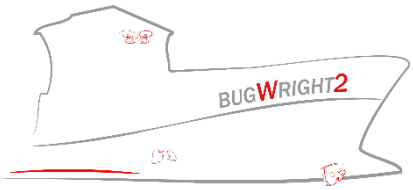
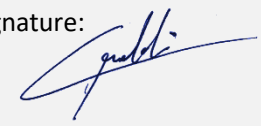


Autonomous Robotic Inspection and Maintenance on Ship Hulls and Storage Tanks

Deliverable report – D4.3

Context	
Deliverable title	3D modeling
Lead beneficiary	CNRS
Author(s)	UIB, NTNU, CNRS, UNI-KLU
Work Package	WP04
Deliverable due date	September 2023 (M45)
Document status	
Version No.	1.0
Type	REPORT
Dissemination level	**Public**
Last modified	12 décembre 2023
Status	RELEASED
Date approved	12 décembre 2023
Approved by Coordinator	Prof. Cédric Pradalier (CNRS) Signature: 
Declaration	Any work or result described therein is genuinely a result of the BUGWRIGHT2 project. Any other source will be properly referenced where and when relevant.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 871260

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

REFERENCED DOCUMENTS

- Deliverable D2.1 – *Crawler adaptation to BUGWRIGHT2's requirements*
- Deliverable D2.2 – *AUV adaptation to BUGWRIGHT2's requirements*
- Deliverable D2.3 – *MAV adaptation to BUGWRIGHT2's requirements*
- Deliverable D4.1 – *Localisation*
- Deliverable D4.2 – *Local Mapping and Obstacle Perception*

- Deliverable D5.1 – *Unified Control Interfaces*
- Deliverable D5.2 – *Autonomous Trajectory Tracking and Obstacle Avoidance*
- Deliverable D5.3 – *Single-Robot Planning, Coverage and Mission Execution*

These documents are available on the Nextcloud.

HISTORY OF CHANGES

Date	Written by	Description of change	Approver	Version No.
19/09/2023	UIB	Starting document		v0.0
23/09/2023	NTNU	Contributions in section IV		v0.1
28/09/2023	UIB	Contributions in section III		v0.2
//2023	UIB	Contributions in section 0		v0.3
//2023	UIB	Contributions in sections I and VI		v0.4
//2023	CNRS	Contribution of section V		v0.5
/12/2023	CNRS	Proofreading		v0.6
07/12/2023	CNRS	Document finalization and validation		v1.0

ABBREVIATIONS

AUV	*Autonomous Underwater Vehicle*
BW2	*BugWright2*
DF	*Distance Field (or Function)*
EKF	*Extended Kalman Filter*
ESDF	*Euclidean Signed Distance Field (or Function)*
FL-MBS	*Forward-Looking MultiBeam Sonar*
FMU	*Flight Management Unit*
FOV	*Field of View*
FSE	*Full State Estimator*
ICP	*Iterative Closest Point*
IMU	*Inertial Measurement Unit*
LiDAR	*Light Detection And Ranging (or Laser Imaging Detection and Ranging)*
LiODOM	*Lidar-only ODOMetry*
LUT	*Look-Up Table*
MAV	*Micro-Aerial Vehicle*
MCA	*Marching Cubes Algorithm*
PF	*Particle Filter*
PoI	*Point of Interest*
SDF	*Signed Distance Field (or Function)*
TSDF	*Truncated Signed Distance Field (or Function)*
UWB	*Ultra-Wide Band*

Executive summary

Deliverable D4.3 focuses on the generation of a 3D model of the outer hull of the vessel under inspection. The model is intended to be built from the data collected by aerial and underwater platforms, in order to cover both underwater and above-water parts of the vessel. Apart from the inherent interest in modelling the inspected surfaces as an additional result of the inspection operations, the resulting models are intended to be useful during the navigation of the climbing robots involved in BUGWRIGHT2 inspections.

I. Introduction

The project BW2 aims at combining the survey capabilities of autonomous *Micro-Aerial Vehicles* (MAV) and small *Autonomous Underwater Vehicles* (AUV), with teams of magnetic-wheeled *climbing robots* operating directly on the surface of the structure, within a common, two-stage inspection framework. Within the BW2 inspection framework, Stage 1 is carried out by means of a MAV that performs an initialization flight during which data from the area of operation are collected to (1) give rise to a common localization framework for all robots involved in Stage 2, and (2) provide a global overview of the structure above-water to facilitate the planning of Stage 2; the remaining robots implement Stage 2, where the rest of the inspection operation takes place making use of the common localization framework, aiming at a detailed inspection above- and underwater.

Focusing on Stage 1, it provides a 3D model of the non-submerged surfaces involved, together with other inspection data, e.g. images, that have been captured during the initialization flight which are processed in search for potential defects to be surveyed at a larger detail. By way of illustration, **Figure 1** shows an example of output for Stage 1. While the MAV is navigating in front of the hull during the initialization flight, it is collecting:

- the laser scans supplied by the onboard 3D laser scanner, which are used for motion estimation, obstacle detection and 3D modelling; and
- also the RGB images obtained from the camera onboard.

Regarding the latter, the collected images are supplied with indication of the presence of defects, if any, and their approximate localizations on the hull, as shown in **Figure 1**. This figure combines the 3D model resulting from a real flight with yellow and red circles that represent positions over the hull. The position of a specific circle is an approximation of the projection of the central point of the corresponding image. This is calculated making use of the depth data supplied by the onboard laser scanner at the time of image capture. In order not to overwhelm the user, a selection of all the images collected by the MAV during the initialization flight is made, i.e. 1 out of 10, 20, etc. and then processed to determine their projections and which ones could correspond to a defective area of the hull, if any. In the figure, the yellow circles denote image positions, while the red circles are for images suspected of containing defects.

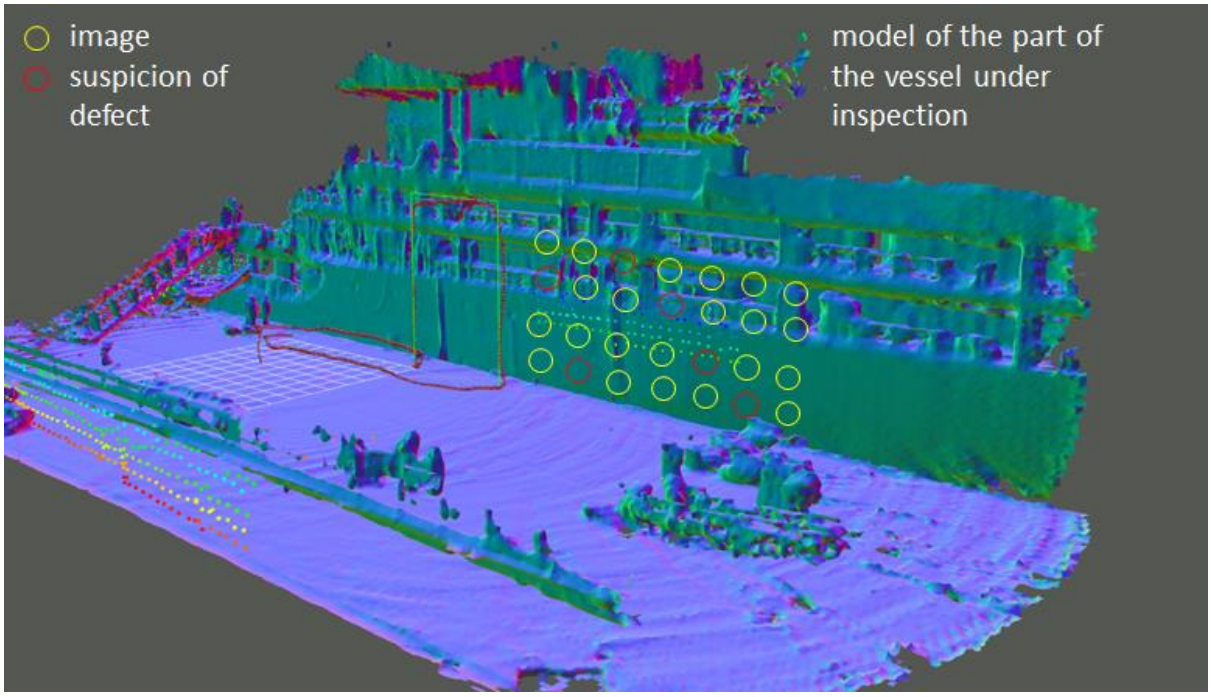


Figure 1: Illustration of the output of stage 1 of the BW2 inspection framework: 3D model (greenish-blueish mesh) and 3D location of the centre of the images captured (yellow circles), with indication of images containing defects (red circles)

Deliverable D4.3 focuses on the 3D modelling functionalities available within the BW2 inspection framework. For this to happen, the corresponding platforms need to be able to navigate in front of or over the structure, estimate its motion and process sensor data. Deliverables D4.1 – *Localisation*, D4.2 – *Local Mapping and Obstacle Perception*, D5.1 – *Unified Control Interfaces* (already released), D5.2 – *Autonomous Trajectory Tracking and Obstacle Avoidance* and D5.3 – *Single-Robot Planning, Coverage and Mission Execution* describe complementary aspects of the control architectures of the respective BW2 platforms. Consequently, a certain level of overlap can be expected between the contents of the aforementioned deliverables and this document, as a result of trying to make every deliverable a reasonably self-contained report.

The rest of this document is organized as follows: Section 0 addresses the problem of aligning the coordinate frames of the different robotic platforms so that data produced by Stage 1 can be used by the platforms involved in Stage 2, ensuring that overall mission planning can be achieved for this stage, and, furthermore, that the inspection data collected can be visualized in a comprehensible way; Section III describes the 3D modelling process for the above-water part of the hull, performed by means of an aerial platform; Section IV focuses on the generation of a model for the submerged part of the hull by means of an AUV operating within Stage 2; Section V deals with the generation of 3D models by means of a high-accuracy 3D topographic scanner, such as the Leica Total Station MS60 (Robotic Total Station); finally, Section VI summarizes and concludes deliverable D4.3.

II. Frame alignment issues

When several robotic platforms have to operate collaboratively on a common task, so that the data captured, and perhaps processed, by one platform can be employed by another, it is necessary to ensure that they all make the same interpretation, i.e. they understand the same from the data. A first condition for this to happen is the alignment of the coordinate frames that are associated with the data handled. In the case of BW2, this is not only necessary to synchronize the robots' views but also to properly display in a comprehensible way the data collected, for both Stage 1 and Stage 2, and as well to allow for overall mission planning during Stage 2.

The common localization framework is built within Stage 1, using the data collected during the initialization flight performed by the aerial platform depicted in **Figure 2**. This flight is intended to calibrate a set of positioning elements deployed throughout the area of operation before the initialization, which will be used by the robots of the Stage 2 for self-localization within the common framework. This set comprises:

- A network of Ultra-Wide Band nodes
- A collection of ARUCO markers

Additionally, an RTK GPS is available for global positioning. **Figure 3** illustrates the two stages and the localization infrastructure involved. As it is indicated in the figure, one of the ARUCO markers becomes the reference frame of the localization framework. This marker is rigidly attached to a submerged ARUCO that is employed by the AUVs mapping the hull underwater to synchronize their coordinate frames.

The fact that the reference frame is over the structure to inspect allows us to counteract the drift introduced by the vessel itself when it is in the water, subjected to short-range motion due to waves and to long-range motion due to tides (an illustration of the effects of the latter can be found in **Figure 4**). As all positions are estimated with regard to the same frame, which is rigidly attached to the structure, all measurements are understood in the same way.

Moreover, in order to avoid changes in the respective control architectures, every robot operates in its own coordinate frame, and it is the central planner who translates positions to the common localization framework using the initialization achieved through Stage 1, i.e. the central planner sends high-level commands transformed from the global user frame to the individual robot frame. This allows for a modular and easily extensible frame alignment approach that permits overall mission planning, individual task execution by the different robots and makes it possible to show understandable feedback to the user.

Stage 1 can run in a fully automated/autonomous way, including the initialization flight, the calibration of the localization devices, the generation of the 3D model above-water and the collection of the initial set of images of the hull and their processing in search of possible defects. Besides, due to the capabilities of the platform of **Figure 2**, it is also possible to perform the initialization flight in a semi-autonomous assisted mode if certain flexibility is required to reach certain positions.

Figure 2: Aerial platform implementing Stage 1 of the BW2 inspection framework

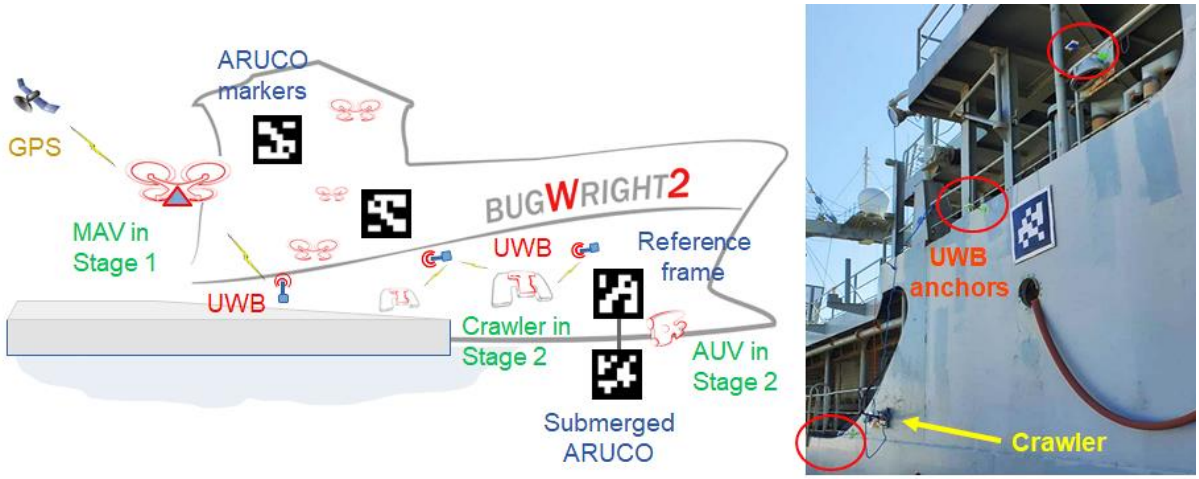


Figure 3: Illustration of Stage 1 and Stage 2, and the localization infrastructure involved, comprising an RTK GPS, ARUCO markers and a network of UWB nodes

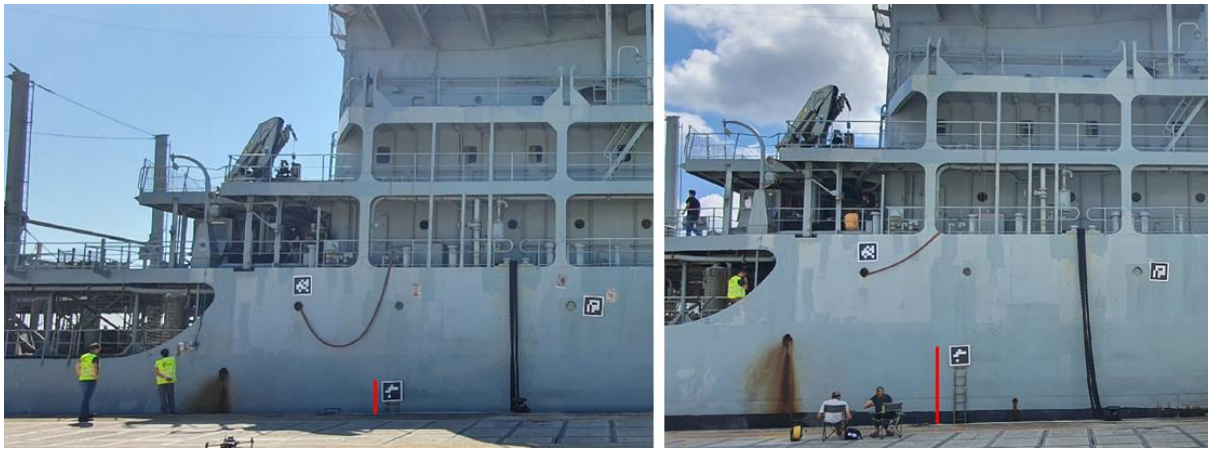


Figure 4: Effect of tides on the vessel apparent position: (left) morning, (right) afternoon

III. Hull modelling above-water

Hull modelling above-water is accomplished by means of the pipeline shown in **Figure 5**. The relevant information about the different pipeline stages can be found in the following sections.

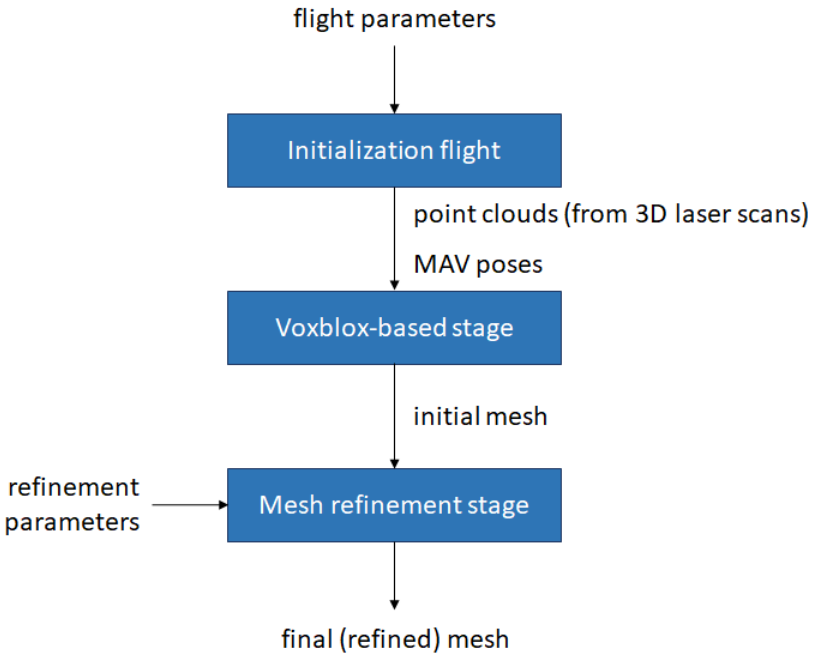


Figure 5: Pipeline involved in hull modelling above-water

1. Preliminaries

As already stated, the model of the hull above-water is part of the outputs of Stage 1. The MAV that implements this stage achieves the capabilities it is equipped with through the following sensor suite, part of which can be observed in Figure 2:

- A 3D laser scanner from Ouster, model *Ouster OS1-64 Gen 2*, which supplies structured 3D point clouds organized into 64 channels, with 2048 bins per channel (maximum), able to operate at 10 or 20 Hz. The maximum range is 100 m, for a Field of View (FOV) of +22.5° to -22.5° (vertical) and 360° (horizontal).
- A downward-looking LIDAR-Lite v3 single-beam laser range finder used to supply height data for a maximum range of 40 meters. This sensor is complemented with a barometric pressure sensor that is included in the *Flight Management Unit* (FMU) of the drone.
- An imaging system which can interchangeably consist of an RGB-D or a lighter RGB camera (the platform in Figure 2 is fitted with an RGB camera, although it cannot be observed as it is looking forward).
- GNSSS and RTK GNSS receivers.

They all feed the control software running onboard, which allows the MAV to estimate and plan its motion to achieve the intended inspection goals in a safe way, detecting the obstacles that might be on its way and taking the corresponding avoidance actions. Of particular relevance for this deliverable is the estimation of motion and the integration of the point clouds that are collected during navigation, both of which are involved in the generation of the hull model.

For a start, the state of the MAV comprises the platform pose $(x, y, z, \varphi, \theta, \psi)$, the linear velocities $(\dot{x}, \dot{y}, \dot{z})$ and accelerations $(\ddot{x}, \ddot{y}, \ddot{z})$, and the angular velocities $(\dot{\varphi}, \dot{\theta}, \dot{\psi})$. The *full state estimator* (FSE) module shown in **Figure 6** supplies these state estimates by fusing the available navigation data (once processed) and a number of different positioning sources. More precisely, the FSE module comprises two cascaded *Extended Kalman Filters* (EKF): the *local EKF* fuses navigation data from motion and relative-position sources, while the *global EKF* combines estimates from the *local EKF* with positioning data from global localization sources such as GPS and/or UWB-based position estimators (if available), as well as from SLAM algorithms (whichever are available) fed by the on-board sensors.

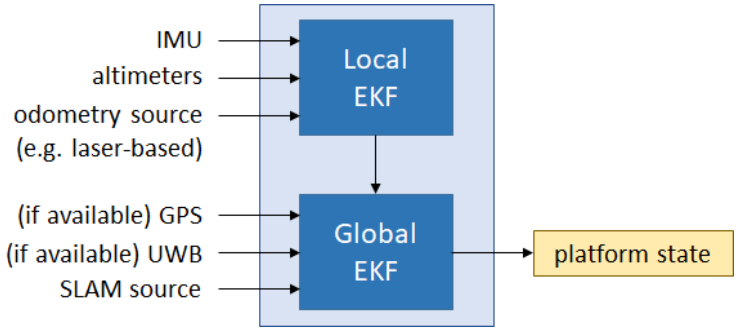


Figure 6: Full-state estimator module for the MAV involved in Stage 1

The odometry source that feeds the local EKF of the full state estimator of **Figure 6** is LiODOM (*LiDAR-only ODOMetry*), a laser-based motion estimation module developed within the framework of project BW2 by

the UIB. As its name suggests, LiODOM can compute the pose of the platform without the help of any other sensor, such as IMU, GPS, etc. As shown in **Figure 7**, it is fundamentally organized into two main components, *odometry* and *mapping*, which run concurrently:

- The *odometry module* consists in turn of two synchronized threads that decouple feature extraction from pose estimation. In the feature extraction stage, the received sweep S_i at time step i is divided into its different scans and points which do not fall within an interval $[r_{min}, r_{max}]$ are discarded. Next, a set of edge features E_i^L (as defined in LOAM[1]) are detected and selected in accordance to the local curvature within the input point cloud. Each scan is additionally split into sectors and a maximum number of points per sector is chosen to ensure an even distribution of point features throughout the environment for better optimization conditioning. The resulting set of edges is then processed by the pose estimation thread: (1) a set of point-to-line correspondences between edges and a local map m_i supplied by the *mapping module*, and (2) the optimal pose of the MAV is computed by means of non-linear optimization on the previous set of correspondences.
- The *mapping module* builds an unoptimized global representation of the environment $\mathrm{M_i}$ i.e. loop closures are not detected nor considered. From this global map, a local representation m_i is extracted according to the current pose of the platform. The local map is used by the pose estimation thread of the *odometry module* to find correspondences with the point cloud currently supplied by the 3D laser scanner.

To further illustrate LiODOM, Figure 9 shows the unoptimized global map resulting after a flight in front of the hull of a real vessel (the dataset was collected during the Lisbon field trials taking place in September 2023, details are provided in Section 4). A more comprehensive description of the control software running onboard the MAV of Stage 1, and particularly regarding LIODOM, can be found in deliverables D4.1, D4.2, D5.1 and D5.2.

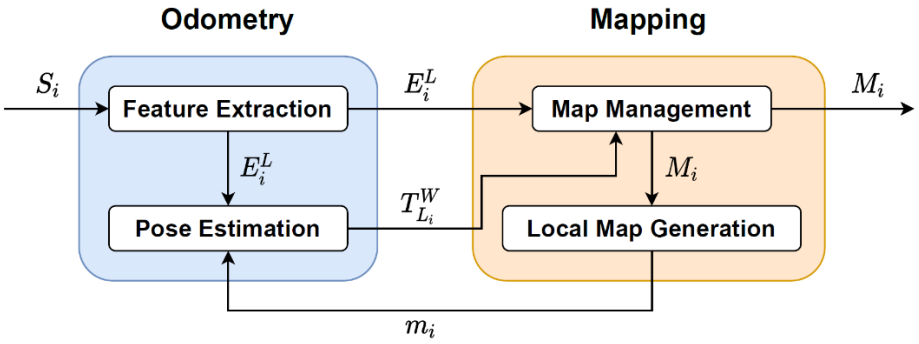


Figure 7: Overview of LiODOM

The final model of the vessel hull is obtained in the form of a mesh. This mesh is computed by means of the point clouds collected during the flight and the poses of the MAV estimated by LiODOM using a map (global point cloud) such as the one shown in **Figure 9**, which corresponds to one of the flights performed during the BW2 field trials that took place in Lisbon in September 2023, while surveying the hull of the

[1] Ji Zhang and Sanjiv Singh, LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. Robotics: Science and Systems Conference (RSS)*, 2014.

vessel shown in **Figure 8**. The consistency of this map is an indication of the accuracy in pose estimation available from LiODOM, what ultimately determines the model of the hull resulting from Stage 1.

Figure 8: Picture of the NRP Bérrio, the vessel involved in the BW2 field trials carried out in Lisbon, September 2023

To generate the mesh, we make use of Voxblox[2], a volumetric mapping library based on *Truncated Signed Distance Fields* (TSDFs). Section 2 next overviews the way how the Voxblox-based software generates an initial mesh, while Section 3 details the post-processing that is performed on this initial mesh in order to refine it and produce a representation of the hull more suitable for the purposes foreseen in BW2; finally, Section 4 reports on the results obtained for a selection of the flights performed during the Lisbon BW2 field trials.

2. Mesh generation

Figure 10 illustrates the workflow of the Voxblox-based software that produces the first version of the mesh, from the robot poses and the point clouds collected during motion to the 3D hull model at the output. As can be observed, the key element in Voxblox is a TSDF. A *Truncated Signed Distance Field* (or function) is a simplified way for representing 3D shapes and surfaces that is used in tasks such as 3D reconstruction and surface rendering to create accurate and efficient representations of 3D scenes. It is particularly useful for integrating many noisy incomplete sensor readings (even from different sensors, like depth images from a 3D camera or point clouds from a LiDAR sensor) into a single smooth and complete

[2] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart and J. Nieto, Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning, In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1366-1373, 2017.

model. Internally, it can be represented by a 3D voxel array after defining the volume that contains the 3D object/scene and the size of each voxel.

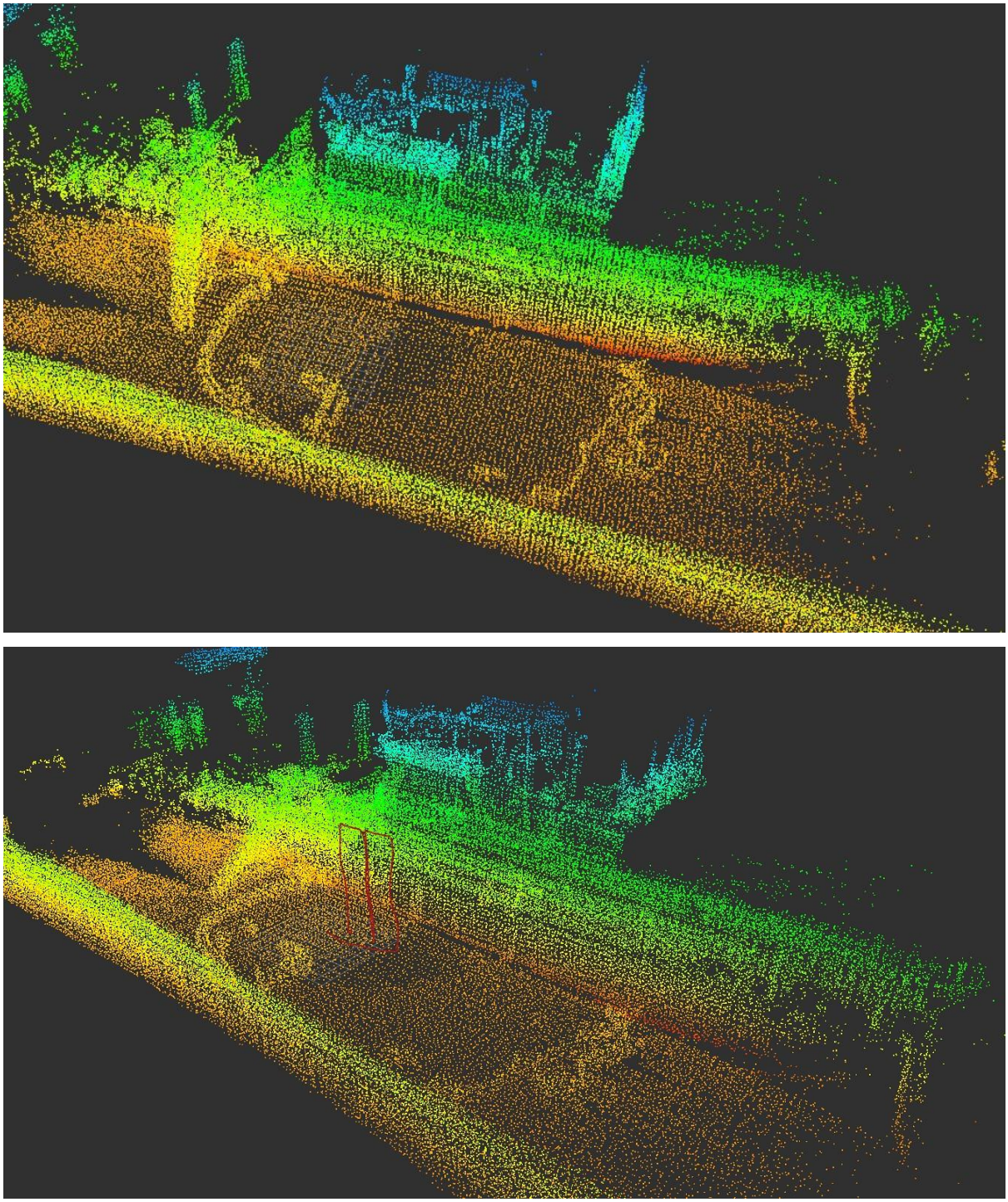


Figure 9: Example of map/point cloud produced by LiODOM, with the estimated MAV path superimposed (bottom) and without (top)

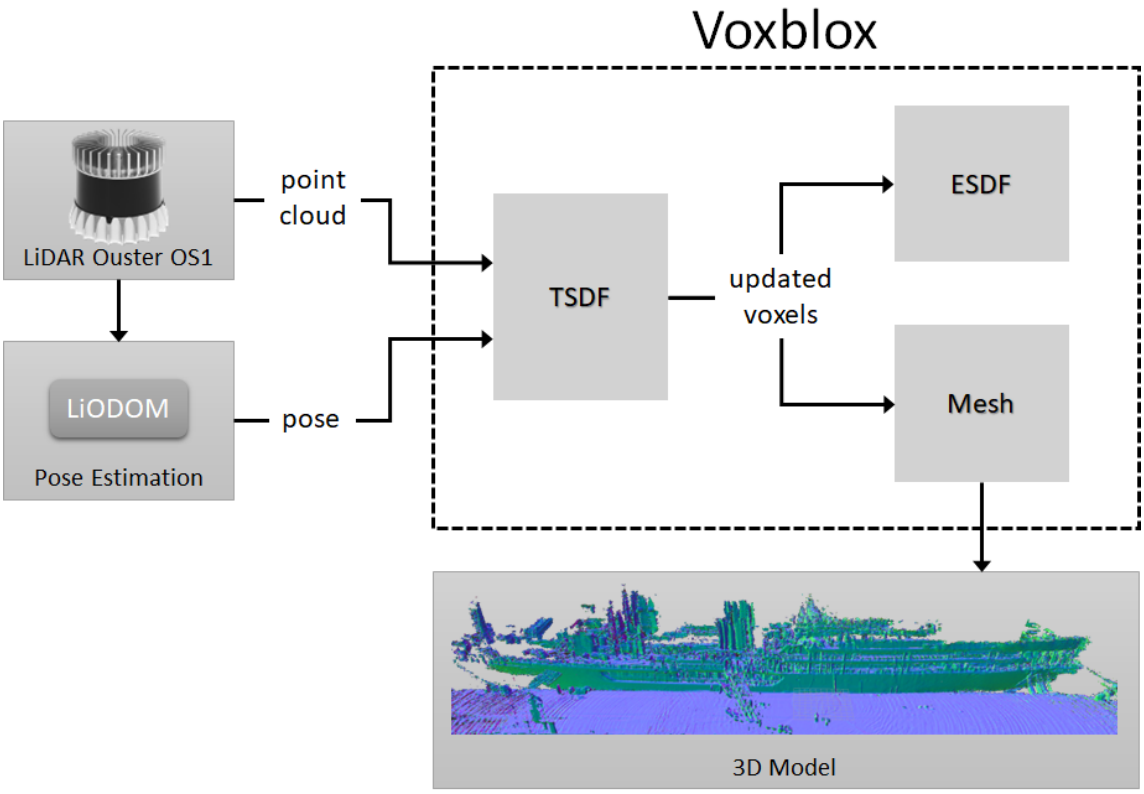


Figure 10: Overview of the workflow of the Voxblox-based software for mesh generation

The acronym TSDF is the result of the combination of different features:

- The basic element of a TSDF is the *distance field* (DF). It refers to the voxel array, where each voxel stores the metric distance from the voxel center to the closest point on the object surface/scene surface: voxels very far from the surface have high-magnitude distance values, while those near the surface have values approaching zero. A d-dimensional environment is represented as a d-dimensional grid of equally sized voxels.
- A SDF (*Signed Distance Field*) stores signed distances into voxels, where voxels in front of the object/surface (i.e. outside the object/surface, in free space) are filled with a positive distance, and voxels behind the surface (inside the object/surface) have a negative distance. Additionally, there is a weight for each voxel to express the uncertainty of the corresponding distance value, what allows integrating depth data from multiple sources by weighted summation, usually through iterative updates of the TSDF.
- In a TSDF, the range of distances that are stored is limited, i.e. truncated, so that we only care about distances up to a certain threshold $\pm\tau$, and ignore any distance beyond that. Truncation is beneficial since large distances are not relevant for surface reconstruction and a restriction of the value range can be utilized to simplify the representation of the 3D shape and reduce the amount of data that has to be stored, i.e. the memory footprint.

Due to its nature, TSDFs are well suited for data-parallel algorithms, i. e. for implementation on GPUs. The attained speed-up facilitates real time processing at high frame rate.

The Voxblox library provides different merging methodologies for generating the TSDF from the different depth readings with different levels of accuracy and efficiency. To build the 3D model we make use of *ray*

casting from the centre of the sensor to all the points in the cloud, computing the distance and weight for the voxels traversed by the ray, as this is the method that leads to the most accurate representation. In short, for a voxel centered at x, if $TSDF_{t-1}(x)$ denotes the integration of all observations $tsdf_j(x)$[3] up to time $t-1$ ($0 \leq j < t$), and $W_{t-1}(x)$ is the uncertainty of $TSDF_{t-1}(x)$, then a new observation $tsdf_t(x)$ with uncertainty $w_t(x)$ is integrated by means of weighted summation:

$$TSDF_t(x) = \frac{W_{t-1}(x)TSDF_{t-1}(x) + w_t(x)tsdf_t(x)}{W_{t-1}(x) + w_t(x)}$$

$$W_t(x) = W_{t-1}(x) + w_t(x)$$

The grid is initialized with $TSDF_0(x) = 0$ and $W_0(x) = 0, \forall x$. On the other side, $w_i(x)$ are usually modelled in accordance to the corresponding depth value, as depth sensing is usually more accurate at close range.

Input point clouds are transformed to the global coordinate frame where the TSDF is represented in accordance to the estimated pose at the corresponding time. Due to the possible drift associated to odometry sources in general, we incorporate an additional step involving ICP (Iterative Closest Point) alignment between the current input point cloud, after transformation to the global frame, and $TSDF_t$, which integrates all the previous depth measurements, to reduce the error in model generation.

As depicted in **Figure 10**, the output of the Voxblox-based software is twofold: an *Euclidean Signed Distance Field* representation (ESDF) and the surface reconstruction in the form of a 3D triangular mesh, being the former also usable for motion planning. The mesh is computed making use of the *Marching Cubes* algorithm (MCA)[4]. This algorithm aims at the generation of a triangle mesh from an implicit function, i.e. one of the form $f(x, y, z) = 0$, where in our case $f = TSDF$. It works by iterating ("marching") over a uniform grid of cubes superimposed over a region of the function f, i.e. the voxel array in this case. To this end, the vertices of every voxel are encoded as 0 or 1 depending on whether their signed distance is negative (inside the object/surface) or positive (outside the object/surface), as shown in **Figure 11**. In this way, if all eight vertices of the voxel are positive, or all eight vertices are negative, the voxel is entirely above or entirely below the surface, and no triangles are emitted. Otherwise, the voxel intersects the surface, and some triangles and vertices have to be generated. Since each vertex can either be positive or negative, there are technically 2^8 possible configurations, but many of these are equivalent to one another. The original paper distinguishes among fifteen unique cases, as shown in **Figure 12**.

The discrete set of different configurations permit using a *look-up table* (LUT) for easily identifying each case given the 8-bit code. Once located in the LUT, each vertex of the generated triangle can be placed on the appropriate position along the voxel edges by linearly interpolating the two distance values that are connected by that edge. Iterating over all voxels adds triangles to a triangle list, so that the final mesh is the union of all these triangles. The smaller the voxels, the smaller the mesh triangles will be, making the approximation more closely match the target function.

[3] $sdf_j(x) = depth_j(x) - cam_z(x)$ and $tsdf_j(x) = \max\left(-1, \min\left(1, \frac{sdf_j(x)}{\tau}\right)\right)$, where $depth_j(x)$ is the depth measured in between the camera and the nearest object surface point p on the viewing ray crossing x, $cam_z(x)$ is the distance in between the voxel and the camera along the optical axis, and $\pm\tau$ is the truncation value.

[4] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. on Conference on Computer Graphics and Interactive Techniques*, pp. 163–169, 1987.

Due to the existence of ambiguities in the trilinear interpolant behaviour in the cube faces and interior, the meshes resulting from the original algorithm present discontinuities and topological issues. The popularity of the MCA and its widespread adoption have resulted in several improvements to deal with the ambiguities and to correctly track the behaviour of the interpolant.

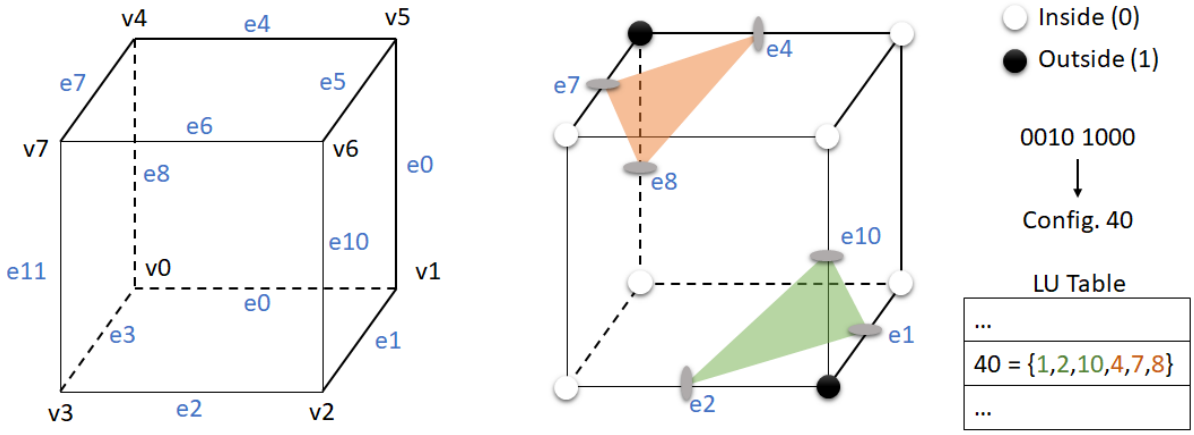


Figure 11: Illustration of the MCA

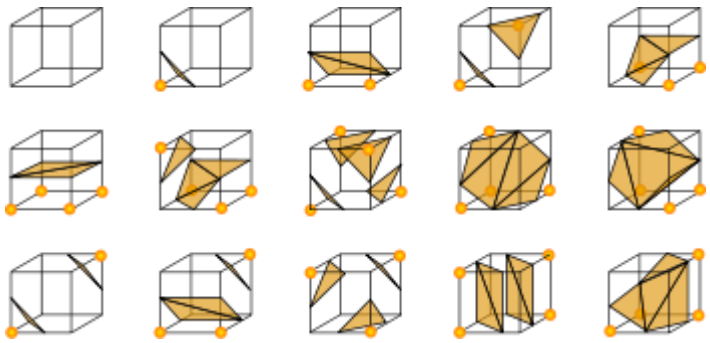


Figure 12: Fifteen cases considered by the original MCA

3. Mesh refinement

Although the mesh produced by the Voxblox-based software is of adequate quality for the intended application, there are a number of post-processing operations that can contribute to making it more suitable for the intended further use. The mesh refinement is accomplished through an additional stage the comprises the four steps depicted in **Figure 13**, which are briefly described next:

- First, the isolated mesh portions are deleted, reducing the memory storage required.
- Second, the possible holes that can appear during mesh generation are closed.
- Third, the mesh is smoothed to avoid the sawtooth effect.
- Finally, the mesh is colorized for enhanced visualization and to make easier the interpretation of the surface model.

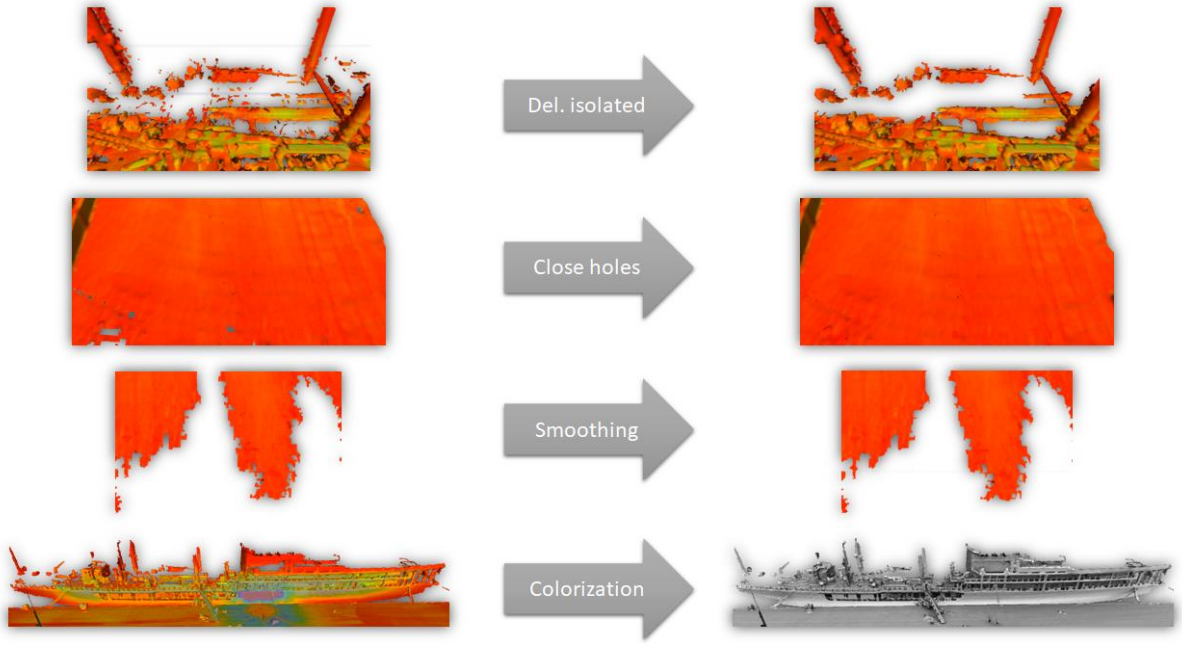


Figure 13: Steps of the model refinement stage

These four steps are automated in the form of a script that runs once the Voxblox-based stage has produced the initial mesh, so that the full process until obtaining a fully usable 3D model of the structure can run without human intervention. There are, however, a number of parameters that can be adjusted if needed to enhance the accuracy of the mesh generated or to tune the process in accordance with the complexity of the case, i.e. the surrounding environment and the specific vessel hull. These parameters and their effects are reviewed below:

- **Resolution.** This is the size of the voxels in the TSDF representation, which has an important impact in the mesh details, as can be observed in **Figure 14**. *Default value = 0.15 m.*
- **Maximum Range.** Points at a larger distance than the *maximum range* are discarded during mesh generation. This parameter defines the extent of the voxel array that represents the TSDF, and hence contributes to reduce the storage needs. This is particularly useful for narrow environments, where a shorter maximum range is recommended. *Default value = 50.0 m.*
- **Robot Radius.** This parameter is used to discard points that are returned by the sensor but belong to the structure of the robot. This avoids involving them in the generation of the mesh, and so prevents further distortions. *Default value = 0.55 m* (to be adjusted for the specific robotic platform involved).
- **Height Threshold (h_t)**. This is the height from which we consider the platform is ready for flying and therefore point cloud recording can start. This avoids some artifacts in the mesh related with the proximity of the ground. *Default value = 1.5 m.*

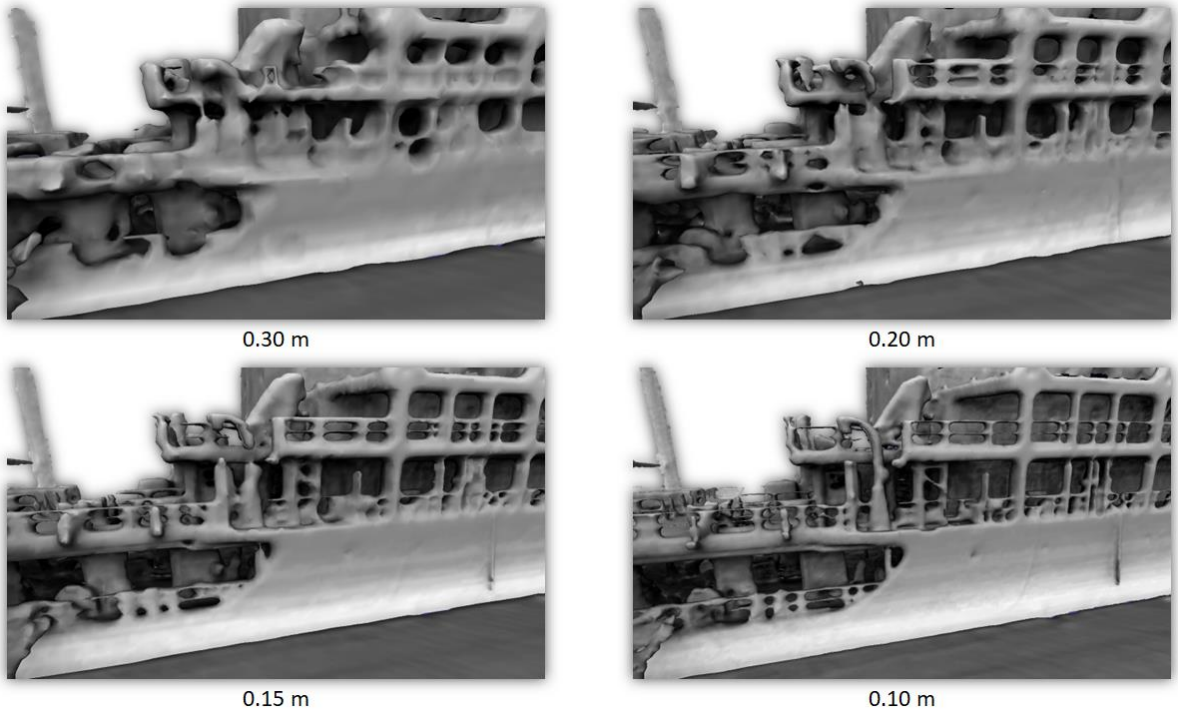


Figure 14: Final meshes at different resolution levels, leading to more or less detailed reconstructions

4. Reconstruction results

This section reports on the reconstruction results obtained during the BW2 field trials taking place in Lisbon, 11-16 September 2023. During the week, a number of field experiments were conducted, in which the support tanker vessel NRP Bérrio was involved. A picture of the vessel can be found in **Figure 15**, apart from the picture already shown in **Figure 8** to illustrate the quality of the map produced by LiODOM, as a representation of the area of operation.

Figure 15: NRP Bérrio.
Photo taken from: Link

Figure 16 to Figure 19 show final meshes of the Bérrio, at different voxel resolutions, respectively 0.30, 0.20, 0.15 and 0.10 m, for a flight covering the entire vessel, from end to end (other flights were also performed, partially covering the ship). As can be observed, the reconstructions achieved are quite accurate at a qualitative level, i.e. in terms of visual quality, in every case.

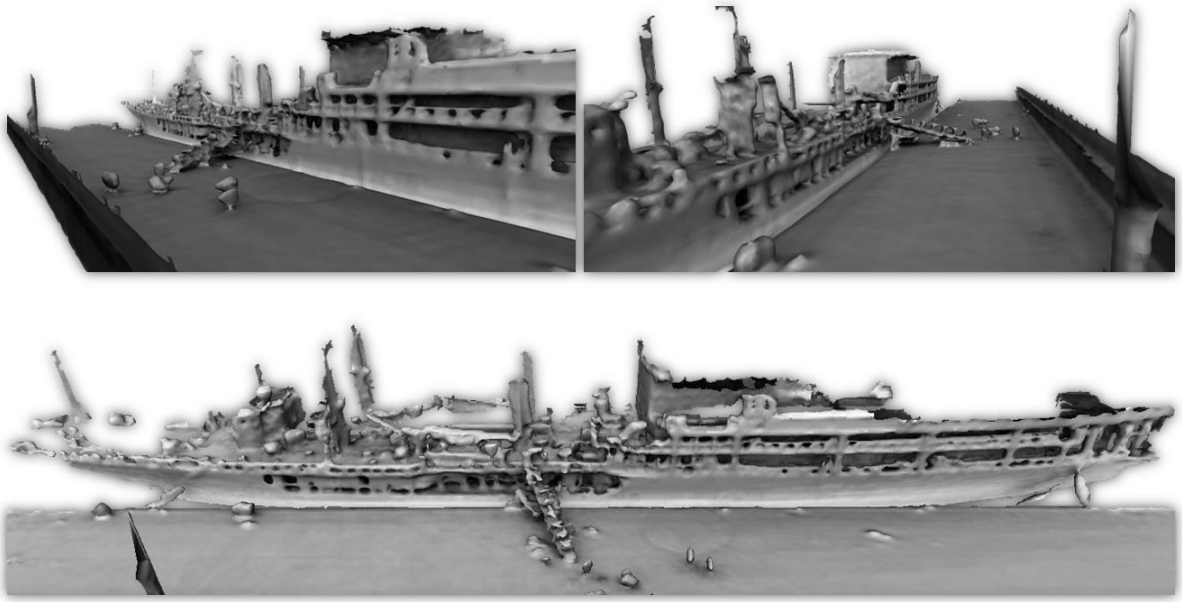


Figure 16: Reconstruction of the vessel with a resolution of 0.30 m

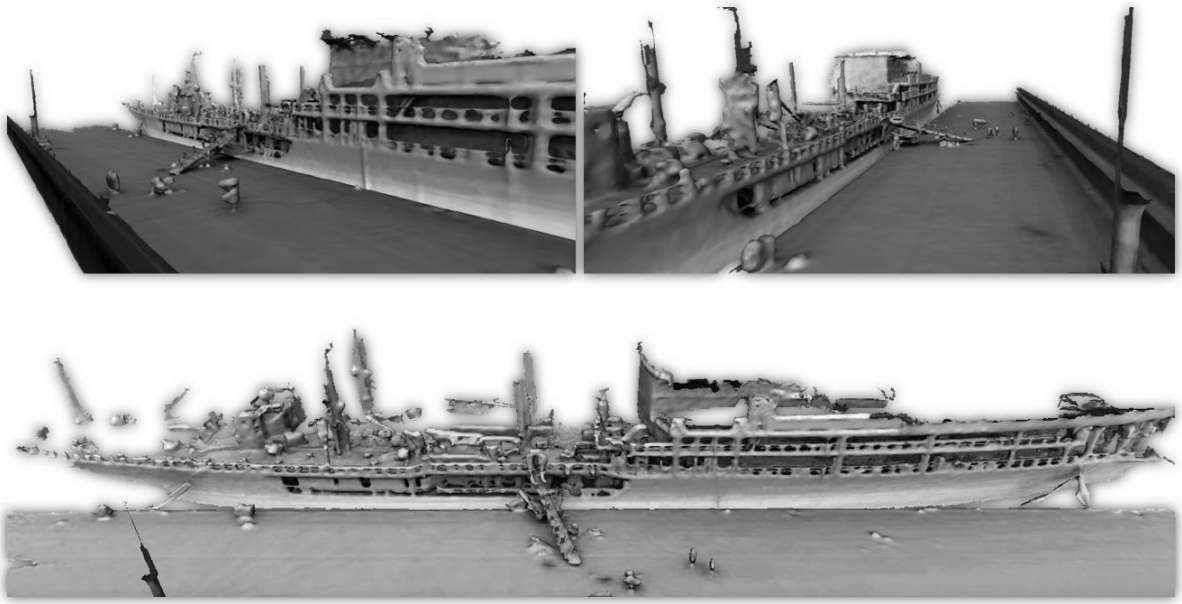


Figure 17: Reconstruction of the vessel with a resolution of 0.20 m

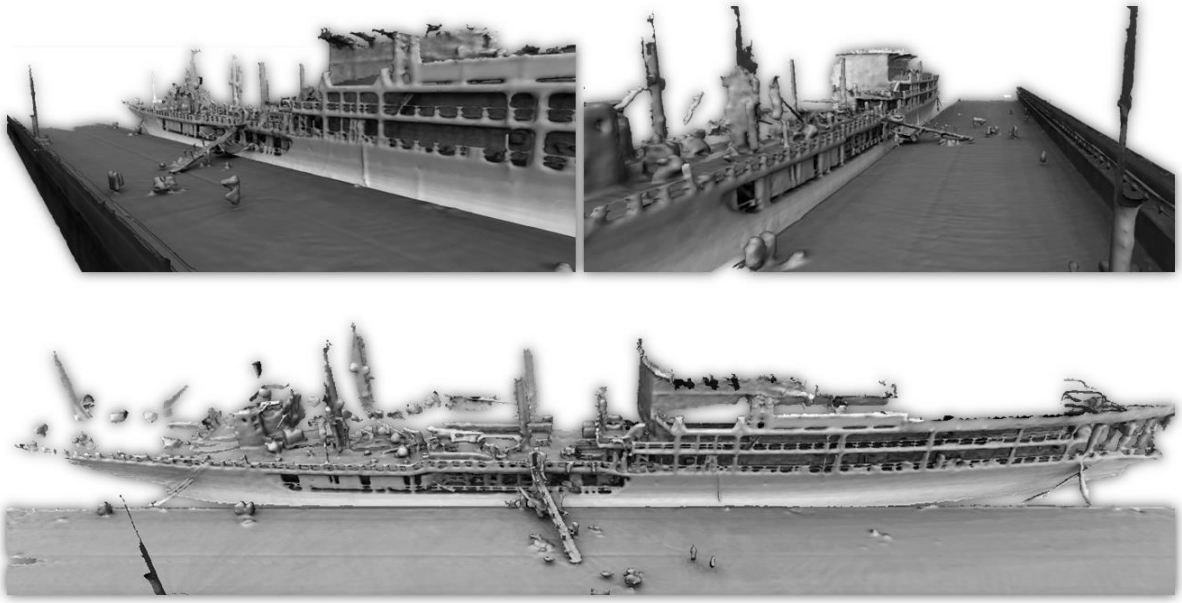


Figure 18: Reconstruction of the vessel with a resolution of 0.15 m

Figure 19: Reconstruction of the vessel with a resolution of 0.10 m

In order to validate the reconstructions at a quantitative level, we have employed a Leica Total Station (MS60, Robotic Total Station kit), which, through scanning of the vessel, has produced a high-resolution point cloud that has been taken as the ground truth for error quantification.

The comparison has been performed through the following two steps:

1. **Models Cleaning.** This step removes unnecessary elements that unavoidably appear in one of the models and not in the other, e.g. different segments of the ground, depending on where the flight has taken place or where the Leica TS has been placed.

2. **Models Alignment.** This step aligns both models making use of the ICP algorithm. Hence, the models are translated and rotated to match each other as accurately as possible.

By way of illustration, the Leica TS model and the full model at 0.10m resolution after cleaning are shown aligned in **Figure 20**.

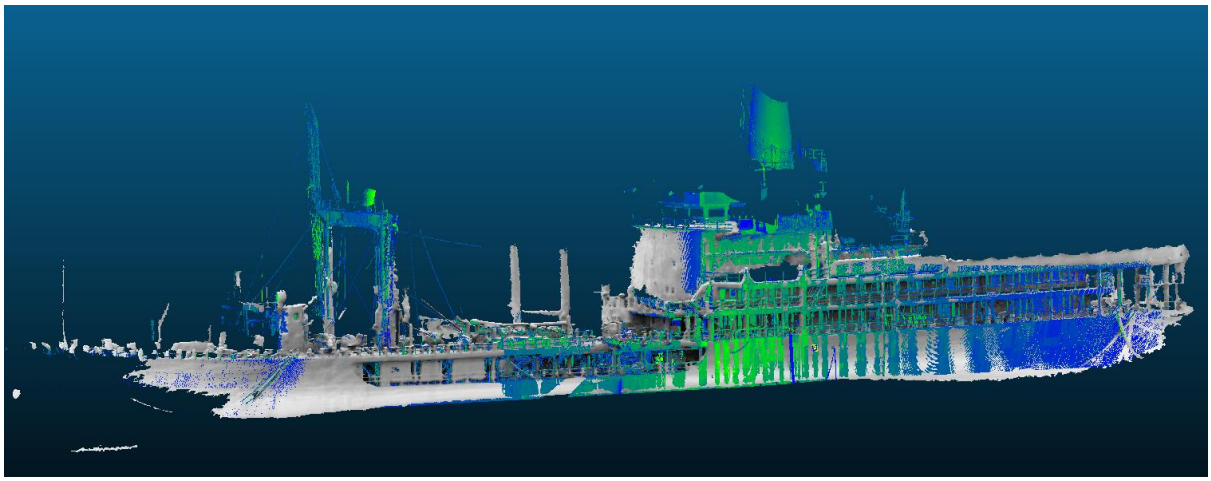


Figure 20: Leica TS point cloud (green/blue) and Voxblox-based mesh (grey) after alignment

The alignment permits calculating statistics of the reconstruction error. Outliers of the matching process are defined as points with an inter-models distance above a threshold. These errors are common in sections that appear in one of the models but not in the other, i.e. the underlying matchings are unreal matchings; see the ship chimney in the Leica TS model in **Figure 20**, which cannot be matched with any section of the Voxblox-based mesh since this part of the vessel was not included in the mesh as the MAV did not fly high enough. Results at 10 cm and 15 cm resolution, and thresholds on outliers from 50 cm to 5 m, are respectively provided in **Error! Reference source not found.** and **Error! Reference source not found.**.

Table 1: Results comparing the 0.10 m resolution mesh with the ground truth point cloud generated by the Leica TS

Threshold (m)	Average Distance (m)	Standard Deviation (m)
0.5	0.0453	0.1864
1.0	0.0543	0.3057
3.0	0.0688	0.6240
5.0	0.0892	0.8648

Table 2: Results comparing the 0.15 m resolution mesh with the ground truth point cloud generated by the Leica TS

Threshold (m)	Average Distance (m)	Standard Deviation (m)
0.5	0.0520	0.1886
1.0	0.0562	0.3092
3.0	0.0666	0.6313
5.0	0.0833	0.8676

As can be observed, the differences between the results for the two resolutions are quite reduced. The behaviour for the different thresholds on outliers that have been considered indicate errors between 5cm and 8-9 cm on average.

IV. Hull modelling underwater

Figure 21: Vehicle involved in hull modelling underwater

Hull modelling underwater is accomplished through the underwater vehicle that is shown in **Figure 21**. As for perception, this vehicle is equipped with two specific sensors: a monocular camera and a multibeam forward looking sonar, respectively supplying optical and acoustic data. Both sensors are horizontally aligned and point to the same direction though with a small, vertical offset between the two. The footprint of the sensors is displayed in **Figure 22** to facilitate the understanding of the coverage that can be attained by means of the sensors onboard.

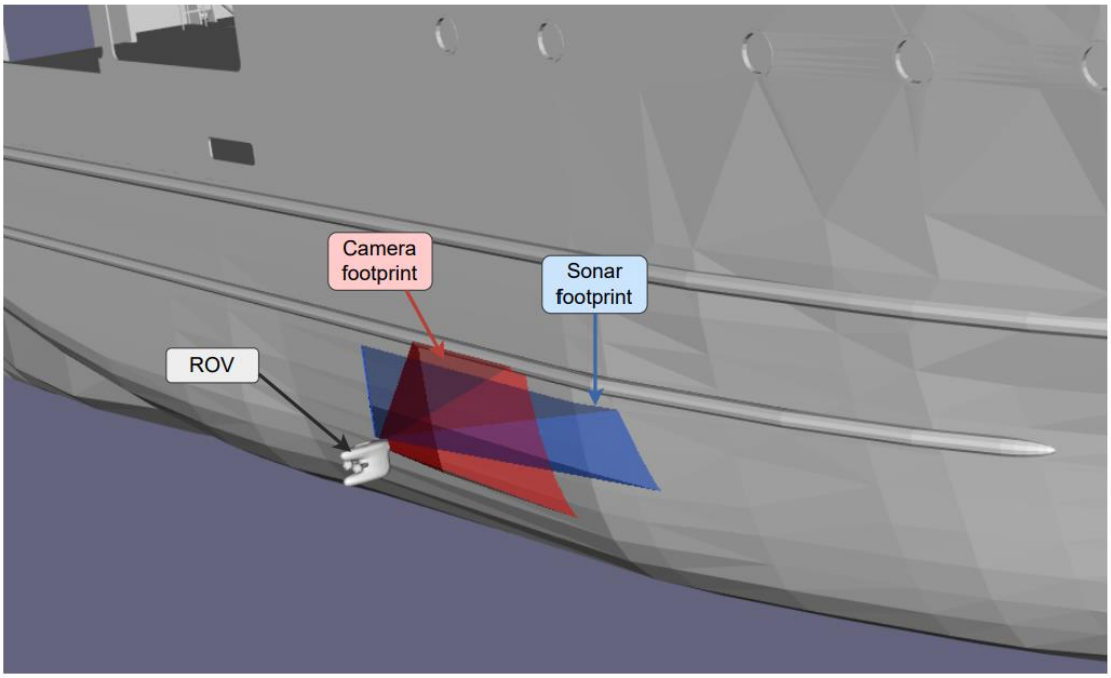


Figure 22: The footprints of the camera and sonar are represented, with the vehicle facing a ship hull

To enable real-time monitoring of the condition of the ship hull, a real-time data feedback stream is setup. The surveyor has access to an inspection map that is incrementally built as the operation goes on. It enables tracking of the inspection progress and of the areas that were inspected by the vehicle. It consists of a 3D acoustic model of the hull generated using the sonar measurements, down sampled to a 3D voxel grid.

Additionally, more accurate models are generated for the more complex and specific regions of the ship, also referred to as *Points of Interest* (PoIs). In this case, the sonar cannot be used as the main perception sensor since the assumption of a flat surface does not hold anymore. Therefore, to enable an efficient visual representation of the PoIs, the sonar is combined with a monocular camera.

1. Acoustic Inspection Map

The inspection map is generated from the forward-looking sonar sensor, using the assumption of the ship hull being locally flat. This means the closest acoustic feature for each beam on the sonar scan can be placed on the zero-elevation plane of the sonar, i.e., at the same depth as the vehicle and accounting for the vertical offset. During the inspection, the vehicle is typically at a distance of one meter to the hull, the maximum reprojection error on the zero-elevation plane becomes ~0.17m. Additionally, given the shape of the hull and its distance, the closest features are likely to be on the zero-elevation plane, making possible the assumption of the flat surface for the acoustic reconstruction.

The 3D map is built in multiple steps. First, the sonar scan is pre-processed to reduce the visible noise, and the closest feature is extracted for each beam. This has the advantage of limiting the data redundancy and the reprojection error. However, there can still remain residual noise in the data due to the environment and direct conversion into LIDAR-like data, creating outliers and biasing the map generation. This can be efficiently corrected by filtering the points using a virtual moving window averaging the contained points to verify the central one, biases related to the geometry of the scene are vanished. The remaining reliable

points are converted to 3D points and placed in the robot reference frame. A voxel is created only if it contains enough points according to the grid resolution, i.e. the number of voxels within a 1 m^3 volume. Since the inspection map is utilised for monitoring purposes, a grid with high resolution is not required but should be high enough to correctly represent the geometry of the hull. Over time, a dense point cloud is generated and converted into a complete inspection map that the inspector can refer to and the vehicle can use for scene understanding. Several examples are provided in **Figure 23** with the dense point clouds of structures and their corresponding inspection maps.

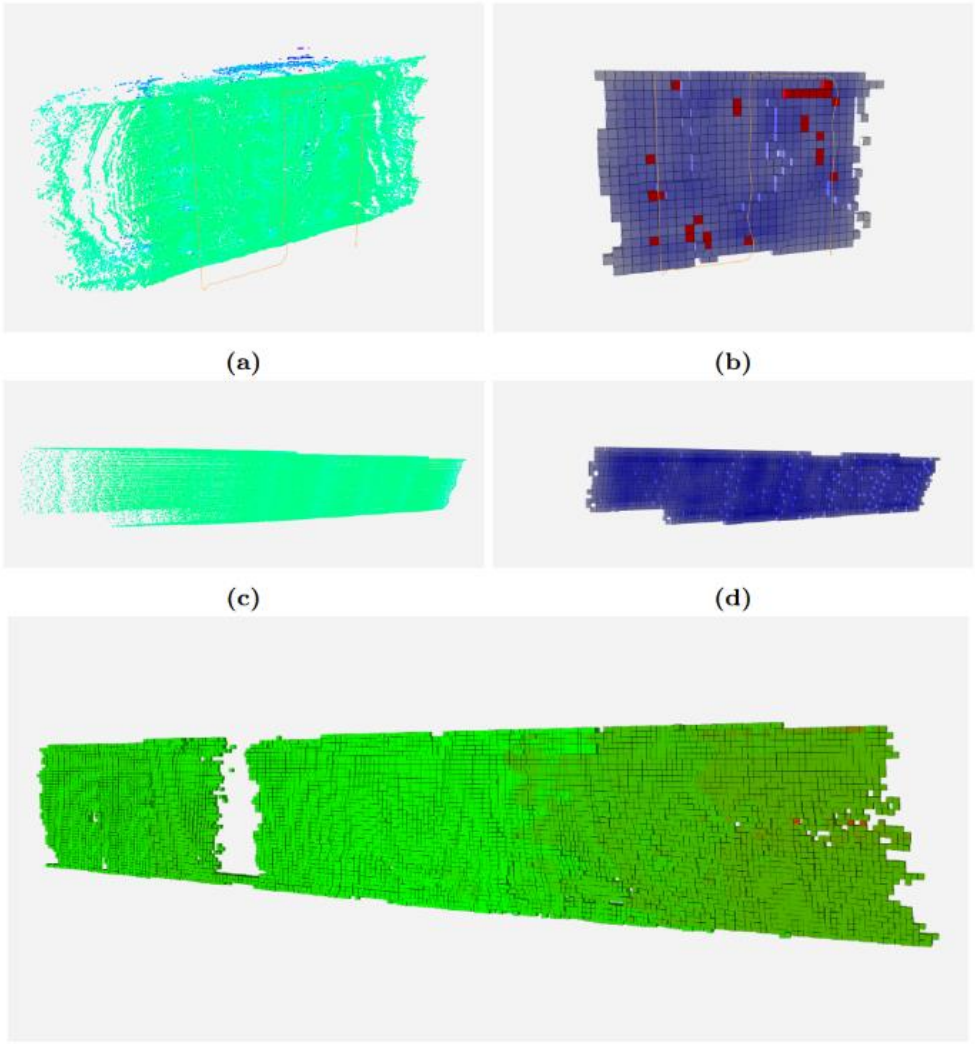


Figure 23: Occupancy maps from three inspection scenarios: (a) and (b) are respectively the dense point cloud and its equivalent voxel map of a harbour wall; (c), (d), and (e) represent sections of a ship hull

2. PoI Models

To efficiently model the PoIs, the sonar is combined with a monocular camera. The combination occurs in the SLAM framework ORB-SLAM3, where the visual odometry and point cloud are made more robust and correctly scaled. This is especially important for the robot situational awareness and for the inspector to assess the structural integrity of the vessel and/or potential deformations. The combination method of the two sensors relies on their positions, i.e. they are horizontally aligned with a vertical offset. This enables

an intuitive estimation of the intersection areas of the sonar's acoustic beams with the image plane. Therefore, each beam can be represented as a set of vertical segments on the image. They are mapped to image pixels by using the Pinhole model. For each beam, the closest detected feature with high intensity is selected; since both perception sensors are horizontally aligned, it is straightforward to obtain the horizontal pixels of a beam with a fixed azimuth. The sensor vertical offset and the acoustic distances to the objects in the observed scene must be considered to estimate the beam's corresponding vertical pixels in the image.

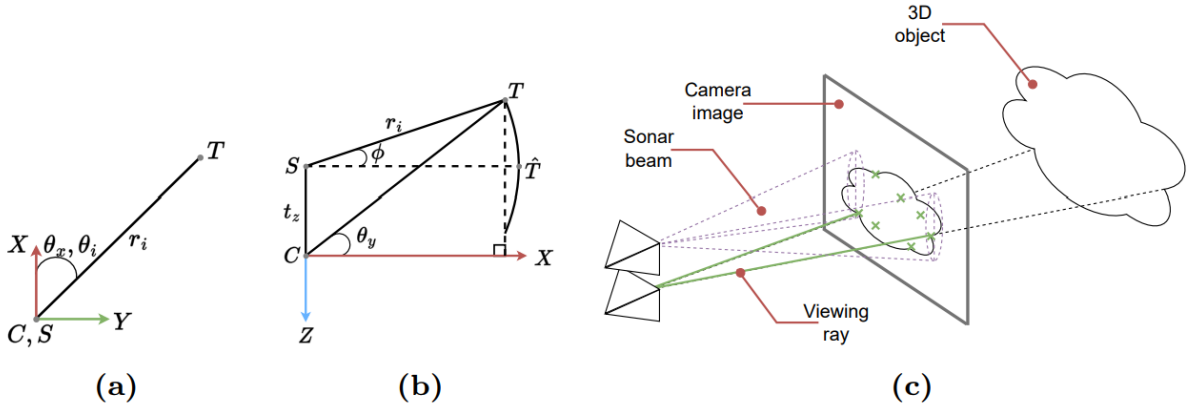


Figure 24: Illustration of the correspondence and matching mechanisms: (a) and (b) are respectively a top-down view and a side view representation of the geometry involved to obtain the pixel position on the camera image of a sonar feature T; (c) depicts a 3D scene with physical representations of the acoustic beams and the correspondences with the visual features

The working principle of the combination and matching method of the features from both sensors is displayed in **Figure 24**. The correspondence mapping and the estimation of the intersection regions between the beams and the image plane enable to constrain the search for good visual features to match. Since only the closest acoustic features are selected, the same is done with the visual features. Although the scenes are observed differently from both sensors, the detected relative distances remain the same, and therefore the closest acoustic and visual features should correspond. Based on this, the closest detected visual feature that has its 2D projection on the image within the intersection region is matched with the acoustic feature. This operation is repeated for each reliable acoustic feature.

Comparing each visual-acoustic match provides information about their scale and reliability. A set of scale ratios is obtained by dividing the acoustic distances by the visual distances. In this case, having a set instead of a single value enables robustness and stability in the determination of the final depth ratio. *Maximum Likelihood Estimation* (MLE) is continuously applied over all matches to obtain a unique and consistent ratio. For this, the set of distance ratios is assumed to follow a Gaussian probability distribution and its log-likelihood is maximised to obtain an optimised mean which corresponds to the depth ratio of the current scene. Additionally, the confidence intervals of the distribution are considered to detect and reject the outliers.

The final estimated trajectory from the SLAM framework has been compared to alternative setups such as a simple monocular SLAM, a Visual Inertial SLAM, and dead reckoning based on a DVL and an IMU. The ground truth was estimated using a mix of GPS fixes and visual landmarks. This enabled a performance comparison experiment and showed that the proposed solution outperformed the others.

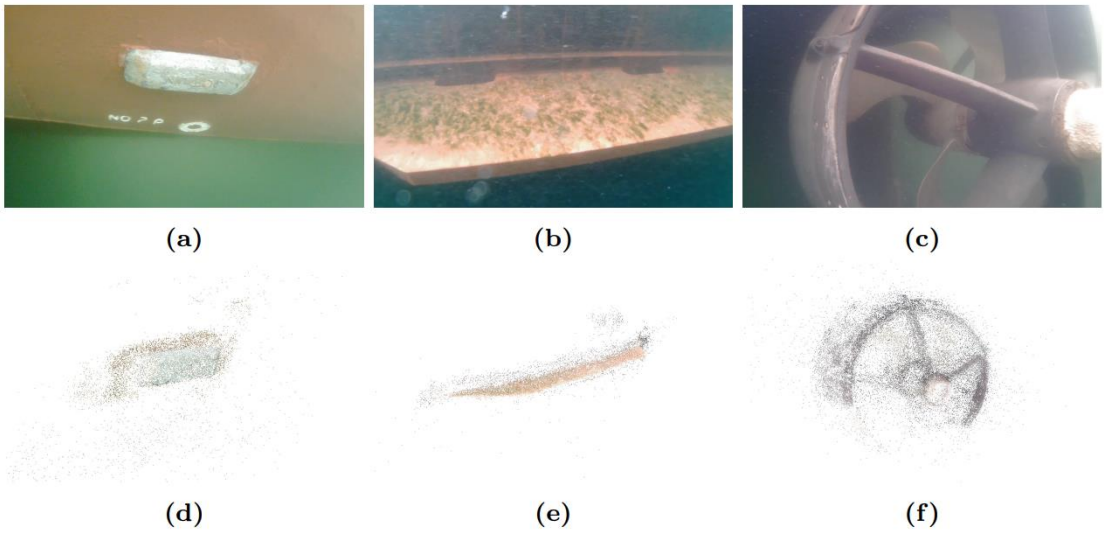


Figure 25: Point clouds for several PoIs generated during an inspection mission. The top row includes images extracted from the videos used to make the point clouds of the second row. (a) and (d) show an anode, (b) and (e) show a bildge keel, and (c) and (f) show a propeller

In practice, during the inspection missions, the updated SLAM framework enabled a better mapping of ship parts such as the propellers and bildge keels. A sample of the point clouds generated during some of the inspection operations is displayed in **Figure 25**.

Finally, the results are refined either in post-processing or during inspection using external devices. This enables the generation of 3D texture models of the inspected PoIs. They are then included in reports along with the mission metadata to comply with the Class Societies' rules and regulations. Example of such reports are shown in **Figure 26**.

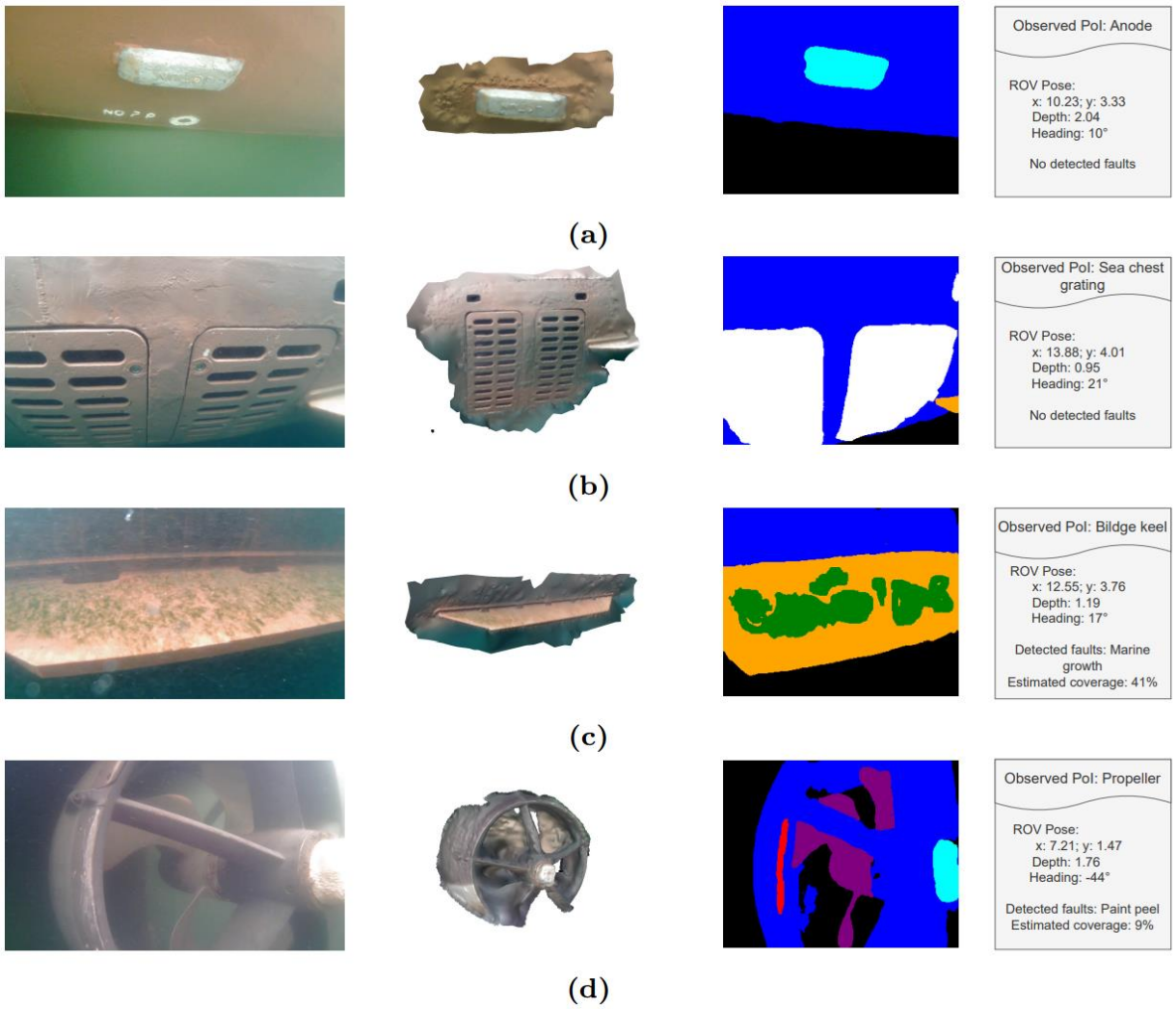


Figure 26: Examples of data that can be generated when a PoI is detected

The first column includes the representative images while the second shows the generated 3D textured model of the captured PoI. The third column displays the semantic segmentation masks that are used to detect the PoIs and faults. Finally, the last column reports on the pose of the vehicle and the inspection findings. In (a), an anode is detected, a sea chest grating is shown in (b), a bildge keel in (c), and a propeller in (d).

V. Model generation by scanning (Leica TS)

This section report on the development required to integrate the Leica Total Station in the 3D modelling of the infrastructure monitored within the BW2 project. Specifically, the Leica Total Station is a very precise instrument capable of scanning a structure from multiple viewpoints to acquire a 3D point cloud and record the position of point of interest. Nevertheless, the Total Station (TS) is not integrated into ROS and does not provide an API to process the proprietary files it generates. As such, a significant software development was required to post-process scanning results.

Because the total station file format is not documented, a reverse engineering effort was required to use it in the BW2 project. This section documents the file format observed while operating the TS M60 in the BW2 project. Two specific modes of operation were identified, either the M60 provides a scan with colour

associated with 3D points, or it provides only the 3D points. We are considering here the XML export from the Leica Total Station. This export consists of one XML file describing the operation of the Total Station, and a number of SDB files referred to by the XML file.

Every SDB file is made of a header of 520 bytes, followed by a sequence of point descriptions. Depending on the presence of color, every point is described by 20 or 23 bytes organized as follows:

Table 3: Data structure for the proprietary format of Leica files

Bytes	Semantic
0-3	X coordinate (32bit float)
4-7	Y coordinate (32bit float)
8-11	Z coordinate (32bit float)
12-13	Intensity return (16bit unsigned)
14-15	Signal to noise ratio (16bit unsigned)
16-17	Horizontal position of the point (16bit unsigned)
18-19	Vertical position of the point (16bit unsigned)
20-23	[R,G,B] colour, 8bit unsigned (if available)

This representation is particularly useful because the horizontal and vertical index of the points (bytes 16-19) provide the neighborhood of the scan points, from which the 3D structure and a mesh can be inferred by just connecting neighbour points, at least for a quick visualization of the recovered mesh.

Note that, every scan collected within one total station setup is saved as an independent SDB file. Nevertheless, the 3D coordinates of the points are all expressed in the 3D frame of the global setup. After parsing, the 3D points are published as ROS PointCloud2 that can be processed with standard tools used in the robotic community.

In addition to the SDB files, the Leica Total Station also provides an XML file that describes all the surveyed points. Parsing this file does not present any significant challenge. In our current application, all the surveyed points are published in the ROS TF framework so as to simplify the alignment of the 3D data with the 3D models produced by LiODOM.

Example of collected data

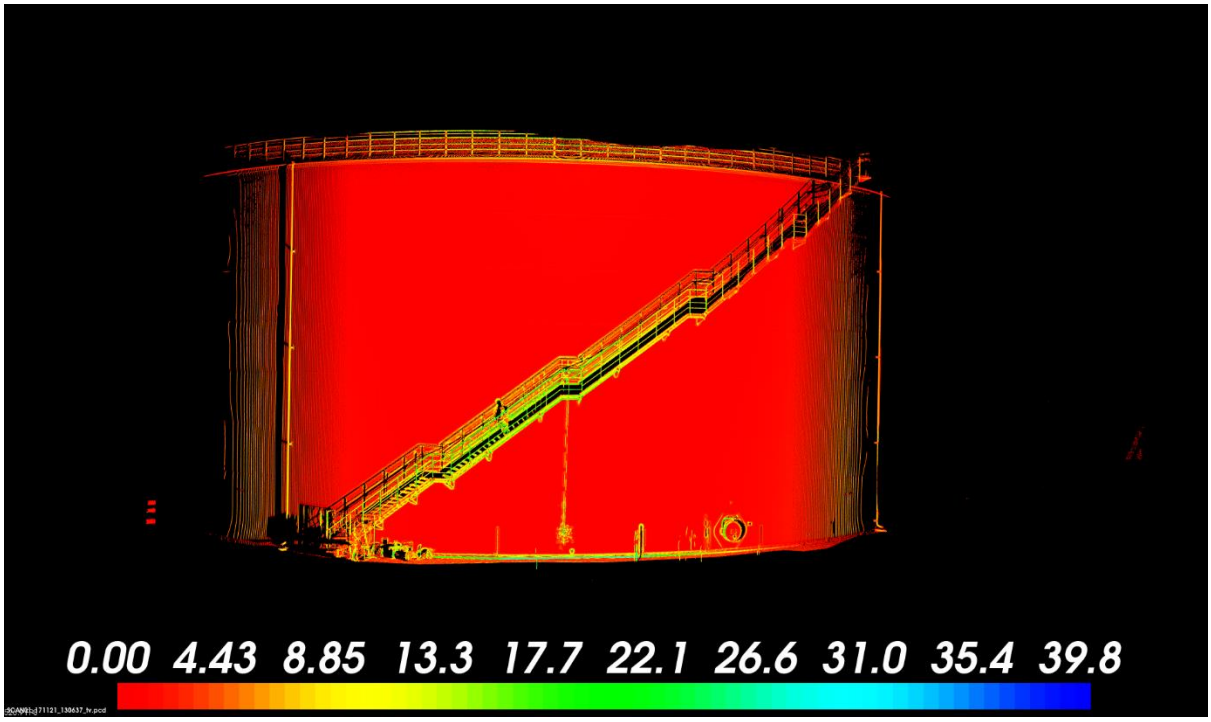


Figure 27: Storage Tank in Bazancourt, France, with color scale representing the signal to noise ratio

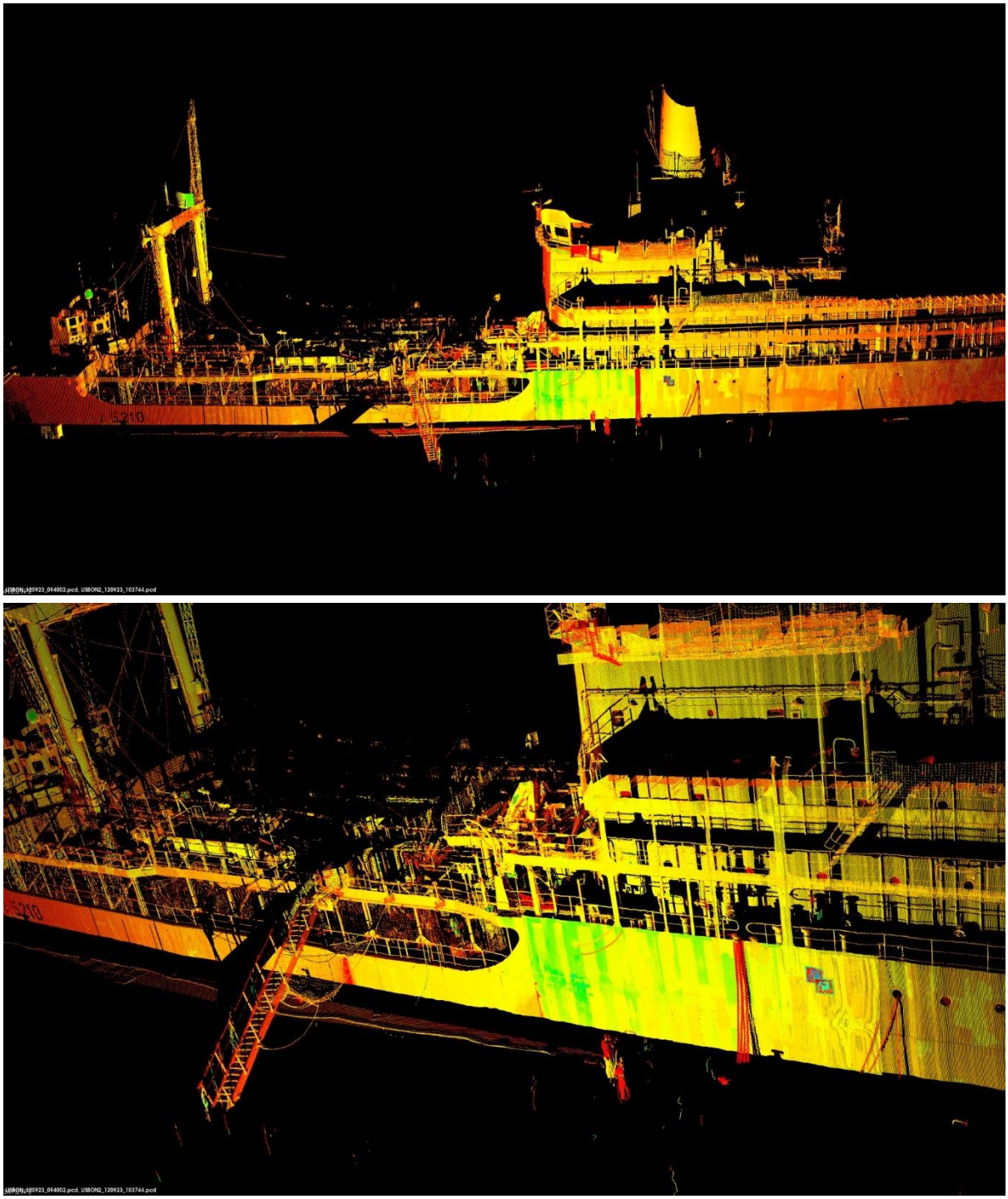


Figure 28: 3D model of the NRP Bérrio, Lisbon, Portugal

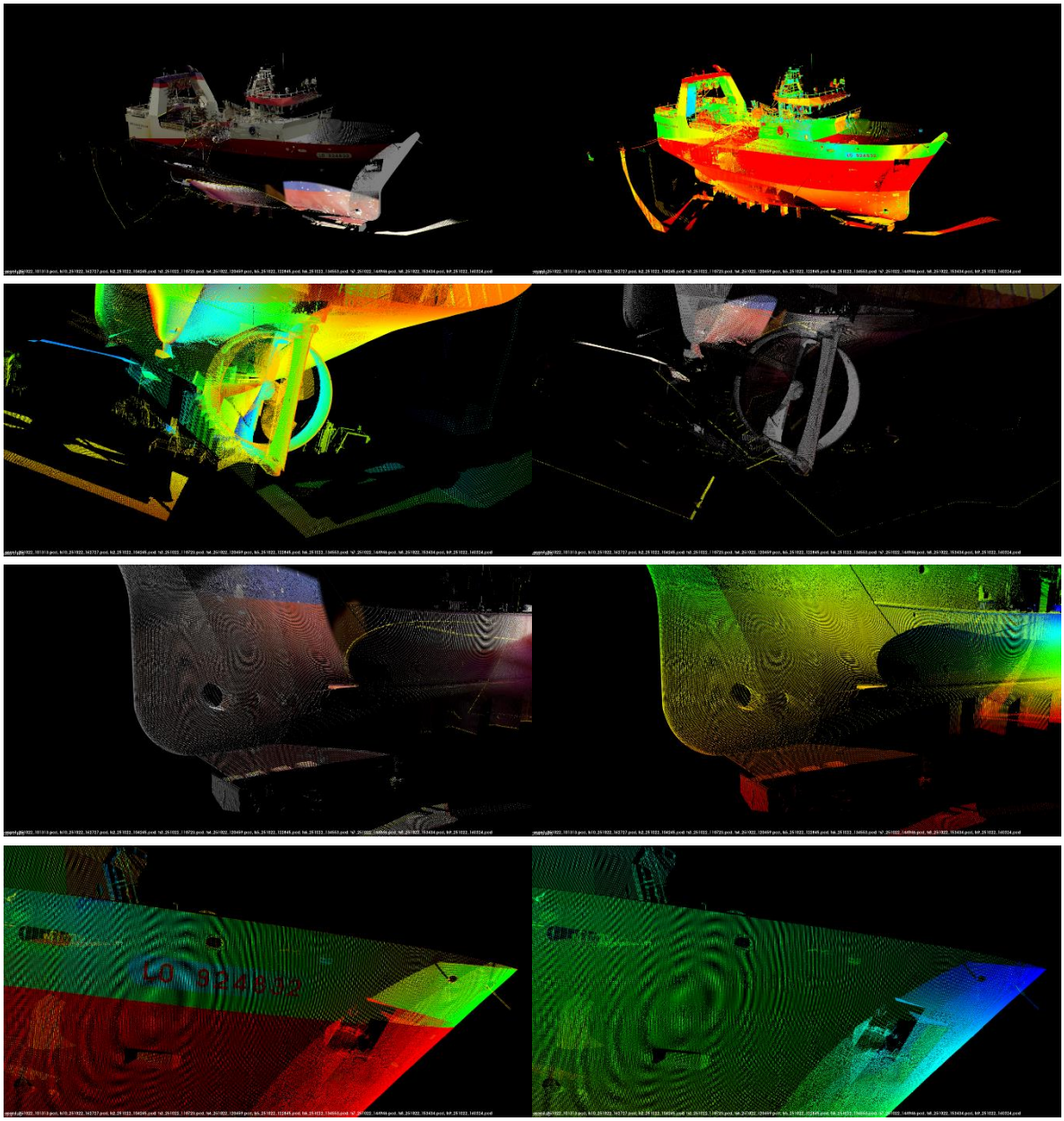


Figure 29: Various rendering of the scan of the JC Coulomb

VI. Conclusion

Deliverable D4.3 summarizes the developments of the BW2 consortium regarding the modelling of the structure/surfaces under inspection, as part of the BW2 inspection framework. The report has briefly described the frame alignment issues that have been necessary to enable the synchronization of the different platforms so that the data captured, and perhaps processed, by one platform can be employed by another. Next, D4.3 has addressed hull modelling considering the two cases, above-water and underwater, since different robotic platforms are necessarily involved. The process involved have been described and the results obtained in recent field trials have been reported. Finally, this deliverable also reported how the Leica Total Station was integrated into the project and how its data could be exported towards opensource file formats.