Building Underwater Mosaics Using Navigation Data and Feature Extraction[*]

M. Laranjeira
Polytechnic School, Federal University of Bahia, Brazil
matheuslaranjeira1@gmail.com

L. Jaulin
ENSTA Bretagne, LABSTICC, Brest, France
luc.jaulin@ensta-bretagne.org

S. Tauvry
ECA Robotics, Brest, France
tauvry.s@ecagroup.com

Abstract

This paper proposes an original approach, based on navigation data analysis, for underwater mosaic creation from videos recorded by an AUV (Autonomous Underwater Vehicle). Two methods are presented: one uses only the AUV navigation data for image merging and the other also uses some techniques of feature extraction and image matching. The process of image matching is described and some improvements are proposed. A lighting correction procedure is also implemented in both cases to improve blending quality. Besides the mosaicking methods, a technique for trajectory refinement is proposed to handle the inherent uncertainties of navigation data and enhance the mosaic reliability. This technique is based on the analysis of overlapping images and it uses the interval algebra to compare proprioceptive and exteroceptive data. The mosaicking methods were tested in two missions carried out in the Mediterranean Sea. All the mosaics created, some additional images and videos are available as multimedia supporting materials.

Keywords: underwater robotics, 2D mosaics, image matching, interval analysis, SLAM

AMS subject classifications: 65-00

[*]Submitted: February 29, 2016; Revised: August 17, 2016; Accepted: August 18, 2016.

1 Introduction

Automatic video mosaicking of the ocean floor has many applications, such as: seabed reconstitution, wreckage visualization, site exploration and visual navigation [14]. In recent years, several techniques of underwater mosaic creation have been developed, and most of them are based on image processing [19],[13].

In this paper we propose a different approach, which relies on the AUV (Autonomous Underwater Vehicle) navigation data for mosaic building. The camera pose is estimated by analysis of navigation data, and all the images taken during the mission are warped to orient them towards the north and correct projective distortion, scaling and rotation. Based on that, two methods of mosaicking are developed: one uses only the navigation data for image merging, and another uses feature extraction techniques for image matching. Both methods also use a lighting correction algorithm to improve image blending.

Previous works using only navigation data for underwater mosaicking were done by Haywood [15] thirty years ago; however, the inaccuracies in position measurements were too large compared to the size of a video pixel [19]. In reference [21], a new method of creating continuous geo-referenced image tiles used for assessment of underwater faunal density and diversity is proposed. The authors use only navigation data for image blending; however, they make 1D mosaics. In this paper, we present a 2D mosaic creation method. To have better results, we also propose a method of trajectory refinement, using interval analysis to handle the uncertainties inherent in the navigation data.

The feature matching technique used in this paper has a slight improvement with regard to references [23] and [18], presenting a new procedure for outlier rejection, in addition to RANSAC (Random Sample Consensus). Also, the mosaicking method using feature extraction is quite different from the classical approach ([19] and [13]) based on image-correlation matching, since we use the *k-nearest neighbor* algorithm to find corresponding features.

The mosaicking methods were tested on two missions carried out in the Mediterranean Sea by the AUV A9 of ECA Robotics. All the mosaics created, some additional images, videos and other supporting materials (125.6Mb in size) are available at `https://sites.google.com/site/matheuslaranjeira1/projects/seabed-mosaics/`.

The paper is organized as follows. Section 2 contains the interval algebra and the data used to estimate the AUV position. In Section 3 the loop detection problem is presented and the notion of t-plane is introduced. This notion will be used in Section 6 in refining the robot trajectory. Section 4 and 5 contain the image processing and matching techniques used by the methods of mosaic creation, presented in Section 7. Two real test-cases are described in Section 8, and some additional comparisons to other work are made in Section 9. Section 10 concludes this paper.

2 AUV Positioning

The algorithms for mosaic creation and trajectory correction use AUV speed, attitude and seabed altitude measurements issued from the INU (Inertial Navigation Unit) as inputs. The INU consists of an IMU (Inertial Measurement Unit), a DVL (Doppler Velocity Log) sensor, a GPS and a pressure sensor. The GPS data were not used for this paper.

The speed vector is defined as $\mathbf{v}(t) = (v_x(t), v_y(t), v_z(t))$, the attitude corresponds to the three Euler angles (roll, pitch and heading) here defined as $\varphi(t), \theta(t)$ and $\psi(t)$, respectively, and the AUV seabed altitude is represented by $h(t)$. These measurements are obtained in the *robot frame*, that is defined by a right-handed orthogonal set $(\vec{X_R}, \vec{Y_R}, \vec{Z_R})$ with origin at the center of the robot. $\vec{X_R}$ points forward, $\vec{Y_R}$ points port-side, and $\vec{Z_R}$ points upward.

The *world frame* is defined by a set $(\vec{X_W}, \vec{Y_W}, \vec{Z_W})$ of orthogonal coordinates, where $\vec{X_W}$ points northward, $\vec{Y_W}$ points westward, and $\vec{Z_W}$ points upward. The origin of the *world frame* coincides with the initial position of the robot in the mission.

The relation between the *world* and *robot* frames is given by the Euler rotating matrix $\mathbf{R}(\varphi(t), \theta(t), \psi(t))$ and the robot translation. Therefore, the rate of displacement $\dot{\mathbf{p}}_W(t)$ of the robot in the *world frame* is calculated by the following state equation [3]:

$$\dot{\mathbf{p}}_W(t) = \mathbf{R}(\varphi(t), \theta(t), \psi(t)).\mathbf{v}_R(t), \tag{1}$$

where $\mathbf{v}_R(t)$ is the speed vector measured in the robot frame. With a simple integral, all the positions $\mathbf{p}_W(t)$ are calculated by

$$\mathbf{p}_W(t) = \int_0^t \dot{\mathbf{p}}_W(\tau)\, d\tau, \tag{2}$$

where the time t belongs to the interval $[0, t_{\max}]$ associated to the duration of the mission.

2.1 Intervals and tubes

Due to the INU measurement uncertainties, we will associate the measurements $x(t_k)$ with an interval $[x](t_k) = [x(t_k) - \delta_x, x(t_k) + \delta x]$, where δ_x represents the uncertainty of measurement provided by the INU manufacturer. To handle the interval data, we will use interval analysis ([20], [2]), a set of numerical methods which allows nonlinear problems, such as localization [8], SLAM (Simultaneous Locating and Mapping) [9] or state estimation [1] to be solved. In our case, interval analysis and the notion of *tube* will be useful to loop detection [2] and trajectory refinement, explained in sections 3 and 6.

A tube $[x](t)$ (see, e.g., [4]) is an interval of functions $[x^-(t), x^+(t)]$ that contains, for all $t \in [0, t_{\max}]$, the intervals of measurements $[x](t)$. The notion of tube is illustrated in Figure 1, extracted from [2]. Note that for all t, $x^-(t) \leq x^+(t)$.

Using interval algebra, we can rewrite equations 1 and 2 as

$$[\dot{\mathbf{p}}_W](t) = \mathbf{R}([\varphi](t), [\theta](t), [\psi](t)).[\mathbf{v}_R](t), \text{ and} \tag{3}$$

$$[\mathbf{p}_W](t) = \int_0^t [\dot{\mathbf{p}}_W](\tau)\, d\tau, \tag{4}$$

where $\int_{t_1}^{t_2} [\mathbf{x}](\tau)\, d\tau = \left[\int_{t_1}^{t_2} \mathbf{x}^-(\tau)\, d\tau, \int_{t_1}^{t_2} \mathbf{x}^+(\tau)\, d\tau\right]$. More details about integrals of tubes are available in [2].

3 Overlapping Images

In this section, we explain how we can detect overlapping images by analyzing only navigation data. The approach presented is based on the paper of C. Aubry about proprioceptive loop detection [2].

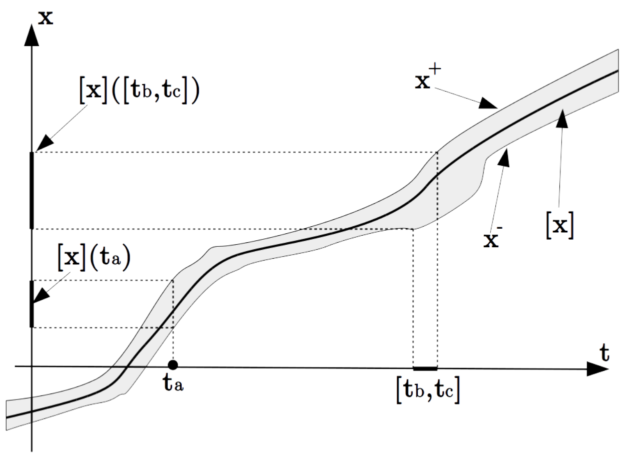


Figure 1: A tube $[x]$ of $\mathbb{R}$ that encloses the function x, where x^+ and x^- are the upper and lower bound of the tube, respectively. $[x](t_a)$ is the interval associated to the AUV position at time t_a

For any trajectory, a set of loops may be defined as

$$\mathbb{T}^*_{lp} = \left\{(t_1, t_2) \in [0, t_{\max}]^2 \mid \mathbf{p}(t_1) = \mathbf{p}(t_2), \, t_1 < t_2\right\},$$

or

$$\mathbb{T}^*_{lp} = \left\{(t_1, t_2) \in [0, t_{\max}]^2 \mid \int_{t_1}^{t_2} \mathbf{v}(\tau) d\tau = \mathbf{0}, \, t_1 < t_2\right\}.$$

However, if we consider our trajectory to be represented by a *tube*, we can rewrite the previous equation as

$$\mathbb{T}_{lp} = \{\mathbf{t} \mid 0 \leq t_1 < t_2 \leq t_{\max}, \, \exists \mathbf{v} \in [\mathbf{v}], \int_{t_1}^{t_2} \mathbf{v}(\tau) d\tau = \mathbf{0}\}, \tag{5}$$

where $\mathbf{t} = (t_1, t_2)$ is called a t-pair and the set $\mathbb{T}_{lp}$ encloses $\mathbb{T}^*_{lp}$.

Nevertheless, in our case, we are looking for the instants where the images taken by the robot overlap. Hence, Equation 5 will be redefined as

$$\mathbb{T} = \{\mathbf{t} \mid 0 \leq t_1 < t_2 \leq t_{\max}, \, \exists \mathbf{v} \in [\mathbf{v}], \int_{t_1}^{t_2} \mathbf{v}(\tau) d\tau \leq dist\}, \tag{6}$$

where *dist* is a distance in meters that ensures the overlapping of images taken at times t_1 and t_2. For seabed inspection, the value of *dist* is normally constant, since the AUV mission is performed at a quasi-constant seabed altitude.

It is important to say that we do not assume, for mosaic building purposes, that the missions are carried out at constant altitude. This assumption is only made in this section to simplify the detection of overlapping. Therefore, this in no way affects the quality of the mosaic. As explained in Section 4.2, the robot altitude (dh) is variable in time. The authors of reference [21] also assume a stable operating altitude in their AUV missions, which have had 3.2 meters of mean altitude and standard deviation (SD) of 0.26 meters. In our missions we have had mean of 6.14 meters and

SD = 0.26, which shows the reasonableness of a quasi-constant altitude hypothesis for image overlapping detection purposes. Still, in reference [2], the authors also propose a simple and efficient way of representing a set $\mathbb{T}$ by plotting its elements in a plane, called the *t-plane*, as shown in Figure 2. In this paper, the t-plane will be useful to graphically represent the t-pairs where images overlap.

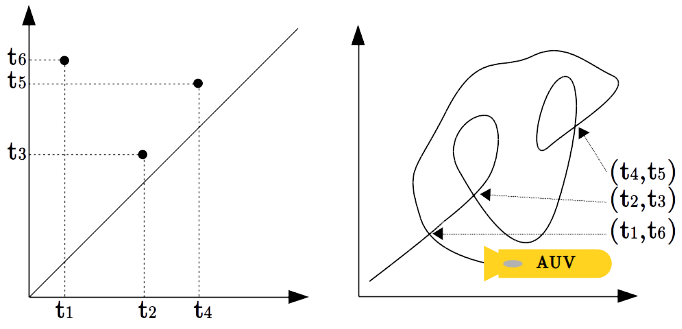


Figure 2: The loops of a trajectory plotted in a t-plane

With the definition of Equation 6, we can use a set inversion test to classify the t-pairs of a t-plane as belonging or not belonging to $\mathbb{T}$. Consider a t-box $[\mathbf{t}] = [t_1] \times [t_2]$ of a t-plane. If $[\mathbf{t}] \subset \mathbb{T}$, all the t-pairs of $[\mathbf{t}]$ are times where their corresponding images overlap. Hence, this box is said to be *feasible*. If $[\mathbf{t}] \cap \mathbb{T} = \emptyset$, then this box is said to be *unfeasible*; otherwise it is *undetermined*. The test used to classify boxes $[\mathbf{t}]$ of a t-plane is presented in the following proposition. **Proposition 1**. Given a t-box $[\mathbf{t}]$, let

$$[Ix] = \int_{[t_1]}^{[t_2]} [v_x](\tau)d\tau, \ [Iy] = \int_{[t_1]}^{[t_2]} [v_y](\tau)d\tau,$$

$$[dist] = \sqrt{[Ix]^2 + [Iy]^2} \text{ and } [margin] = [0, dist_{max}],$$

where $[dist] = [dist^-, dist^+]$ is the distance between the positions of the robot at times $[t_1]$ and $[t_2]$ and $dist_{max}$ is the maximum distance for image overlapping. We then have the test

$$\left. \begin{array}{c} [t_1] - [t_2] \subset \mathbb{R}^- \text{ and} \\ [dist] \subset [margin] \end{array} \right\} \Rightarrow [\mathbf{t}] \subset \mathbb{T},$$

and

$$\left. \begin{array}{c} [t_1] - [t_2] \subset \mathbb{R}^+ \text{ or} \\ dist^- > dist_{max} \end{array} \right\} \Rightarrow [\mathbf{t}] \cap \mathbb{T} = \emptyset.$$

The proposed procedure used to characterize the t-plane is a branch and bound algorithm similar to SIVIA (Set Inverter Via Interval Analysis) [17], [2]. In this algorithm, the t-plane will be partitioned into boxes that will be classified according to the test described previously. The inputs of the algorithm are the tubes $[v_x]$ and $[v_y]$, the accuracy ε, the distance $dist_{max}$ and the duration of the mission t_{max}. The algorithm returns three subpavings: $\mathbb{T}^{\text{in}}$ containing the boxes $[\mathbf{t}]$ that have been proved inside $\mathbb{T}$; $\mathbb{T}^{\text{out}}$ containing the boxes $[\mathbf{t}]$ which do not belong to $\mathbb{T}$; and $\mathbb{T}^{?}$ containing small boxes $[\mathbf{t}]$ for which nothing is known. This procedure is described in detail in Algorithm 1.

The subpaving $\mathbb{T}^{\text{in}}$ contains the set of t-pairs for which it is certain that there are overlapping images. It represents the main outcome of the algorithm, and it

Algorithm 1 Overlapping(in: $\varepsilon, t_{\max}, dist_{max}, [v_x], [v_y]$)

```
 1: $\mathcal{Q} := \{[0, t_{\max}] \times [0, t_{\max}]\}, \mathbb{T}^{\text{in}} = \emptyset, \mathbb{T}^{?} := \emptyset, \mathbb{T}^{\text{out}} := \emptyset$
 2: **if** $\mathcal{Q} \neq \emptyset$ **then**
 3:     take an element $[\mathbf{t}]$ in $\mathcal{Q}$
 4: **else**
 5:     **return** $\mathbb{T}^{\text{in}}, \mathbb{T}^{?}, \mathbb{T}^{\text{out}}$
 6: $\left\{\begin{array}{l} [Ix] := \int_{[t_1]}^{[t_2]} [v_x](\tau)d\tau;\ [Iy] := \int_{[t_1]}^{[t_2]} [v_y](\tau)d\tau \\ [dist] := \sqrt{[Ix]^2 + [Iy]^2};\ [margin] := [0, dist_{max}] \end{array}\right.$
 7: **if** $[t_1] - [t_2] \subset \mathbb{R}^+$ or $dist^- > dist_{max}$ **then**
 8:     $\mathbb{T}^{\text{out}} := \mathbb{T}^{\text{out}} \cup [\mathbf{t}]$; go to 2
 9: **if** $[t_1] - [t_2] \subset \mathbb{R}^-$ and $[dist] \subset [margin]$ **then**
10:     $\mathbb{T}^{\text{in}} := \mathbb{T}^{\text{in}} \cup [\mathbf{t}]$; go to 2
11: **if** width($[\mathbf{t}]$) $< \varepsilon$ **then**
12:     $\mathbb{T}^{?} := \mathbb{T}^{?} \cup [\mathbf{t}]$ ; go to 2
13: Bisect $[\mathbf{t}]$ and store the resulting boxes in $\mathcal{Q}$; go to 2
```

will be useful in Section 6, for the trajectory refinement. A selected t-pair and its corresponding overlapping images are shown in Figure 3.

4 Image Processing

4.1 Image distortions

During underwater missions, the camera is not always parallel to the seabed. The AUV has some movements in roll and pitch that may be relevant, and the pictures taken may have some perspective distortions that should be corrected, besides rotation and scaling. We propose a method of compensation based on the analysis of the camera's field of view. A similar approach was successfully used in reference [21].

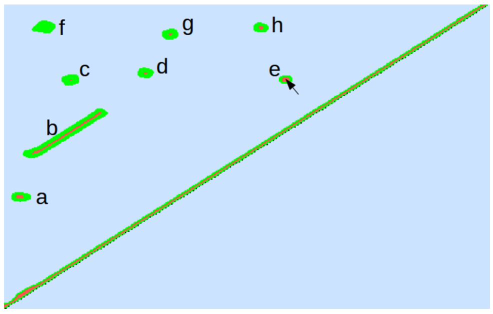
Each picture taken during the mission is associated with a quadrilateral that circumscribes the region imaged, defined by the intersection between the camera field of view and the seabed in the image-taking instant. The coordinates of each corner of the quadrilateral (A,B,C,D) in both robot and world frames can be calculated, since the Euler matrix, the robot altitude and position are known (see Figure 4).

4.2 The coordinates of image corners

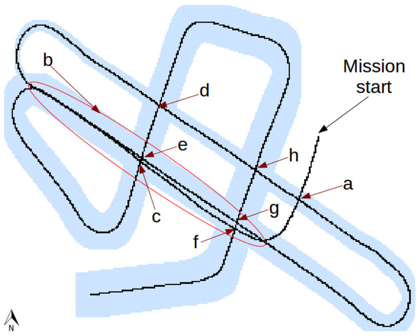
To calculate the coordinates of the aforementioned quadrilateral, we define some quantities: δh is the robot seabed altitude in the instant t_k when the image was taken; the horizontal and vertical aperture of the camera are represented, respectively, by α_H and α_V. We also assume that the camera is at the center of the AUV and that the marine relief is almost flat.

The line connecting the camera center to point P (A, B, C or D) has the following equation (in the robot frame):

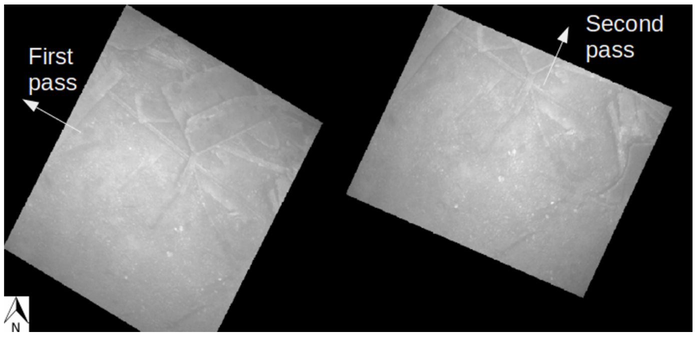
$$x_R = \pm z_R. \tan(\alpha_V/2) \text{ and } y_R = \pm z_R. \tan(\alpha_H/2), \tag{7}$$



(a)



(b)



(c)

Figure 3: (a) Selection of a t-pair (in zone e) in a t-plane: $\mathbb{T}^{\text{in}}$ in red, $\mathbb{T}^{\text{out}}$ in blue and $\mathbb{T}^{?}$ in green. (b) The corresponding trajectory: the center of the path is represented by the black line and the uncertainty of the position by the blue band. (c) The overlapping images of the selected t-pair

where $\pm$ depends on the location of P. We also know that the intersection between this line and the seabed occurs at $z_F = -\delta h$ in a frame F, that is the world frame translated to the robot center. So, from the Euler rotation matrix, we have

$$\begin{aligned} z_F &= \\ &- x_R \sin(\theta) + y_R \cos(\theta)\sin(\varphi) + z_R \cos(\theta)\cos(\varphi) = -\delta h. \end{aligned} \tag{8}$$

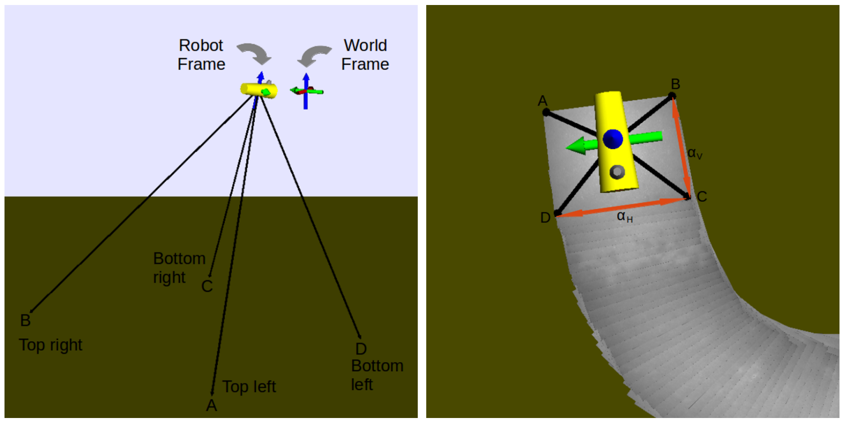
If we replace, in Equation 8, the values of x_R and y_R defined in Equation 7, we can define z_R as

$$z_R := \frac{-\delta h}{\cos(\theta)\cos(\varphi) - \tan(\alpha_H/2)\cos(\theta)\sin(\varphi) + \tan(\alpha_V/2)\sin(\theta)}.$$

Since z_R is known, x_R and y_R are also known.

Now that we have found the coordinates of the corner in the robot frame, we can change to the world reference through the equation

$$\mathbf{p}_W^A(t_k) = \mathbf{R}(\varphi(t_k), \theta(t_k), \psi(t_k)).\mathbf{p}_R^A(t_k) + \mathbf{p}_W^{AUV}(t_k), \tag{9}$$



(a) (b)

Figure 4: 3D simulation of seabed reconstitution from video images. (a) Visualization of the camera's field of view and its intersection with the seabed. (b) An example of mosaic in a 3D simulation

where $\mathbf{p}_W^{AUV}(t_k)$ and $\mathbf{p}_W^{A}(t_k)$ are, respectively, the coordinates of the AUV position and the image corner at time t_k in the world frame. The coordinates of the other corners are calculated in a similar way.

The goal of calculating these coordinates is to correct the perspective effect in the images and also orient them towards the north. Before explaining the derivation, we need to change the units of these coordinates, from meters to pixels. This procedure is explained in the next subsection.

4.3 Changing the coordinates from meters to pixels

To convert the coordinates from meters to pixels, the first step is to define a constant $k_{px} = \frac{\text{Pixels}}{\text{Meters}}$ valid for all images. This ratio represents the correspondence between the displacement in meters of a point on the seabed and its displacement in the image in pixels. It means that each point of mosaic has coordinates in pixels and meters. The strategy of fixed seabed resolution was also used in reference [21]; however, the authors assign navigation coordinates to the image center there.

Suppose that $\mathbb{P}^{tk} = \{\mathbf{p}_W^A(t_k), \mathbf{p}_W^B(t_k), \mathbf{p}_W^C(t_k), \mathbf{p}_W^D(t_k)\}$ is a set containing the coordinates in meters of the four corners (A,B,C,D) of an image taken at time t_k. So, the four corners $P0r, P1r, P2r$ and $P3r$ have their coordinates in pixels calculated as follows:

$$Pir(x,y) = k_{px} \cdot \begin{bmatrix} -\mathbb{P}_y(i) + \max(\mathbb{P}_y) \\ -\mathbb{P}_x(i) + \max(\mathbb{P}_x) \end{bmatrix}, \qquad (10)$$

where $i = 0, 1, 2, 3$ is the index of the corners in the set $\mathbb{P}$, and $\mathbb{P}_x$ and $\mathbb{P}_y$ are the x and y coordinates of corners in $\mathbb{P}$. The relation between pixel coordinates and meters coordinates is explicit in Figure 5: $\vec{X}_{img} = -\vec{Y}_R$ and $\vec{Y}_{img} = -\vec{X}_R$.

4.4 Image warping

The linear transformation $A.Ps + B = Pr$ that deforms the original picture to compensate for its distortions is represented in Figure 5. The coordinates (in pixels) Ps and Pr are, respectively, the coordinates of the original and compensated image. Considering $Pi = \begin{bmatrix} Pi_x \\ Pi_y \end{bmatrix}$, the terms of the matrices

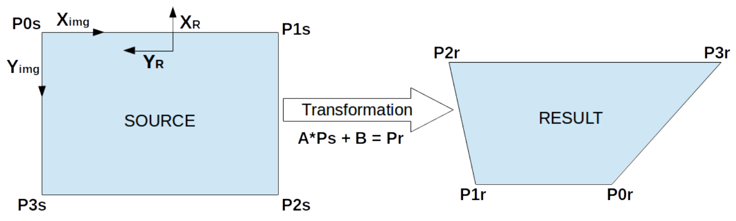


Figure 5: Schematic of perspective effect compensation from the application of a linear transformation

$$A = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

are calculated as follows:

$$a_{00} = \frac{P1r_x - P0r_x}{P1s_x};\ a_{01} = \frac{P3r_x - P0r_x}{P3s_y};$$

$$a_{10} = \frac{P1r_y - P0r_y}{P1s_x};\ a_{11} = \frac{P3r_y - P0r_y}{P3s_y};$$

$$b_0 = P0r_x \text{ and } b_1 = P0r_y.$$

Once these coefficients are known, we can apply the transformation $A.Ps + B = Pr$ to each pixel of the image, as in Algorithm 2, where the pseudo-functions *calculateCoord*, *pixCoord* and *calcCoef* implement the calculations of subsections 4.2, 4.3 and 4.4, respectively. An example of perspective effect correction is shown in Figure 6.

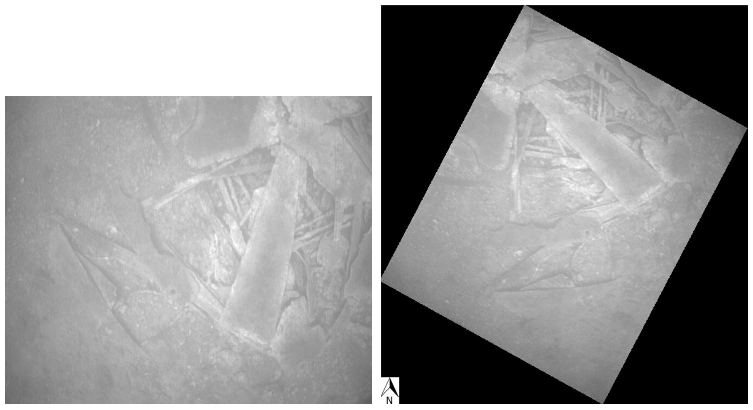
4.5 Lighting correction

It was observed that the images taken by the AUV were not homogeneously lighted: the middle of the picture was much brighter than the edges. This poses a problem for a clear mosaic, as shown in images *Online Resource 1* and *Online Resource 2*, available as multimedia supporting material.

The method presented here for lighting correction is similar to what is proposed in [21]. We were also inspired by [7], where lighting correction relies on finding a lighting model for all images in the mosaic. Initially, the plan was to use the most representative image of the seabed as a reference for the equalization of illumination. However, we realized that for several different reference images the resulting lighting model was very similar. So, we have chosen the first picture taken in the mission as the basis for the lighting correction. This first image is always taken at a altitude above 15 meters, so the seabed is blurred and we can easily deduce the lighting model.

Algorithm 2 LinTransf(in: $img_k, dh, \mathbf{p}_W^{AUV}, \varphi, \theta, \psi, k_{px}, t$)

```
1: $img_{res} := \emptyset$
2: $\mathbb{P} := calculateCoord(\mathbf{p}_W^{AUV}(t), \varphi(t), \theta(t), \psi(t), dh(t))$
3: $(P0r, P1r, P3r) := pixCoord(k_{px}, \mathbb{P})$
4: $(A, B) := calcCoef(img_k, P0r, P1r, P3r)$
5: $r = 0, c = 0, c_{res} = 0, r_{res} = 0$
6: while $r < nbrows(img_k)$ do
7:     while $c < nbcolumns(img_k)$ do
8:         $c_{res} := A(0,0).c + A(0,1).r + B(0)$
9:         $r_{res} := A(1,0).c + A(1,1).r + B(1)$
10:        $img_{res}(c_{res}, r_{res}) := img_k(c, r)$
11:        $r++, c++$
   return $img_{res}$
```



(a) (b)

Figure 6: The original image taken by the AUV (a) and the result after applying Algorithm 2 (b)

The image of reference (img_{calib}) is subdivided by a grid of 10×10 subregions, and the average gray level is calculated for each subregion. The gray level of each subregion is stored in a matrix, called the calibration matrix ($calib$), which has the same dimensions as the images taken by the AUV. Before it can be used, the calibration matrix is smoothed, as shown in Figure 7. The average gray level ($\bar{G}$) of the calibration image is also calculated.

Figure 7b represents the lighting model of the images taken in the mission. Therefore, the lighting correction of a generic image $img_k(i,j)$ is made by applying the model to all its pixels using the equation

$$img_k(i,j) = \frac{\bar{G} \times img_k(i,j)}{calib(i,j)}, \tag{11}$$

where (i,j) are the coordinates of a pixel in img_k.

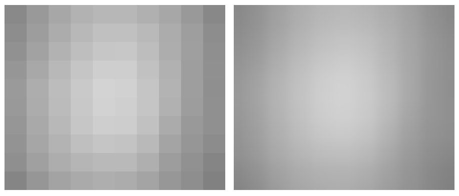


Figure 7: Lighting model represented by a calibration matrix raw (a) and smoothed (b)

5 Image Matching

Our objective with image matching is to deduce the robot displacement between two overlapping images. This calculation has two uses: mosaic synthesis and trajectory correction. The image matching method is the same for these two purposes, and will be explained in the following paragraphs.

5.1 The detection of interest points

The first step to match two images is to detect the significant points (called points of interest) of these two images. In other words, we must find, for each image, which pixels have an easily recognizable surroundings. Once these points of interest are detected, the next step is to match them.

There are several algorithms employed to allow the detection and mapping of significant points [12]. For this project, the SURF (Speeded Up Robust Features) [5] algorithm was used for the detection and characterization of points of interest. This choice was made due to the results presented in reference [12], as well as due to previous work done at ENSTA Bretagne, where SURF was successfully used. An interesting alternative to SURF is proposed in [24]. The only pre-made image processing used to assist in the interest point detection is a contrast enhancement by histogram equalization (see the difference by comparing figures 8 and 3).

SURF uses the "Fast Hessian detector" for interest point detection [5]. Consider a point $\mathbf{x} = (x,\ y)$ in an image I, the Hessian matrix H in $\mathbf{x}$ at a scale σ is defined by

$$H(\mathbf{x},\sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{xy}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma), \end{bmatrix}$$

where $L_{xx}(\mathbf{x},\sigma)$ is the convolution of the Gaussian second order derivative $\dfrac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point $\mathbf{x}$, and similarly for $L_{xy}(\mathbf{x},\sigma)$ and $L_{yy}(\mathbf{x},\sigma)$ [5]. Pixels with an abrupt variation in gray level in their surroundings are detected by calculating the Hessian determinant: the greater the determinant is, the more steeply the gray level varies. These pixels are selected as points of interest.

To be invariant to scale, "Fast-Hessian" is applied to an image pyramid made from recursive smoothing and downsampling, to get out of the base and achieve the highest level of the pyramid [5]. Points of interest are thus detected in all levels of the pyramid (multi-scale detection).

5.2 Feature extraction

After detection, the points of interest must be characterized, so we can identify them in different images. SURF uses the vector $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ as a descriptor of points of interest. The values d_x and d_y are the Haar wavelet response in the horizontal and vertical direction, respectively.

A quadratic grid oriented with 4×4 square subregions is defined around the interest point. Each square has a subdivision of 2×2, where the values of d_x and d_y are calculated and referenced to the prevailing orientation of the grid. This ensures the descriptor is invariant to image rotation. The dominant orientation of the grid is deduced from a sliding rotating window, which is explained in detail in [5] and [6]. The descriptor of SURF has a total of 64 dimensions (4 summations calculated in 16 sub-regions).

5.3 Matching interest points

Now that the points of interest have been detected and characterized, it is possible to find them in different images. The mapping of corresponding points is based on the *k-Nearest Neighbor* algorithm using some filters of false matches inspired by references [18] and [23]. Four filtering tests have been established.

Consider two images: img_A and img_B. The first step is simply to detect and store the coordinates of the feature points in sets kp_A and kp_B, where

$$kp_A = \{(x^0, y^0), (x^1, y^1), ..., (x^n, y^n)\}$$

is a set with n key-points of img_A (idem for kp_B). Their descriptors will be computed and stored in sets des_A and des_B, where

$$des_A = \{\{\mathbf{v}^{00}, ..., \mathbf{v}^{015}\}, ..., \{\mathbf{v}^{n0}, ..., \mathbf{v}^{n15}\}\}$$

has n descriptors of key-points in kp_A. Then, the *k-Nearest Neighbor* algorithm will choose the two best (nearest) matches in des_B, and vice versa, for each element of des_A. Hence, two sets are generated, $matches_{AB}$ and $matches_{BA}$, where

$$\begin{aligned} matches_{AB} = &\{\{\{Idx_A^0, Idx_B^{00}, dst^{00}\}, \{Idx_A^0, Idx_B^{01}, dst^{01}\}\} \\ &, ..., \\ &\{\{Idx_A^m, Idx_B^{m0}, dst^{m0}\}, \{Idx_A^m, Idx_B^{m1}, dst^{m1}\}\}\} \end{aligned}$$

has m matches. Idx_A is the index of the descriptor in the set des_A and dst is the distance between the matched descriptors.

The choice of the two best matches is based on the euclidean distance between descriptors. The smaller this distance is, the more similar the compared features are. If the measured distance is very low for the best match and much larger for the second best match, we can safely accept the first match as the right one, since it is unambiguously the best choice. On the contrary, if these distances are very close, there is a real possibility of making an error, and both matches should be rejected. Therefore, pairs will be selected if the ratio of the distance of the best match to the distance of the second best match is not greater than the empirical threshold. This first filtering test is called the *ratio* test.

A lot of ambiguity is removed with the ratio test, and the sets of correspondences $matches_{AB}$ and $matches_{BA}$ are relatively reliable. We will now extract the matches that are in agreement with both sets through a symmetry test. This test requires that both points must be the best candidate for each other for a match pair to be accepted.

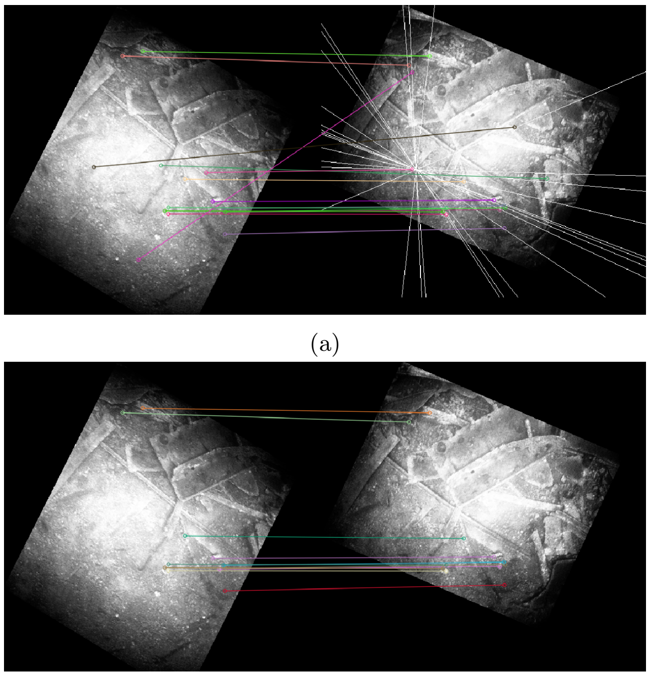
Even with the symmetry test, it is still possible to have mismatches, particularly with images containing very similar objects. A third test taking into account the location of objects in the images should be applied. This additional filtering test is based on RANSAC, and uses the fundamental matrix to reject matches that do not obey the epipolar constraint. This constraint imposes that for a match pair to be accepted, the feature point of the second image should be on the epipolar line of its corresponding pair in the first image.

Algorithm 3 DevTest(in: $matches_R, kp_A, kp_B$)

```
1: $\mathbb{S} := \emptyset$, $matches_F := \emptyset$, $i := 0$
2: while $i \leq length(matches_R)$ do
3:     $pos_A := kp_A(matches_R(i) \rightarrow Idx_A)$
4:     $pos_B := kp_B(matches_R(i) \rightarrow Idx_B)$
5:     $\delta := pos_B - pos_A$
6:     $\mathbb{S} := \mathbb{S} \cup \delta$
7:     $i++$
8: $i := 0$
9: while $i \leq length(\mathbb{S})$ do
10:    $shift := \mathbb{S}(i)$
11:    if $\overline{\mathbb{S}} - \sigma_{\mathbb{S}} \leq shift \leq \overline{\mathbb{S}} + \sigma_{\mathbb{S}}$ then
12:        $matches_F := matches_F \cup matches_R(i)$
   return $matches_F$
```

The fourth and final test is the test of the standard deviation. It is necessary because there may be a false match on an epipolar line which will not be filtered by the RANSAC method. An example is shown in Figure 8, where we see that the displacements of the false matching points are too far from the mean shift of the rest of interest points. So, the false matches can be filtered only if we consider the matching points whose displacement is near to the mean shift to be valid. The deviation test is an innovative algorithm presented in this paper, and it is not explained in reference [18] as the others are. Therefore, we present it in Algorithm 3, where $matches_R$ is the matching set issued from the RANSAC test, and $\overline{\mathbb{S}}$ and $\sigma_{\mathbb{S}}$ are the average and the standard deviation of $\mathbb{S}$.

It may seem inconsistent if we assume a quasi-planar marine relief and, at the same time, we use RANSAC, which is an algorithm based on a fundamental matrix model. Indeed, this approach can only yield good results if the objects placed on the sea-floor have a small height relative to the robot's overflight altitude. This is the most important limitation of the proposed feature based mosaic building technique. It is also the main reason for which navigation methods for mosaic building are more reliable, since there is no way that wrong 3D models or limitations in image matching jeopardize the mosaic quality.


(a)

(b)

Figure 8: Filtering of RANSAC outliers with a standard deviation test. (a) Mismatches do not filtered by RANSAC. The white lines are the epipolar lines. (b) Good matches preserved after applying the standard deviation test

6 Refinement of Trajectory

In this section, we compare the proprioceptive and exteroceptive data to refine the AUV trajectory and make better underwater mosaics.

The trajectory adjustment is made from the comparison between the position estimation issued from the INU and a correction proposed by a pair of overlapping images selected from the t-plane. Considering a pair of images, we can associate it with an interval representing the AUV displacement between the image-taking instants. So, if we compare the position given by the INU with the displacement calculated by image matching techniques, the certainty of positioning will be improved, since it must obey both inertial and visual constraints.

First we calculate the AUV displacement between two overlapping images. As we are interested in a displacement in meters referenced to the world frame, we cannot use the procedure of Algorithm 3. Otherwise, it is calculated as in Algorithm 4, where $\mathbb{P}_A(0)$ and $\mathbb{P}_B(0)$ are the top left corner coordinates (in meters) of images A and B. $\overline{\mathbb{D}}$ and $\sigma_{\mathbb{D}}$ are the mean and the standard deviation of $\mathbb{D}$.

If the images A and B were taken in t_1 and t_2, the new positions $[\mathbf{p}](t_1)$ and $[\mathbf{p}](t_2)$ will be reset as

$$\begin{array}{c}[\mathbf{p}]\,(t_1) = [\mathbf{p}](t_1) \cap ([\mathbf{p}](t_2) + [\Delta]) \\ [\mathbf{p}]\,(t_2) = [\mathbf{p}](t_2) \cap ([\mathbf{p}](t_1) - [\Delta])\end{array}.$$

Algorithm 4 DeltaPos($in : matches_F, kp_A, kp_B, k_{px}, \mathbb{P}_A, \mathbb{P}_B$)

```
 1: $\mathbb{D} := \emptyset, i := 0$
 2: while $i \leq length(matches_F)$ do
 3:     $P0r_A := pixCoord(k_{px}, \mathbb{P}_{\mathbb{A}}(0))$
 4:     $P0r_B := pixCoord(k_{px}, \mathbb{P}_{\mathbb{B}}(0))$
 5:     $pos_A := \mathbb{P}_A(0) - \frac{kp_A[matches_F(i)\to Idx_A] - P0r_A}{k_{px}}$
 6:     $pos_B := \mathbb{P}_B(0) - \frac{kp_B[matches_F(i)\to Idx_B] - P0r_B}{k_{px}}$
 7:     $dpos := pos_B - pos_A$
 8:     $\mathbb{D} := \mathbb{D} \cup dpos$
 9: $[\Delta] = [\overline{\mathbb{D}} - \sigma_{\mathbb{D}}, \overline{\mathbb{D}} + \sigma_{\mathbb{D}}]$
10: return $[\Delta]$
```

Note that both positions are adjusted, in times t_1 and t_2, since both are uncertain. The goal of this procedure is to reduce this uncertainty through the comparison of inertial and visual information. This point adjustment is propagated for the entire trajectory as in Algorithm 5, where $[\mathbf{p}]$, $[\mathbf{v}]$, t_{max} and $d\tau$ were presented in Section 2. If more image pairs are selected, more adjustments can be done and the trajectory estimate will be more certain. An example of successive trajectory adjustments is shown in a video (Online Resource 8) uploaded as multimedia supporting material. Figure 9 shows the difference between the original and adjusted trajectory.

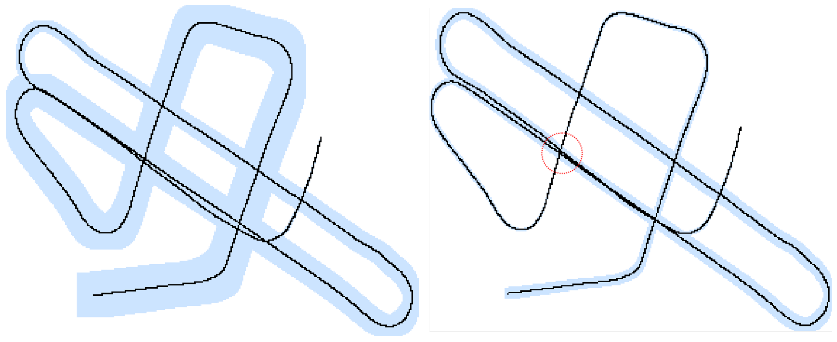
Algorithm 5 TrajRef($in : [\mathbf{p}], [\mathbf{v}], t_1, t_2, t_{max}, d\tau, [\Delta]$)

```
 1: $[\mathbf{p}](t_1) = [\mathbf{p}](t_1) \cap ([\mathbf{p}](t_2) + [\Delta])$
 2: $[\mathbf{p}](t_2) = [\mathbf{p}](t_2) \cap ([\mathbf{p}](t_1) - [\Delta])$
 3: $k = t_1$
 4: while $k \leq t_{max}$ do
 5:     $[\mathbf{p}](k+1) = [\mathbf{p}](k+1) \cap ([\mathbf{p}](k) + [\mathbf{v}](k).d\tau)$
 6:     $k++$
 7: $k = t_2$
 8: while $k > 0$ do
 9:     $[\mathbf{p}](k) = [\mathbf{p}](k) \cap ([\mathbf{p}](k+1) - [\mathbf{v}](k).d\tau)$
10:     $k--$
    return $[\mathbf{p}]$
```

7 Building Mosaics

The mosaic is built after the path adjustment, so that the robot position is consistent with the proprioceptive and exteroceptive observations. This will make a mosaic more faithful to the reality of the seabed. Two techniques have been developed for mosaic building: one uses only the inertial data from the INU and the other uses the tools of image matching described in Section 5.

The main advantages of the first method over the image matching tools are reduced computation time and heightened robustness to seabed changes. With image



(a) Original trajectory (b) Adjusted trajectory

Figure 9: Example of trajectory refinement. The red circle indicates the place of adjustment

matching, if the robot drives on a sandy bottom, for example, it will be much more difficult to find significant points in the images. The attitude and velocity data obtained from the INU, on the contrary, are always reliable and we are confident the mosaic is created faithful to reality, despite sometimes having less net picture fittings than we could have using the techniques of image matching.

7.1 Mosaic from inertial data

The first step is to determine the mosaic size. Consider now that

$$\begin{aligned}\mathbb{P} =&\{\{\mathbf{p}_W^A(t_0), \mathbf{p}_W^B(t_0), \mathbf{p}_W^C(t_0), \mathbf{p}_W^D(t_0)\},\\ &..., \{\mathbf{p}_W^A(t_{\max}), \mathbf{p}_W^B(t_{\max}), \mathbf{p}_W^C(t_{\max}), \mathbf{p}_W^D(t_{\max})\}\}\end{aligned}$$

is a set containing the coordinates in meters of the corners of *all* images taken during a mission. Therefore, the mosaic size in square meters is calculated as follows:

$$\begin{gathered}\Delta P_x = \max(\mathbb{P}_x) - \min(\mathbb{P}_x) \text{ and}\\ \Delta P_y = \max(\mathbb{P}_y) - \min(\mathbb{P}_y),\\ \text{so, } S = \Delta P_x \times \Delta P_y.\end{gathered}$$

The conversion to pixels is simply made by $S_{pix} = k_{pix}^2.S$, and the coordinates in pixels of the corners of an image img_k to be added to a mosaic img_{mos} are calculated using Equation 10.

Algorithm 6 describes the method of building mosaics using only inertial data, based on calculations and functions already presented in previous sections. It is similar to Algorithm 2; however, $\mathbb{P}$ contains the coordinates of corners of *all* images taken during the mission. The function *lightCorrection* implements the lighting correction presented in Section 4.5.

7.2 Mosaic from image matching

For construction of the mosaic from inertial data, it is also necessary for the image matching method to first undergo the image processing procedures (lighting correction

Algorithm 6 MosaicINU(in: $imgfile, dh, \mathbf{p}_W^{AUV}, \varphi, \theta, \psi, k_{px}$)

```
1:  $img_k := \emptyset, k := 0$
2:  $\mathbb{P} := calculateCoord(\mathbf{p}_W^{AUV}, \varphi, \theta, \psi, dh)$
3:  $(\Delta_x, \Delta_y) := calculateMosaicSize(\mathbb{P})$
4:  $img_{mos} := createVoidImage(\Delta_x, \Delta_y)$
5:  while $k \leq length(imgfile)$ do
6:      $img_k := read(imgfile(k))$
7:      $img_k := lightCorrection(img_k)$
8:      $(P0r, P1r, P3r) := pixCoord(k_{px}, \mathbb{P})$
9:      $(A, B) := calcCoef(img_k, P0r, P1r, P3r)$
10:     $r = 0, c = 0, c_{res} = 0, r_{res} = 0$
11:     while $r < nbrows(img_k)$ do
12:         while $c < nbcolumns(img_k)$ do
13:             $c_{res} := A(0,0).c + A(0,1).r + B(0)$
14:             $l_{res} := A(1,0).c + A(1,1).r + B(1)$
15:             $img_{mos}(c_{res}, r_{res}) := img_k(c, r)$
16:             $l++, c++$
17:     $k++$
    return $img_{mos}$
```

and distortion compensation). However, the location of the images in the mosaic is not defined by the coordinates of their corners in the world frame but by the average displacement of points of interest between successive images. With this method, the

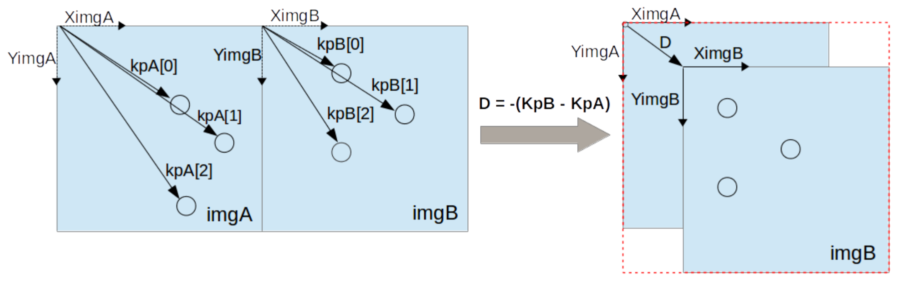


Figure 10: Connection between two images from the displacement of points of interest. The red rectangle defines the size of the mosaic after the addition of the new image img_B

mosaic is constructed from an initial image to be enlarged by the successive addition of images, as in Figure 10. Algorithm 7 describes the method, where $\mathcal{F}_{LT}$ is a function that implements Algorithm 2 and $addToMosaic(\mathbb{D}, img_{src}, img_{dst})$ is a function that places the image img_{src} in image img_{dst} displaced $\mathbb{D}$ pixels from the last added image.

Compared with the inertial method described previously, image matching provides better connections between images, so the mosaic becomes more readable and understandable. A disadvantage is that use of image processing tools always leaves the veracity of the results uncertain.

Algorithm 7 MosaicFeatures (in: $imgfile, dh, \mathbf{p}_W^{AUV}, \varphi, \theta, \psi, k_{px}$)

```
1: $img_A := \emptyset, img_B := \emptyset, img_{mos} := \emptyset, k := 1$
2: $img_A := read(imgfile(0))$
3: $img_A := lightCorrection(img_A)$
4: $img_A := \mathcal{F}_{LT}(img_A, dh, \mathbf{p}_W^{AUV}, \varphi, \theta, \psi, k_{px}, 0)$
5: $addToMosaic([0,0], img_A, img_{mos})$
6: while $k \leq length(imgfile)$ do
7:     $img_B := read(imgfile(k))$
8:     $img_B := lightCorrection(img_B)$
9:     $img_B := \mathcal{F}_{LT}(img_B, dh, \mathbf{p}_W^{AUV}, \varphi, \theta, \psi, k_{px}, k)$
10:    $(matches, kp_A, kp_B) := features(img_A, img_B)$
11:    for all elements of $matches$ do
12:        $pos_A := kp_A(matches \rightarrow Idx_A)$
13:        $pos_B := kp_B(matches \rightarrow Idx_B)$
14:        $\delta := -(pos_B - pos_A)$
15:        $\mathbb{D} := \mathbb{D} \cup \delta$
16:    $addToMosaic(\overline{\mathbb{D}}, img_B, img_{mos})$
17:    $img_A := img_B$
18:    $k++$
   return $img_{mos}$
```

8 Test Cases and Results

The techniques of mosaic building were implemented in software developed on QtCreator using the language C++ and the library *OpenCV* for image processing.

The software was tested with data from two missions carried out in the Mediterranean Sea by the AUV A9 of ECA Robotics. The first mission was on a wreck, where 2027 photos were taken during 10 minutes (sampling period of 40ms for navigation data). The second mission was on a posidonia field, with 4806 photos taken during 25 minutes and navigation data available every 500ms (even if measurements were made every 40ms). The mosaics from the wreck are available as multimedia supporting material (Online Resources 3 and 4).

In terms of performance, the mission on the wreck had the mosaic built by both methods; however, the mission over the posidonia had its mosaic made only by the inertial method, since the images were not rich enough in interest points. The 2027 photos taken on the wreck were mosaicked in 1782ms using only proprioceptive data, contrasted with 1062470ms if we use also image matching. It proves that the proprioceptive method is extremely fast, since we need less than $9ms$ to add a picture to the mosaic. The 4806 photos of the second mission were mosaicked in 8601ms (less than 2ms per photo). Even with more photos, the mosaic was built faster because there was less navigation data to be read. The tests were made on an ordinary computer with a 2.3GHz Quad Core processor and 4Gb of memory; Table 1 summarizes the comparison between both mosaicking methods.

The trajectory refinement was tested on the wreck mission, and an uploaded video (Online Resource 7) shows a reduction of 17% in the relative error of object placement in a loop. It mignt seem a small improvement, but if we consider that we pass from $94cm$ to $78cm$ of relative error, we realize that it is a significant result.

The uncertainty of positioning at the end of the mission was also reduced, from ($\Delta x = 14.37, \Delta y = 13.32$)m to ($\Delta x = 9.14, \Delta y = 5.58$)m. To better be tested, trajectory refinement should be applied to longer missions, so the uncertainties of measurement can accumulate and be more relevant.

The use of interval algebra for trajectory refinement has the additional advantage of low computational effort, when compared to alternative Bayesian approaches [16], [11]. For example, Algorithm 5 is run on $5ms$ when applied to the mission over a wreck, and all the navigation data with all its uncertainties can be stored in a $1Mb$ file. Interval algebra seems then to be an appropriate approach for future on-line SLAM.

With respect to mosaicking quality, we achieve, in consecutive image blending, a root means square pixel reprojection deviation of $(x = 2.1, y = 2.0)px$, calculated on a set of 233,203 matched key-points (see Table 1). If we don't use image matching, we have a deviation of $(x = 6.2, y = 9.0)px$, which is also a significant result, since we have used $k_{px} = 60$ and, therefore, the uncertainty of positioning of objects on the seafloor is $(x = 10.3, y = 15.0)cm$. Such data attest to the quality and reliability of the mosaicking methods proposed in this paper.

Mosaicking method / Measurement	Only navigation	With feature extraction
Root Means Square deviation (x, y)	$(x = 6.2, y = 9.0)$ px	$(x = 2.1, y = 2.0)$ px
Time of mosaicking	17, 824 ms	1, 062, 470 ms

Table 1: Comparison between mosaicking methods in a mission on a wreck (2027 photos)

9 Additional Comparisons to Other Work

Underwater mosaicking techniques have been extensively developed during the past 10 years, and some prior relevant contributions must be mentioned and compared to those in this paper. Mosaicking using feature extraction and navigation data was achieved by the authors of [10]. Each image of the mosaic was assigned to the camera geographical location and rotated in heading, as pitch and roll were assumed to be stable enough. In this paper, we have presented a more complete approach, since a three-dimensional rotation was applied to correct perspective distortions, and all the mosaic pixels have their equivalent coordinates in meters. The authors of [21] also use 3D rotation for perspective correction while creating continuous geo-referenced image tiles using only navigation data. However, their method was tested on the creation of one-dimensional mosaics, while in this paper we are in a 2D context.

Regarding feature based mosaicking, Pizarro and Singh (2003) [22] have presented an effective method for producing mosaics from ROV (Remotely Operated Vehicle) images. Unfortunately, their method would not be efficient over large areas, with small errors magnifying along the trajectory, particularly in featureless seabed zones [21]. In our case, we do not have this problem since we can use navigation data for mosaic creation in featureless zones.

10 Conclusion

This paper has presented two methods of underwater mosaicking: one is based on image matching and is an evolution of current techniques [18][23]; the other is an innovative method based only on the analysis of proprioceptive data. The navigation uncertainty is taken into account and can be reduced by a trajectory refinement method based on interval algebra.

The main contribution of this paper is the use of interval algebra to handle the inherent uncertainties of navigation data. The reliability of mosaic building is then improved by applying the proposed method of trajectory refinement, which is only possible with interval methods; to our knowledge, there are no probabilistic methods to reliably detect loops or overlapping images with inertial data. We can also say that, for the first time, proprioceptive (or inertial) data are used to reliably detect loops, to avoid matching errors. As demonstrated in the end of Section 8, this increased the internal consistency of the mosaic by 17%, with a relative placement error of objects in a loop smaller than $78cm$.

A secondary improvement for image matching is also proposed, where a deviation test was presented as a new outlier rejector, in addition to the widely known RANSAC. This new test is important for improvement of the reliability of mosacking, since the motion estimation is strongly tied to the quality of correspondences and RANSAC has proved to be flawed in some cases. Another improvement is the association of images with their corner coordinates. This makes it possible to correct distortions and orient all images towards the north. Since all images have the same orientation and visual perspective, we can directly compute the displacement of key points between images (Figure 10) and blend them more easily. Computational performances were presented in Section 8.

In the future we plan to combine the two presented mosaicking methods, to simultaneously have reliable, clear and sharp mosaics.

11 Supplementary Materials

This paper has supplementary multimedia files. The material is 125.6Mb in size and includes:

Online Resource 1 : A piece of mosaic without lighting correction

Online Resource 2 : A piece of mosaic with lighting correction

Online Resource 3 : Mosaic of a wreck using only INU data

Online Resource 4 : Mosaic of a wreck using also feature extraction

Online Resource 5 : Example of mosaic creation using image matching

Online Resource 6 : Example of mosaic creation using only navigation data

Online Resource 7 : Improvements brought with trajectory refinement

Online Resource 8 : Example of trajectory correction

These are available at:
`https://sites.google.com/site/matheuslaranjeira1/projects/seabed-mosaics/`

12 Biographies of the Authors

Matheus Laranjeira was born is Salvador, Brazil, in 1990. He graduated in robotics at ENSTA Bretange (France, 2014) as part of the exchange program BRAFITEC. In 2015 he received his Brazilian diploma in electrical engineering from the Federal University of Bahia. His research is focused on underwater robotics, trajectory estimation, control and video servoing.

Luc Jaulin was born in Nevers, France in 1967. He received the Ph.D. degree in automatic control from the University of Orsay, France in 1993 and the Habilitation à Diriger des Recherches degree in 2000. He is currently full Professor of Robotics at the ENSTA (Ecole Nationale Suprieure de Techniques Avances) Bretagne, engineering school at Brest, France since 2004. His research is on underwater and sailboat robotics using interval methods and constraint propagation in the OSM team (Ocean Sensing and Mapping). He is the co-author of more than 150 papers in international journals and congresses. He received the Moore Prize in 2012 for his work on localization and map building of underwater robots using interval constraint propagation.

Sebastien Tauvry received his Masters Degree in Signal Processing from the Graduate School of Engineering of the University of Rennes 1 in France in 1997. He has been product manager for marine data processing at ECA Group, a leading company in underwater robotics for the past 40 years. He has more than 15 years in underwater payload experience and was system architect for the AUV A9.

References

[1] F. Abdallah, A. Gning, and P. Bonnifait. Box particle filtering for nonlinear state estimation using interval analysis. *Automatica*, 44:807–815, 2008.

[2] C. Aubry, R. Desmare, and L. Jaulin. Loop detection of mobile robots using interval analysis. *Automatica*, 49(2):463–470, 2013.

[3] F. Le Bars, A. Bertholom, J. Sliwka, and L. Jaulin. Interval SLAM for underwater robots; a new experiment. *NOLCOS*, pages 42–47, 2010.

[4] F. Le Bars, J. Sliwka, O. Reynet, and L. Jaulin. State estimation with fleeting data. *Automatica*, 48(2):381–387, 2012.

[5] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. *ECCV*, 3951:404–417, 2006.

[6] H. Bay, T. Tuytelaars, L. V. Gool, and A. Ess. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110:346–359, 2008.

[7] M. Borgetto, V. Rigaud, and J. F. Lots. Lighting correction for underwater mosaicking enhancement. *The 16th International Conference on Vision Interface*, 2003.

[8] V. Drevelle and P. Bonnifait. High integrity GNSS location zone characterization using interval analysis. *ION GNSS*, pages 2178–2187, 2009.

[9] Cyril Drocourt, Laurent Delahoche, Eric Brassart, Bruno Marhic, and Arnaud Clérentin. Incremental construction of the robot's environmental map using interval analysis. In Christophe Jermann, Arnold Neumaier, and Djamila Sam, editors, *Global Optimization and Constraint Satisfaction*, pages 127–141. Springer Berlin Heidelberg, 2005.

[10] J. Escartn, R. Garca, O. Delaunoy, J. Ferrer, N. Gracias, A. Elibol, X. Cufi, L. Neumann, D. J. Fornari, S. E. Humphris, and J. Renard. Globally aligned photomosaic of the lucky strike hydrothermal vent field (Mid-Atlantic Ridge, 3718.5n): Release of georeferenced data, mosaic construction, and viewing software. *Geochemistry, Geophysics, Geosystems*, 9(12):(unpaginated), 2008.

[11] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS Titanic with SLAM information filters. In *Proceedings of Robotics Science and Systems*, pages 57–64, Cambridge, MA, June 2005. MIT Press.

[12] S. Gauglitz, T. Hollerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94:335–360, 2011.

[13] N. Gracias and J. Santos-Victor. Automatic mosaic creation of the ocean floor. *OCEANS '98 Conference Proceedings*, 1:257–262, 1998.

[14] N. Gracias and J. Santos-Victor. Underwater video mosaics as visual navigation maps. *Computer Vision and Image Understanding*, 79:66–91, 2000.

[15] R. Haywood. Acquisition of a micro scale photographic survey using an autonomous submersible. *OCEANS '86*, pages 1423–1426, 1986.

[16] L. Jaulin. A nonlinear set-membership approach for the localization and map building of an underwater robot using interval constraint propagation. *IEEE Transaction on Robotics*, 25(1):88–98, 2009.

[17] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, 2001.

[18] R. Laganiere. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.

[19] R. L. Marks, S. M. Rock, and M. J. Lee. Real-time video mosaicking of the ocean floor. *IEEE Journal of Oceanic Engineering*, 20:229–241, 1995.

[20] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[21] K. J. Morris, B. J. Bett, J. M. Durden, V. A. I. Huvenne, R. Milligan, D. O. B. Jones, S. McPhail, K. Robert, D. M. Bailey, and H. A. Ruhl. A new method for ecological surveying of the abyss using autonomous underwater vehicle photography. *Limnology and Oceanography: Methods*, 12:795–809, 2014.

[22] O. Pizarro and H. Singh. Towards large area mosaicing for underwater scientific applications. *IEEE J. Oceanic Eng.*, 28:651–672, 2003.

[23] R. Prados, R. Garcia, and L. Neumann. *Image Blending Techniques and their Application in Underwater Mosaic*. Springer, New York, 2014.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT and SURF. *ICCV*, pages 2564–2571, 2011.