an estimate ${}^{C_1}\hat{\mathbf{p}}_f$ of the true solution. Then, the estimated feature position in global coordinates can be obtained by:

$${}^{G}\hat{\mathbf{p}}_f = \frac{1}{\gamma_f}\mathbf{R}^T({}^{C_1}_{G}\hat{q}) \begin{bmatrix} \alpha_f & \beta_f & 1 \end{bmatrix}^T + {}^{G}\hat{\mathbf{p}}_{C_1} \quad (8)$$

One problem with this approach is that nonlinear optimization algorithms do not guarantee a global minimum, only a local one, provided they converge. Another problem is that if feature $f$ is recorded around the same pixel location in all the frames in which it appears, then the measurements we will get are going to be linearly dependent, thus providing no information about the depth of $f$. In other words, feature tracks that have small baseline have to be considered outliers, unless we exploit other local information around the feature to infer its depth. Another potential pitfall is that the inter-camera transformations $\{{}^{C_i}_{C_1}q, {}^{C_i}\mathbf{p}_{C_1}\}$ might be themselves noisy, which will also affect the solution of the least squares problem.

### *E. Vision-based Update*

Consider again a single feature $f$, which has been tracked in $n$ consecutive camera frames, $C_1, C_2, ..., C_n$, but stopped being tracked at the current frame, $C_{n+1}$. In this case we initiate the vision-based update step. After having estimated the 3D position of feature $f$ in global coordinates, we expect that its projection on the image plane of camera frame $C_i$, according to the pinhole camera model, will be:

$$\hat{\mathbf{z}}_f^{C_i} = \begin{bmatrix} \frac{{}^{C_i}\hat{X}_f}{{}^{C_i}\hat{Z}_f} & \frac{{}^{C_i}\hat{Y}_f}{{}^{C_i}\hat{Z}_f} \end{bmatrix} \quad (9)$$

where

$$\begin{bmatrix} {}^{C_i}\hat{X}_f & {}^{C_i}\hat{Y}_f & {}^{C_i}\hat{Z}_f \end{bmatrix}^T = \mathbf{R}({}^{C_i}_{G}\hat{q})({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_{C_i}) \quad (10)$$

The actual measurement obtained from the feature tracks, on the other hand, is $\mathbf{z}_f^{C_i}$, so for a single feature $f$ viewed from a camera frame $C_i$ this gives rise to the residual

$$\mathbf{r}_f^{C_i} = \mathbf{z}_f^{C_i} - \hat{\mathbf{z}}_f^{C_i} \quad (11)$$

If we take the Taylor expansion of the function $\mathbf{r}_f^{C_i}$ about the point $({}^{C_i}_{G}\hat{q}, {}^{G}\hat{\mathbf{p}}_f, {}^{G}\hat{\mathbf{p}}_{C_i})$ we will get the following linearization:

$$\mathbf{r}_f^{C_i} \simeq \mathbf{H}_f^{C_i}\tilde{\mathbf{X}} + \mathbf{U}_f^{C_i}\,{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n}_f^{C_i} \quad (12)$$

where $\mathbf{H}_f^{C_i}$ and $\mathbf{U}_f^{C_i}$ are the Jacobians of $\mathbf{r}_f^{C_i}$ at the chosen point of linearization. $\mathbf{n}_f^{C_i} \sim \mathcal{N}(0, \mathbf{R}_f^{C_i})$ is the uncertainty in the residual, with $\mathbf{R}_f^{C_i} = \sigma_{im}^2 \mathbf{I}_{2\times 2}$. Essentially, $\sigma_{im}$ models the uncertainty in the camera measurements, and we are currently modeling it as being the same for all camera frames, regardless of whether a particular image has high levels of motion blur, or whether the viewed scene is nearby or far away. The total projection residual from all camera frames in which feature $f$ was tracked is therefore a stack of the individual residuals:

$$\mathbf{r}_f \simeq \mathbf{H}_f \tilde{\mathbf{X}} + \mathbf{U}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{n}_f \tag{13}$$

where $\mathbf{n}_f$ is the total uncertainty of the residual due to feature $f$. We make the assumption that observations of the same feature in consecutive camera frames are statistically independent, which makes the covariance of said uncertainty $\mathbf{R}_f = \sigma_{im}^2 \mathbf{I}_{n\times n}$. The problem with the residual in Eq. (13) is that the state errors $\tilde{\mathbf{X}}$ are correlated with the errors in the feature position estimate ${}^G\tilde{\mathbf{p}}_f$, since the former was used to derive the latter, as described previously. So, using Eq. (13) as the EKF residual will bias the estimates. That is why we can use the closest residual that ignores the first-order dependence on ${}^G\tilde{\mathbf{p}}_f$:

Let $\mathbf{A}_f$ be a matrix such that $\mathbf{A}_f^T \mathbf{U}_f = \mathbf{0}$, in other words, the columns of $\mathbf{A}$ form an orthonormal basis of the null space of $\mathbf{U}_f^T$. While the choice of $\mathbf{A}$ is not unique, the final correction factor applied to the state and covariance will not be affected by this. The residual due to a single feature $f$ that does not depend on the feature position errors is:

$$\mathbf{r}'_f = \mathbf{A}_f^T \mathbf{r}_f \simeq \mathbf{A}_f^T \mathbf{H}_f \tilde{\mathbf{X}} + \mathbf{A}_f^T \mathbf{n}_f = \mathbf{H}'_f \tilde{\mathbf{X}} + \mathbf{n}'_f \tag{14}$$

where $\mathbf{n}'_f \sim \mathcal{N}(0, \mathbf{R}_f)$ due to the column-wise orthonormality of $\mathbf{A}$. Now, the total residual for all the features that ceased to be tracked, and are part of the update, is a stack of the above individual residuals:

$$\mathbf{r}' = \mathbf{H}' \tilde{\mathbf{X}} + \mathbf{n}' \tag{15}$$

At this point we make another assumption, whereby observations of different features are statistically independent, which makes the covariance of $\mathbf{n}'$ be the identity matrix, scaled by $\sigma_{im}^2$. The residual of Eq. (15) is in a form where the EKF framework can be applied, though a numerical speed-up step is possible, as described in [10].

### *F. Depth Sensor Update*

Our amphibious robots are equipped with a pressure sensor that is exposed to the water and measures its hydrostatic pressure. The sensor is calibrated so that its reference point is at the surface of the sea, and its pressure measurements are converted into depth measurements in a linear fashion. We model the incoming data at time $t_k$ of absolute depth from the sea surface as:

$$z_k = d_k + n_k, \quad n_k \sim \mathcal{N}(0, \sigma_{depth}^2). \tag{16}$$

That said, we are interested in the difference between the robot's initial and current depth, in which case our measurements are:

$$\begin{aligned} z'_k &= z_k - z_0 = d_k - d_0 + n_k = {}^G Z_{I_k} + n_k \quad (17)\\ n_k &\sim \mathcal{N}(0, \sigma_{depth}^2) \end{aligned}$$

The EKF state estimate for the depth change is ${}^G\hat{Z}_{I_k}$, so the measurement residual becomes $r_k = {}^G Z_{I,k} - {}^G\hat{Z}_{I,k} + n_k$.

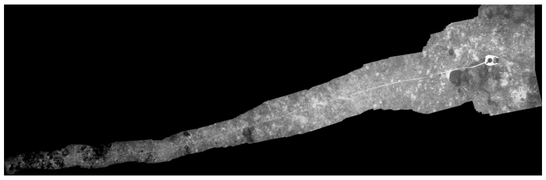


Fig. 4. The near-straight line 30 meter-long trajectory executed in the first experiment.

The covariance matrix of the uncertainty in the residual is

$$\begin{aligned} S_k &= \mathbf{H}_{depth} \mathbf{P}_{k|k-1} \mathbf{H}_{depth}^T + \sigma_{depth}^2 \quad (18)\\ \mathbf{H}_{depth} &= \begin{bmatrix} 0_{1\times 14} & 1 & 0_{1\times 6N} \end{bmatrix} \end{aligned}$$

The Kalman gain is then given by $\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_{depth}^T S_k^{-1}$, so the state correction vector is $\mathbf{K}_k r_k$. This correction will potentially affect the entire state vector. For the vector components of the EKF state additive correction is applied, while for the quaternion components, the following correction is used:

$$\begin{aligned} \delta q &= \begin{bmatrix} \delta\boldsymbol{\theta}/2 & \sqrt{1 - \|\delta\boldsymbol{\theta}\|/4} \end{bmatrix} \quad (19)\\ \hat{q}_{k|k} &= \delta q \otimes \hat{q}_{k|k-1} \end{aligned}$$

It is worth mentioning at this point that the approximation of the error quaternion by $\delta\boldsymbol{\theta}$, as explained previously only holds for small angles, so if $\|\delta\boldsymbol{\theta}\| >= 4$ the estimate will most likely have diverged. Finally, due to the choice of the optimal Kalman gain, the update of the covariance matrix is:

$$\mathbf{P}_{k|k} = (\mathbf{I}_\xi - \mathbf{K}_k \mathbf{H}_{depth}) \mathbf{P}_{k|k-1} \tag{20}$$

where $\xi = 6N + 15$ is the dimension of the covariance matrix. We perform the depth-based update right before the augmentation step, every time an image is recorded. One very important issue that needs to be mentioned is the presence of numerical instabilities when the effects of these updates are compounded, thus for example, making the state covariance non-symmetric. This particular issue is addressed by forcing the symmetry of the covariance after each update.

## IV. Experimental Results

To test the validity of our adaptation of the above mentioned algorithm, we collected underwater datasets with ground truth from the waters of the island of Barbados, and we performed offline experiments to test our implementation. Two of these datasets are going to be presented below together with the state estimates. Images on both datasets were recorded from the back camera at 15Hz with resolution 870 × 520. The IMU data is coming from a MicroStrain 3DM-GX1 MEMS unit, sampled at 50Hz. The first dataset, depicted in Fig. 4, features a straight 30 meter-long trajectory, where the robot moves at approximately 0.2 meters/sec forward, while preserving its depth. The sea bottom is mostly flat, and the robot moves about 2 meters over it. A white 30 meter-long tape has been placed on the bottom, both to provide ground truth for distance travelled, and to facilitate the robot's guided straight line trajectory. The second dataset corresponds to an experiment that took place over the same site, and under the same conditions

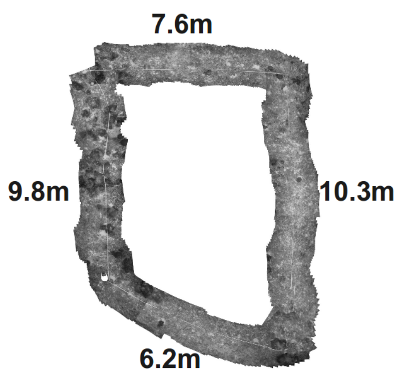
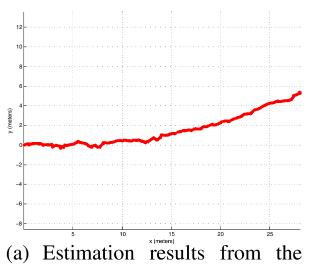
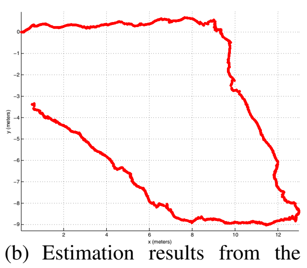


Fig. 5. The second experiment, where the robot is guided to perform a loop closure while recording IMU data and images.

mentioned above, however the shape of the trajectory was a closed loop, as depicted in Fig. 5. The total length was approximately 33 meters, and the side lengths are shown on the figure.



(a) Estimation results from the straight line.



(b) Estimation results from the loop.

Fig. 6. (a) The estimated distance travelled is approximately 29 meters, which is quite accurate. (b) The length of the top segment is overshot by 1m, while the lengths of the remaining segments are estimated accurately. Loop closure is 3 meters apart, due to estimation errors in the yaw angle.

The reconstruction of the straight line trajectory, as shown in Fig. 7 was very accurate in terms of distance travelled, with only 1 meter error over a 30 meter-long trajectory. Lengths were in general, very accurately estimated in the case of the loop, where the loop closure was approached within 3 meters, mainly due to errors in yaw, which is problematic for the gyroscope sensors in low-cost IMUs. Another factor that contributed to the loop results was the presence of motion blur in both datasets, which makes feature matching susceptible to errors.

We observed that the inclusion of the depth sensor measurements improved the estimates of the position's z-axis, provided that the depth-based updates were given more confidence than the accelerometer's propagations.

## V. Discussion

Underwater environments are particularly challenging due to limited visibility, locally self-similar structures, and the unpredictability of motions due to surge and currents. Over the course of our experimentations we have investigated the impact of different parameters in the accuracy and robustness of the state estimation algorithm. An important consideration was always the effect of each parameter to the performance of the algorithm. It is well known that constant velocity motion results makes the state estimation problem unobservable [12]. Contrary to terrestrial experiments, our underwater vehicles are in constant motion due to currents and surge. As such, initial estimation of the gravity vector and the initial orientation of the robot become very challenging. We average initial sensor readings for estimating the gravity vector and the initial orientation and accept the inaccuracies that this condition introduces.

## VI. Conclusions and Future Work

In this paper we presented the adaptation of a 6DOF state estimation algorithm to the underwater domain. Extensive experimentation enabled us to identify the major contributors to instability of the filter. Domain specific knowledge, such as the velocity profile used by our vehicle, allowed us to successfully estimate the pose of the vehicle in conditions that make the state unobservable.

Real time performance is a requirement for this work, thus trade-offs between performance and accuracy/robustness are investigated, especially in the area of feature detection and matching. On the other hand, ignoring the performance issues and focusing only on accuracy opens the field for post-processing the results offline in order to attempt high quality reconstruction of the observed environment both in the image and the 3D geometry domain.

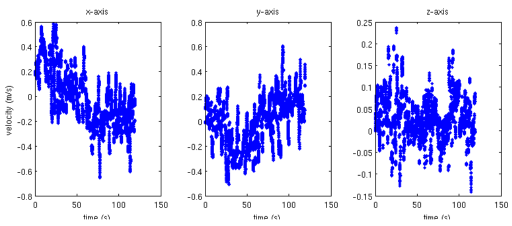
Autonomous operations over coral reefs would benefit greatly from accurate localization algorithms. The ability to revisit the same area over long periods of time would enable the investigation of climate change on the fragile ecosystem of coral reefs, as well the effect of conservation efforts.

## Acknowledgments

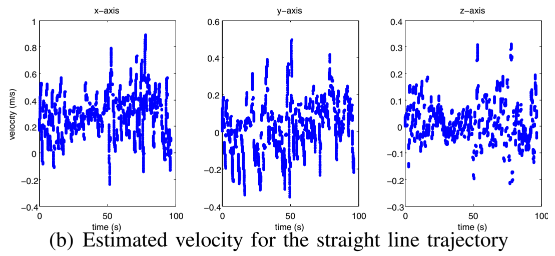
The authors would like to thank Philippe Giguere for assistance with robot navigation; Yogesh Girdhar and Chris Prahacs for their support as divers; Marinos Rekleitis and Rosemary Ferrie for their help with the experiments.

## References

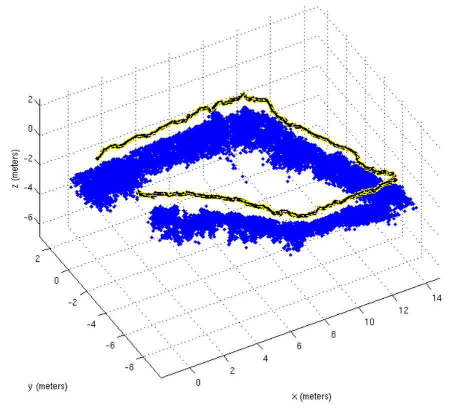
[1] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052 –1067, jun. 2007.
[2] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *CVPR*, 2010, pp. 1498–1505.
[3] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. I652 – I–659.
[4] K. Konolige et. al., "Large scale visual odometry for rough terrain," in *Proc. Int. Symposium on Research in Robotics*, 2007.
[5] P. T. Furgale and T. D. Barfoot, "Stereo mapping and localization for long-range path following on rough terrain," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 4410–4416.
[6] M. Cummins and P. Newman, " Highly Scalable Appearance-Only SLAM FAB-MAP 2.0 ," in *Robotics Science and Systems*, 2009.
[7] J. Sattar, G. Dudek, O. Chiu, I. Rekleitis, P. Giguère, A. Mills, N. Plamondon, C. Prahacs, Y. Girdhar, M. Nahon, and J.-P. Lobos, "Enabling autonomous capabilities in underwater robotics," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008.
[8] B. Woźniak and J. Dera, *Light absorption in sea water*. Springer Verlag, 2007.
[9] S. Skaff, J. J. Clark, and I. Rekleitis, "Estimating surface reflectance spectra for underwater color vision," in *British Machine Vision Conf.*, Leeds, U.K.,, 2008.
[10] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, 2007, pp. 3565–3572.



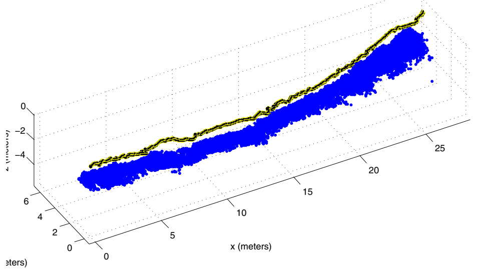
(a) Estimated velocity for the loop trajectory



(b) Estimated velocity for the straight line trajectory



(c) Estimated reconstruction of both the trajectory and the 3D structure of the seafloor.



(d) Estimated reconstruction of the straight line trajectory and the 3D features seen along the way.

Fig. 7. (a), (b) The maximum forward velocity of our robot has been experimentally determined to be close to $1m/s$, so both of these estimates are within the bounds of normal operation. While ground truth values for the velocity estimates are not available (e.g. via a DVL), in the case of the near-straight line the robot traversed 30 meters in 130 seconds, which means that on average the forward velocity of the robot was 0.23m/s. (c), (d) The depth of the estimated features from the robot's camera is correctly estimated to be around 2m.

[11] ——, "A dual-layer estimator architecture for long-term localization," in *Proc. of the Workshop on Visual Localization for Mobile Platforms*, Anchorage, AK, 2008, pp. 1–8.

[12] E. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *Int. J. of Robotics Research*, 2010.

[13] J. Kelly, S. Saripalli, and G. S. Sukhatme, "Combined visual and inertial navigation for an unmanned aerial vehicle," in *Proc. 6th Int. Conf. Field and Service Robotics*, Chamonix, France, 2007.

[14] P. Corke, C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu, "Experiments with underwater robot localization and tracking," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 4556–4561.

[15] J. Salvi, Y. Petillot, S. Thomas, and J. Aulinas, "Visual SLAM for underwater vehicles using video velocity log and natural landmarks," in *OCEANS*, 2008, pp. 1 –6.

[16] N. Gracias, S. van der Zwaan, A. Bernardino, and J. Santos-Victor, "Results on underwater mosaic-based navigation," in *MTS/IEEE OCEANS*, vol. 3, 2002, pp. 1588 – 1594.

[17] S. Negahdaripour and X. Xu, "Mosaic-based positioning and improved motion-estimation methods for automatic navigation of submersible vehicles," *IEEE J. of Oceanic Engineering*, pp. 79 –99, 2002.

[18] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard, "Visually navigating the RMS Titanic with SLAM information filters," in *Proc. of Robotics: Science and Systems*, 2005, pp. 57–64.

[19] I. Mahon and S. Williams, "Slam using natural features in an underwater environment," in *Control, Automation, Robotics and Vision Conf.*, vol. 3, 2004, pp. 2076 – 2081 Vol. 3.

[20] W. G. Breckenridge, "Quaternions proposed standard conventions," JPL, Tech. Rep. INTEROFFICE MEMORANDUM IOM 343-79-1199, 1999.

[21] N. Trawny and S. I. Roumeliotis, "Indirect Kalman Filter for 3D Attitude Estimation," University of Minnesota, Dept. of Comp. Sci. and Eng.,, Tech. Rep. 2005-002, March 2005.

[22] F. Landis Markley, "Attitude estimation or quaternion estimation?" *The J. of the Astronautical Sciences*, vol. 52, pp. 221–238, 2004.

[23] O. A. Bauchau and L. Trainelli, "The vectorial parameterization of rotation," *Nonlinear Dynamics*, vol. 32, pp. 71–92, 2003.

[24] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *Int. J. of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.

[25] F. Mirzaei and S. Roumeliotis, "A Kalman Filter-based Algorithm for IMU-Camera Calibration: Observability Analysis and Performance Evaluation," *IEEE Trans. on Robotics*, vol. 24, pp. 1143–1156, 2008.

[26] F. Shkurti, I. Rekleitis, and G. Dudek, "Feature tracking evaluation for pose estimation in underwater environments," in *Proc. of the 8th Canadian Conf. on Computer and Robot Vision*, 2011, pp. 160–167.

[27] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *ECCV*, 2006, pp. 404–417.

[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, 2004.

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

[30] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center surround extremas for realtime feature detection and matching," in *ECCV*, 2008, pp. 102–115.

[31] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 1994, pp. 593 – 600.

[32] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Int. Conf. on Computer Vision Theory and Application*. INSTICC Press, 2009, pp. 331–340.

[33] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework: Part 1," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-02-16, 2002.