Monocular Odometry for Underwater Vehicles with Online Estimation of the Scale Factor

Vincent Creuze*

**LIRMM – CNRS / University of Montpellier.*
Montpellier, France (e-mail: vincent.creuze@lirmm.fr).

Abstract: This paper introduces a new visual odometry method for underwater vehicles. It is based on images grabbed by a monocular video camera, and aided by inertial and pressure measurements. This approach offers several advantages. Firstly, it is compact and runs very fast, even with limited computational resources. This allows to embed it on very small vehicles. Secondly, and probably most importantly, the method is able to estimate online the scale factor of the observed scene, thanks to the combined measures of a low-cost IMU and a pressure sensor. The paper ends with an experimental validation onboard the *Leonard* underwater vehicle.

Keywords: Marine Robotics, Computer vision, Localization, Odometry.

1. INTRODUCTION

Accurate localization is required when underwater vehicles are operating close to the seabed. For instance, while performing predefined survey tracks (e.g. photogrammetric acquisition of a wide area) or doing remote manipulation (e.g. collecting samples or operating on an industrial structure), the accuracy of acoustic based localization systems is not enough. Indeed, USBL (Ultra Short BaseLine) or LBL (Long BaseLine) acoustic systems are accurate enough to control the vehicle during its descent into the water column, but provide too noisy measures (from 10cm to several meters), at too low frequency (often less than 1Hz), for accurate operations near the seabed. On mid and big sized underwater vehicles, the solution consists in using inertial measurement systems based on FOG (Fiber Optical Gyroscopes), combined with the ground speed measurements of a DVL (Doppler Velocity Log) and the absolute acoustic localization of an USBL or an LBL for instance. These methods are unfortunately heavy (several kg) and very expensive (more than 100k€) and are not suited to small or very small vehicles, due to the limited payload of such vehicles. These are some of the reasons for which various localization and mapping methods based on video have been proposed for several years. Some works exploit monocular vision as Garcia (2001), Gracias (2002), or Mahon (2004), who used also a sonar to preselect best areas for the vision system. Some other papers are based on stereoscopic systems, like Negahdaripour (2005). This author performed numerous real-time experiments (15Hz), demonstrating that stereoscopic methods have smaller accuracy errors (2%) than monocular approaches, especially for low contrasted scenes (e.g. ship-hulls or walls of pools). The author suggests that this is due to the inaccurate pose estimation of monocular methods when too few feature points are available. Drap (2015) used a stereoscopic system to perform real-time (10Hz) stereoscopic localization and pose estimation of a submarine to guide the pilot so as to completely cover a shipwreck. After the dive, the images of a third high resolution camera are combined with the images of the low-resolution stereoscopic system to build an accurate 3D model of the scene.

Some other authors have combined vision with other measurements. This was the case of Eustice (2006), who performed monocular SLAM, robust to low-overlap constraint, on a part of RMS Titanic. Using an inertial unit, his approach improves the reliability of data association. More recently, Shkurti (2011), proposed to integrate the data from both a low-cost IMU and a pressure sensor in an Extended Kalman Filter (EKF). Then, the authors track features (SURF with Approximate Nearest Neighbor matching) in several monocular frames. This gives a 3D position estimate, regarded as true and used to correct the EKF. In particular, the authors reported for experimental accuracy of 1 meter over a 30-meter-long straight trajectory, with a system running at 15Hz. In spite of the heavy computations of feature detection and matching, real-time was achieved but necessitated trade-offs between performances and robustness. Warren (2012) used the data from a magnetometer to constrain the pose estimation of a stereoscopic system installed on the Sirius AUV. This minimizes the angular drift and the authors reported for error smaller than 6.4 meters over a 300-meter-long trajectory. The stereoscopic system ran at 1Hz to save energy.

The previous examples integrate the data of external sensors (IMU, depth sensor) either to improve/constrain the pose estimation process or directly to estimate a state of the vehicle that will be corrected by the visual information. Most of them rely on heavy computations. Some of them may fail during station keeping operations, due to the difficulty to perform bundle adjustment while keeping the robot still.

In this paper, we propose a method for underwater odometry (not for mapping), that is able to online estimate the scale factor of the observed scene. This method is based on a monocular video camera, associated with a low-cost MEMS

IMU (Inertial Measurement Unit) and a pressure sensor (depth sensor). It is fast enough (at least 30Hz) to be included in the low-level control loop of an underwater vehicle and is enough computationally efficient to be implemented on very small computer boards (running at 10Hz on a Raspberry Pi 3 Model B). Moreover, this method does not drift with time, but only with the covered distance. This makes it very well suited for station-keeping, or for accurate control of the motion of a vehicle during underwater manipulations. It can also be combined with an external acoustic positioning system (e.g. USBL) for longer range navigation. Its specificity with respect to previous work is that the inertial data are directly used in the image processing, and the depth measurements allow to compute the altitude of the vehicle in a straightforward manner. Compared with methods based on stereovision, this approach uses frames grabbed by a single monocular camera. This makes it more compact and this requires less computation and energy. Section 2 will present the details of the algorithm. In section 3, we present experimental results obtained in a pool with the *Leonard* underwater vehicle (Fig. 1). These experiments have also been reproduced in real-time during the demonstrator session of IFAC 2017.

2. VISUAL ODOMETRY ALGORITHM

2.1 Technical setup

To perform the visual odometry only three devices are needed (Fig. 2): a downward facing monocular video camera (resolution: 640x480), a depth sensor, and a MEMS IMU (9 dof) running at least at the same frequency as the camera frame rate.

In our case, the IMU is located inside the housing of the camera, but could be placed anywhere else as long as a calibration procedure allows to align the axes of the IMU with the axes of the camera and the axes of the ROV. The positions of the camera frame and the pressure sensor with respect to the body frame of the ROV have to be accurately measured. This will be used later to compensate for the effects induced by pitch, roll and yaw motions.

2.2 Assumptions

For clarity purposes, in the following, we will assume that the camera and the IMU axes are aligned. We will also assume that the camera frame $\mathcal{R}_c$ coincides with the body frame $\mathcal{R}_b$ of the vehicle and that the pressure sensor is located at the center of the body frame of the vehicle. In practical conditions, these assumptions are not realistic, but can easily be compensated by well-known geometric transformations that we will not detail here. We also assume that the calibration of the camera (pinhole model) has been done, so that the intrinsic parameters are known. The distortion coefficients, are also assumed to be known and are used to undistorted any grabbed image. This will not appear in the following, but is of course taken into account in the implemented code. Finally, we also assume that the seabed is quite regular, i.e. without any sudden depth variation.

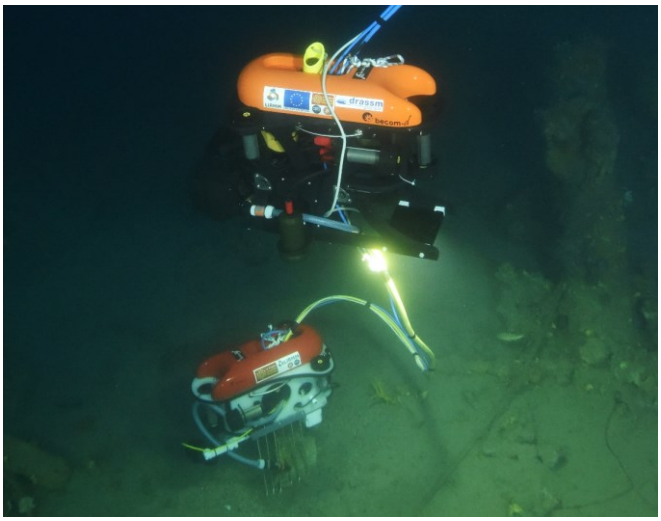

Fig. 1. The *Leonard* ROV (at the top of the picture) during coordinated archaeological operations with its twin brother *Speedy* ROV (at the bottom of the picture), on the *Lune* shipwreck (Depth: 90 meters, Toulon, France). Courtesy of: F. Osada/T. Seguin - DRASSM.

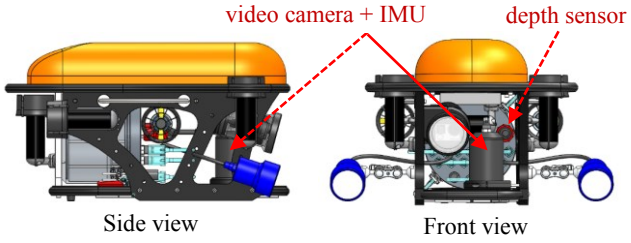

Fig. 2. Example of setup, here mounted on the *Leonard* ROV. The IMU is mounted inside the camera housing. The depth sensor is located on the ROV's main housing.

2.3 Algorithm

The algorithm is based on several steps:

Initialization

At the initialization of the algorithm (at time t_0), an image I_0 is grabbed by the camera and the depth p_0 is recorded, as well as the attitude (yaw, pitch, roll) of the camera provided by the IMU (i.e. the attitude of the ROV) and denoted $\Theta_0 = (\varphi_0, \theta_0, \psi_0)$.

Then one selects an initial set S_0 of n_0 strong feature points. The selection method is not detailed in this paper but could done for instance by selecting the most relevant Harris points in the image, as described by Shi and Tomasi (1994).

Each point of the S_0 set is denoted $m_{i,0} = (u_{i,0}, v_{i,0})$, with $i \in [1, n]$ and $(u_{i,0}, v_{i,0})$ being its coordinates in the image I_0, i.e. at time $t = t_0$.

Iterations

The iterative process starts by grabbing a new image I_k at time $t = t_k$. As for the initialization, the attitude $\Theta_k = (\varphi_k, \theta_k, \psi_k)$ and the depth p_k are also recorded.

Then, one calculates the optical flow for every points m_i of S_0 using the iterative Lucas-Kanade method with pyramids, as proposed by Bouguet (2000). This allows to find the new positions $m_{i,k} = (u_{i,k}, v_{i,k})$ of the points of S_0. During this step, some points cannot be tracked and are lost (e.g. when optical flow fails, or when points disappear or exit from the image). In this case, we eliminate them from the set S_0. The rest of the points form the S_k set.

We now apply a yaw, pitch and roll compensation to the points of S_k. This gives the attitude-compensated points $m_{\varphi\theta\psi_{i,k}} = (u_{\varphi\theta\psi_{i,k}}, v_{\varphi\theta\psi_{i,k}})$, located where the feature points should be (in the image) if the pitch and roll angles had not changed with respect to their initial values (at time t_0). For this, we use the attitudes $\Theta_0 = (\varphi_0, \theta_0, \psi_0)$ and $\Theta_k = (\varphi_k, \theta_k, \psi_k)$ measured by the IMU respectively at time $t = t_0$ and $t = t_k$.

Once the variation of the attitude has been compensated, the mean zooming ratio $< \rho_k >$ is computed. For this, for every duets of the n_k points remaining in the S_k set, we firstly compute $(d_{ij})_k$ the distance between points $m_{\varphi\theta\psi_{i,k}}$ and $m_{\varphi\theta\psi_{j,k}}$ at time $t = t_k$ and $(d_{ij})_0$ at time $t = t_0$, with $i < j$ and $i, j \in S_k$.

Then, the mean zooming ratio $< \rho_k >$ is defined as:

$$< \rho_k >= \frac{2}{n_k(n_k - 1)} \sum_{i<j \text{ and } i,j \in S_k} \frac{(d_{ij})_k}{(d_{ij})_0} \quad (1)$$

It has to be mentioned that the observed zooming effect is only due to the depth variation of the robot between t_0 and t_k. Indeed, ρ_k is not affected by the seabed depth variations as the features points used correspond to fixed point of the seabed that are tracked from t_0 to t_k.

From the mean zooming ratio $< \rho_k >$, and the depth p_k measured by the pressure sensor, one can compute as follows the altitude a_k of the vehicle, i.e. its distance to the seabed. Let us consider the 2D example of Fig.3. For better readability, the yaw-pitch-roll compensated feature points $m_{\varphi\theta\psi_{i,k}}$ are simply denoted $m_{i,k}$ on the figure. The seabed point corresponding to $m_{i,0}$ and $m_{i,k}$ image points is denoted M_i.

The focal length of the camera is denoted f and the center of projection (optical center) coincides with the center of the ROV's body frame $\mathcal{R}_b$. The position of the ROV is denoted η_k and u_k is the image coordinate of point $m_{i,k}$. The distance between the principal axis and the seabed point M_i is denoted d.

For sake of clarity, in this example, we consider the distance d between a point M_i and the principal axis, but the same results could be obtained between two points M_i and M_j by application of the intercept theorem.

From the pinhole model, we have:

$$u_{i,k}.\, a_k = u_{i,0}.\, a_0 = f.\, d \quad (2)$$

Introducing $\rho_{i,k} = \frac{u_{i,k}}{u_{i,0}}$ the individual zooming factor, we can rewrite (2) as:

$$\rho_{i,k}.\, a_k = a_0 \quad (3)$$

As the relation between altitude a_k (in meters) and depth p_k (also in meters) is:

$$a_0 = a_k + (p_k - p_0) \quad (4)$$

we can combine (3) and (4), and get:

$$a_k = \frac{p_k - p_0}{\rho_{i,k} - 1} \quad (5)$$

Once applied to the entire set S_k of feature point, this gives:

$$a_k = \frac{p_k - p_0}{< \rho_k > - 1} \quad (6)$$

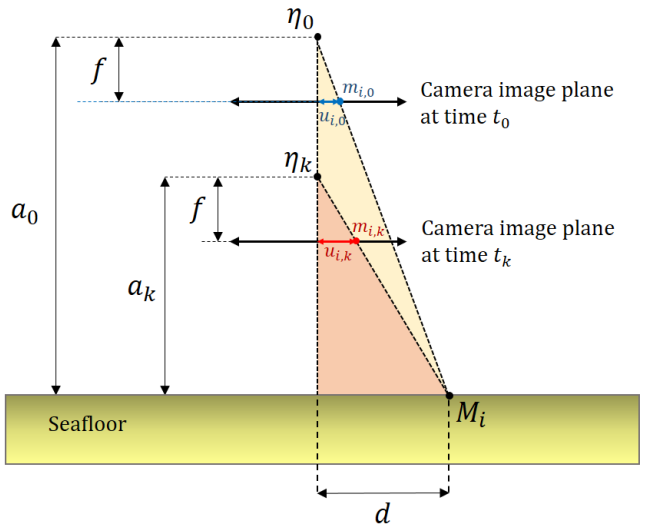

Fig. 3. Positions of a seabed point M_i in the camera image plane, when the ROV is at altitude a_0 and altitude a_k.

The last step of the algorithm is the computation of the current position of the vehicle (at t_k), with respect to its initial position (at t_0). For this purpose, we firstly compute $m_{\varphi\theta\psi\boldsymbol{\rho}_{i,k}} = (u_{\varphi\theta\psi\boldsymbol{\rho}_{i,k}}, v_{\varphi\theta\psi\boldsymbol{\rho}_{i,k}})$, which are the n_k positions of the feature points, compensated for attitude variation, but also for zooming effect (i.e. variation of the altitude). Once this is done, the Δx and Δy variation of the ROV's position is given by:

$$\begin{cases} \Delta x = -\frac{a_k}{f_x.\, n_k} \sum_{i \in [1, n_k]} (u_{\varphi\theta\psi\rho_{i,k}} - u_{i,0}) \\ \Delta y = \frac{a_k}{f_y.\, n_k} \sum_{i \in [1, n_k]} (v_{\varphi\theta\psi\rho_{i,k}} - v_{i,0}) \end{cases}$$

where f_x and f_y are the focal lengths of the camera, expressed in pixels.

The iterative process is performed while the number n_k of points in the S_k is large enough to smooth the disturbance induced by the irregularities of the seabed (in fact a_k is a mean altitude, as the seabed is not necessarily flat). When too many points have been lost (for instance when the vehicle has moved away from its original position), n_k goes under a certain threshold n_{min}. Then, the iterative process is stopped and the algorithm returns to the initialization step, and renew its set of points. The new "initial position" is set to the last "current position".

It has to be noticed also that in case of non-flat bottoms, after every reset of the process (initialization step) the depth trajectory has to vary in order to estimate again the altitude of the ROV. Navigating at constant depth does not allow to estimate the altitude. However, when the vehicle navigates close to the seabed (2m or less), a variation of a few centimeters in the depth trajectory (as almost every commercial ROVs naturally do) is sufficient.

3. EXPERIMENTS AND DISCUSSION

3.1 Experimental setup

Experiments have been conducted with the *Leonard* ROV (Fig. 1). This vehicle has been designed at LIRMM in 2015. Its features are summarized in Table 1.

Table 1. *Leonard* ROV's technical features

Size	70 x 50 x 50 cm
weight	28 kg
max operating depth	100 meters
IMU	Sparkfun ArduIMU V2
depth sensor	Range: 0-150 m Relative accuracy: 2 cm
Camera	IM-E630 & 640x480 Grabber
Thrusters	6 Seabotix BT150

During the experiments, the visual odometry algorithm and the control of the ROV were computed from the surface (through the tether) by a laptop PC (Intel® Core™ i7 5600U 2.60GHz). The codes are written in C++. For some of the functions needed in the presented algorithm, we used OpenCV library (Bradski2000). The frequency of the algorithm was limited by the fps of the camera, so it worked at 25fps (PAL analog camera). Under these conditions, the CPU load was around 8%.

3.2 Altitude estimation

To compare the performances of the method with a reliable ground truth, we performed tests in a pool. The bottom of the pool was flat with sparse 5-cm-high ripples (a removable plastic carpet protects the bottom from impacts). These tests have been reproduced during demonstrator session of IFAC 2017.

The first test consisted in the validation of the altitude estimation. The ROV remained stable in the horizontal plane and performed a vertical descending trajectory, as depicted on Fig. 4. As we knew precisely the depth of the pool (1.05 meter), the ground truth for altitude was computed from the pressure measures (depth sensor). During this test, the algorithm has been initialized only once (120 feature points) as the set of feature points remained larger than $n_{min} = 30$. On this picture, one sees the measured depth trajectory of the ROV in blue, the true altitude of the ROV in black and the altitude estimated by the algorithm in red. This latter tracks well the true one. However, as the position of the optical center was not accurately known, one observes a 3cm offset. The noise level in the altitude estimation is about 2cm during this test, but one can predict that it depends on the altitude range.

One observes also that the altitude is not estimated during the first seconds. Indeed, the estimation is not performed until $< \rho_k > -1$ is large enough to avoid dividing by a too small number in (6). This figure also shows that the method does not drift when time grows, which is normal as errors may accumulate only during successive resets (initializations) of the algorithm (i.e. error is dependent on the horizontal distance covered by the ROV).

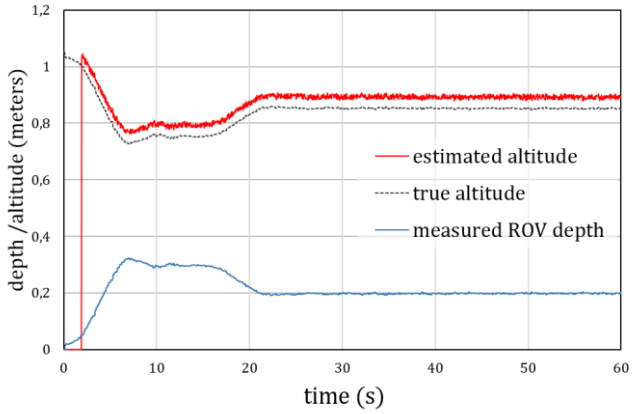

Fig. 4. Time history of the depth trajectory of the ROV (in blue) and the estimated altitude (in red) during combined horizontal station keeping and vertical descent.

3.3 Horizontal odometry

During the second test, the ROV performed a vertical yoyo (30cm) to estimate its altitude and the returned to the surface. After this first motion, it has been manually hauled along a rail, so as to draw a 2-meter-long straight trajectory. Figure 5 shows the experimental result. The beginning of the trajectory is noisy as it corresponds to the yoyo phase. The second part of the trajectory is straight. The estimated length is 2.07m, while one observes offsets smaller than 3cm along the ideal straight trajectory. An accurate evaluation of the performance is however difficult as the ROV was moved manually and the undesired induced roll and pitch disturbances (yaw disturbance was mechanically impossible) could have generate these offsets in the camera's position.

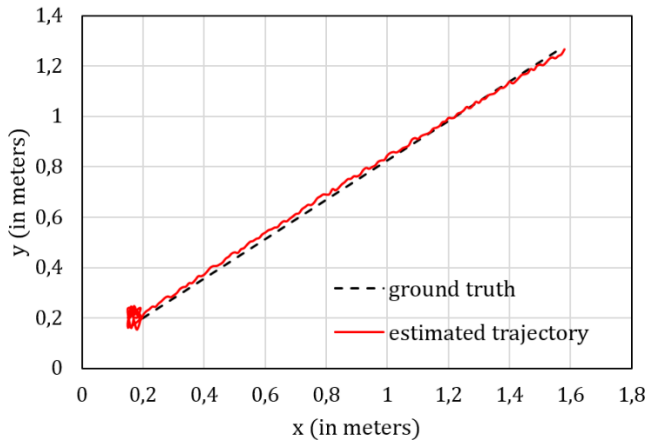

Fig. 5. Estimated horizontal trajectory. The ROV firstly does a yoyo to estimate the altitude and then is manually hauled along a 2-meter-long straight rail.

3.4 Tests at sea

The ROV has been operated several times at sea, where we could only obtain a qualitative appreciation of the proposed method as no ground truth was available. These experiments demonstrated that natural seafloor always offer far enough features to track. We also observed that the algorithm is robust towards erratic motions of a minor part of the feature points. For instance, we have experienced seabed with moving shrimps or fishes, without observing any major disturbance of the estimated position. To improve the robustness, we have added the computation of the standard deviation of the $\frac{(d_{ij})_k}{(d_{ij})_0}$ ratio. Above a certain threshold, we reset the points (initialization step).

In presence of major disturbances, such as large amount of suspended particles moving with the sea current, the algorithm fails.

6. CONCLUSION

In this paper, we have introduced a new method for underwater monocular odometry. This method combines vision with IMU and depth measurements. This allows to evaluate the altitude of a vehicle and its position. The position error of this algorithm does not grow with time, but only with the covered distance, which made this method ideal for horizontal servoing during manipulation or for station-keeping. It is also a good complement of absolute acoustic positioning systems (USBL, LBL) for operations close to the seabed. Experiments in pool and at sea are commented.

REFERENCES

Bouguet Jean-Yves (2000). Pyramidal Implementation of the Lucas Kanade Feature Tracker. Intel Corporation, Microprocessor Research Labs.

Bradski, G. (2000). Opencv_library, Dr. Dobb's Journal of Software Tools.

Drap Pierre, et al (2015). Underwater Photogrammetry and Object Modeling: A Case Study of Xlendi Wreck in Malta. Sensors. 15(12), pages 30351-30384.

Eustice RM, Singh H, Leonard JJ, Walter MR (2006). Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information filters. The International Journal of Robotics Research. 25 (12), pages 1223-1242.

Garcia Rafael, Cufi Xavier, and Carreras Marc (2001). Estimating the motion of an underwater robot from a monocular image sequence. IEEE/RSJ IROS 2001 - International Conference on Intelligent Robots and Systems, pages 1682–1687.

Gracias Nuno and Santos-Victor Jose (2000). Underwater video mosaics as visual navigation maps. Elsevier Journal of Computer Vision and Image Understanding - Special issue on underwater computer vision and pattern recognition archive, Volume 79, Issue 1, Pages 66-91.

Mahon I. and Williams S. (2004). Slam using natural features in an underwater environment. IEEE ICARCV 2004 Control, Automation, Robotics and Vision Conference. pages 2076–2081.

Negahdaripour Shahriar (2006). An ROV Stereovision System for Ship-Hull Inspection. IEEE Journal of Oceanic Engineering, 31(3), pages 551-564.

Shi J. and Tomasi C, (1994). Good Features to Track. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600.

Warren M., Corke P, Pizzaro O., Williams S., and Upcroft B., (2012) Visual sea-floor mapping from low overlap imagery using bi-objective bundle adjustment and constrained motion. ACRA 2012 - Australasian Conference on Robotics and Automation.

ACKNOLEDGMENTS

The author greatly acknowledges support of the European Union through FEDER grant n° 49793 and support of Région Languedoc-Roussillon-Occitanie for ARPE Seahand grant.