Autonomous Robotic Inspection and Maintenance on Ship Hulls and Storage Tanks

Deliverable report – D8.3

Context	
Deliverable title	Augmented reality for assisted maintenance
Lead beneficiary	RWTH
Author(s)	Torsten Kuhlen, Simon Oehrl, Sebastian Pape
Work Package	WP08
Deliverable due date	September 2023 (M45)
Document status	
Version No.	1.2
Type	REPORT
Dissemination level	**Confidential**, only for members of the consortium (including the Commission Services
Last modified	26 March 2024
Status	RELEASED
Date approved	26 March 2024
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

ABBREVIATIONS

AR	Augmented Reality
VR	Virtual Reality

REFERENCED DOCUMENTS

1. BugWright2 Grant Agreement (GA) Number 871260
2. D7.2 Virtual Reality for Real-Time Mission Monitoring
3. D7.3 User Interface for Inspection Mission Planning

These documents are stored on the file sharing site hosted by CNRS.

HISTORY OF CHANGES

Date	Written by	Description of change	Approver	Version No.
16.10.23	Sebastian Pape	Initial Writing		1.0
20.03.24	Sebastian Pape	Rewrite due to break-through in the integration week in Athens ‘24		1.1
22.03.24	Simon Oehrl	Minor Changes		1.2

Executive summary

This document serves as a reference for the design and technical challenges of an assisted maintenance system using augmented reality. Due to significant progress made in the last integration week in Athens in March 2024 and this document was reworked to include the improvements upon our initial results.

In the following, we will first talk about our used hardware setup and world alignment system in the technical setup part. Then, we go shortly into implementation details. Finally, we go over the evaluation of this system and give a short conclusion and outlook for the future of systems like this.

I. Technical Setup

1. Augmented Reality Hardware

In the beginning of the BUGWRIGHT2 project the augmented reality (AR) hardware sector was evaluated by RWTH and the decision for a HoloLens 2 by Microsoft was made, as it was one of the very few options at the time and all other options seemed more experimental and less reliable at that time. Additionally, the HoloLens 2 provides support for the Unreal Engine 4, which was used for work package 7 at the time. As the user interfaces were expected to be similar a device with strong support for Unreal Engine 4 was needed. Later in the project runtime, WP7 switched the user interface implementation to a React/WebXR basis, which was also supported by the HoloLens 2.

The HoloLens 2 features reliable self-tracking, a good wireless data connection, a Windows operating system and relatively bright projection screens.

While on paper this hardware setup seemed perfect for this application, during the projects runtime we stumbled upon multiple challenges with the hardware limitations of the device:

The screens of the HoloLens2 are bright in indoor settings but are not bright enough in outdoor settings to make out sufficient detail for actual maintenance support. In our indoor testing, the user interface could clearly be made out, but in actual test-cases that we performed in Bazancourt (France) and in other integration weeks in brightly lit outside conditions, the display brightness is insufficient. This inhibits visibility of the user interface and its interaction in bright sunlight.

Another challenge was the limited compute power of the HoloLens 2. While the amount of data visualized in our AR system is generally quite low, it had implications on the debugging steps. E.g., generally, the surface mesh of the vessel does not need to be rendered in AR as the real vessel in direct sight. This drastically reduces the rendering performance and loading times as the generated mesh tends to be quite large. However, rendering the mesh is still very useful for checking the alignment process described in I.2.

While the self-tracking of the HoloLens 2 works very well and was deemed stable in our testing, an initial alignment of the “origin” of the self-tracking with a known position in the real world needs to be established to overlay the virtual augmentations over the real world in the right positions. This alignment functionality is not built into the device and needed to be implemented manually as described in the following section.

2. World Alignment System

The major challenge in using AR for maintenance purposes is the alignment of the data displayed within the HoloLens 2 to the real world. The HoloLens 2 displays its data within a virtual world which needs to be correctly aligned to the real world to ensure that the data is visualized at the correct position. Thus, an incorrect alignment leads to wrongly positioned data and confusion of the user during the maintenance.

In WebXR, the HoloLens 2 uses its starting position and orientation as its tracking origin while all the data is anchored in a general coordinate system developed in WP4. To derive a transformation between the these two coordinate systems we attempted the usage of fiducial markers to track multiple positions on the vessel and in turn derive the position of the HoloLens 2 in a known tracking frame from there.

Initially, we experimented with AR-toolkit markers, as AR-toolkit works out of the box within WebXR. For this CNRS provided us with small markers printed flexible magnetic backing sheet that could be stuck to any ferro-magnetic surface. We soon switched the tracking library to OpenCV with ArUco markers, as AR-toolkit support was very limited and very similar markers with magnetic backing already existed for other partners in the project. As the other partners also needed the exact positions of these markers for their tracking systems, we could leverage their setup and measurements.

The OpenCV library provides a very rich feature set for marker tracking, especially for the ArUco markers used in the project. Unfortunately, OpenCV is not directly available for usage in JavaScript/Typescript environments, thus we had to compile a custom version to web-assembly to use it in our React/WebXR environment.

While the implementation of the marker tracking is relatively easy with the backing of the OpenCV library, it still needs a quite involved process for calibration of all camera parameters. For this custom software was written, which is described in II.2.

With a working tracking system, the resulting position of the marker in relation to the build-in webcam is relatively stable, but still needs alignment with the origin of the device to factor out the position of the webcam within the device. As these numbers are not provided by Microsoft directly, they were determined manually.

All implementations and the encountered challenges for this marker tracking will probably be replaced in future generations of augmented reality hardware and software, as (ArUco) marker tracking gets more popular and a proposal for WebXR tracking[1] is in development.

[1] *https://immersive-web.github.io/marker-tracking/*

II. Implementation Details

Figure 1: Example ArUco marker (left) and some of the markers mounted on a ship during an integration week in Lisbon '23 (middle). The image on the right shows a tracked overlay over the real marker shown in the HoloLens (note: the offset occurs due to the screenshot process and is not visible in the device itself).

1. Theoretical Positional Initialization

As described above, a strong need for the alignment of the virtual (augmented) world and the real world needs to be established for proper interaction and data visualization. As this alignment needs to be established on every usage of the AR hardware it should be executed automatically in the background. To align the virtual world with the real world some correspondence needs to be found by the device.

For this, an external tracking system could be used, but this approach was deemed impractical for actual usage by clients on the scale of a huge ship. Thus, we opted for an alignment process based on computer vision and marker tracking. For this setup some kind of marker, e.g. ArUco markers (see Figure 1) are attached to the vessel and their positions are measured in relation to the vessel by other project partners.

With the known positions of the markers in relation to the vessel, the position of the augmented reality device can be derived by traversing a transformation tree that looks as follows:

→ root
  └ vessel mesh M_{vessel}
    └ marker positions P_{Marker_i}
  └ virtual origin of the augmented reality device O_{AR}
    └ physical webcam position P_{Webcam}

Here the root of the tree can be chosen arbitrarily and will be used as the root of the visualization scene later.

To initialize the device position the virtual origin O_{AR} shall be chosen such that $O_{AR} * P_{Webcam} * T_{Marker_i} = \mathrm{M}_{\mathrm{vessel}} * P_{Marker_i}$ for Marker i. If multiple markers are tracked simultaneously the usage of a least square optimization to choose O_{AR} with minimal error can help reduce errors. In our testing in the last integration week in Athens 2024, the usage of a single sufficiently large marker yielded good enough alignment already.

Once this correspondence is established all data received from the robots can easily be transformed into the frame used by the AR hardware and displayed accordingly.

In our testing we interactively determined P_{Webcam} to most accurately fit our HoloLens hardware and testing setup. During the real-world experiments on a large vessel we noticed that this alignment is very critical at larger distances as only a few degrees of tilt in the webcam alignment lead to several meters of error over longer distances.

2. Webcam based Marker Tracking

Tracking markers from webcam images can easily be implemented based on the existing open-source computer vision framework OpenCV2. As it is not easily available for usage in JavaScript with ArUco tracking support at the time of writing, a custom version needs to be compiled to web assembly with *Emscripten*. This step will become obsolete in the future, as OpenCV strives to integrate an official *JavaScript* build again.

For OpenCV to process the marker images correctly the camera needs to be calibrated first. For this, some images need to be taken with the camera and processed with OpenCV. Here it is important to note, that the images should be taken with the same device API that is later used, as e.g. on the HoloLens 2 some APIs provide images that are pre-distorted to remove lens distortion and other don't. To avoid incorrect calibration parameters, we implemented a simple website that can be visited with the HoloLens 2 that takes picture with the onboard webcam and outputs the processed camera calibration parameters. For this, the user can interactively take pictures of a calibration pattern, visible in **Figure 2**, while wearing the device.

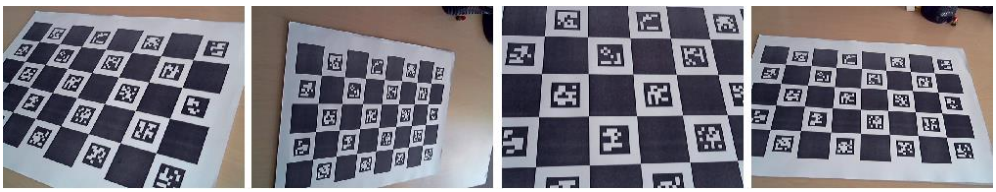

Figure 2: Example images of a ChArUco board, taken from docs.opencv.org

After the one-time calibration is established, all images taken with the webcam are undistorted using these parameters and the marker detection is run on the image. This process outputs the corner points of each marker within the image. From these corner points and a known size of the marker, the position and orientation of the marker T_{Marker_i} can be estimated (See **Figure 1**).

As this process is quite compute intensive and the HoloLens 2 still needs to render at interactive framerates this detection process should be run asynchronously in a background thread. This process can be run with about 1-2 frames per second on the HoloLens 2. While this seems slow, theoretically only one image is needed for the whole alignment process. From our practical testing, we would still advise to run this alignment constantly in the background to counter any drift of the devices internal tracking, movement of the vessel or some of the following inaccuracies:

[2] *https://opencv.org/*

- The inaccuracies for the actual position of the webcam P_{Webcam} in the hardware setup can easily lead to a big offset in the far distance.
- The provided marker positions P_{Marker_i} also carry some error. While position can be measured quite accurately with the laser scanner used in the project, slight variances in, e.g., the tilt of the markers lead to several meters of offset over far distances. While this can be countered with bigger markers and constant readjustment of O_{AR} the alignment in the real world will always carry a small error.
- Variations in shading due to sun-angle can disturb the tracking of the markers completely, making them impossible to track. In our testing setup tracking a 10*10cm marker over 5m was easily possible. In real world setups we could track some large markers (50cm) over 50m, others could not be recognized correctly over even small distances like 10m. In the last integration week, some even larger markers (77cm) were used by other project partners, leading to high accuracy in the used light-controlled indoor environment.

3. User Interface

The user interface was designed and implemented in conjunction with WP7.2 and WP7.3. For all details on this design, implementation and evaluation please refer to D7.2 and D7.3.

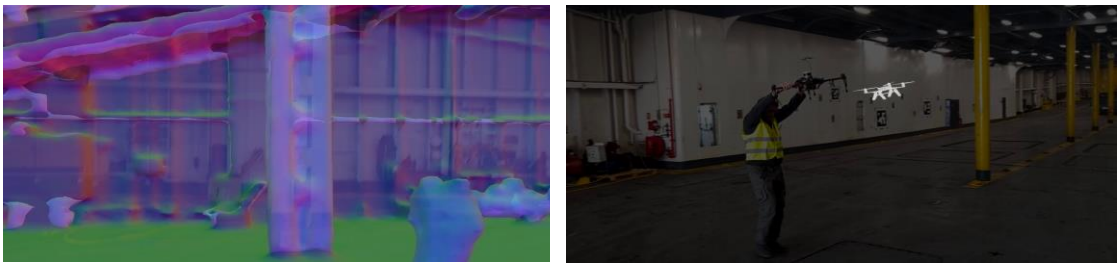

Figure 3: The aligned overlay of the mesh (left image) over the real world shows a very good alignment but was deemed not helpful during the inspection. In the right image, the same environment is shown without the mesh, but with live data tracking from an aerial drone platform. The estimated position is shown the side the real drone.

The AR user interface uses all the same data retrieval mechanisms and provides the same rendering capabilities as the virtual reality (VR) interface. The only difference between both interfaces is the rendering of the vessel, which is omitted in the AR interface, as the ship can be seen by the user anyway and an additional rendering was deemed confusing in our testing (see **Figure 3** left). Showing the position of the robot estimations not only helped the development of the systems, but also allows the direct data overlay in the real world (see **Figure 3** right).

III. Evaluation

The evaluation of the AR setup was not conducted with experts using the actual AR hardware during the project's runtime, as the tracking and alignment process was deemed too instable for any evaluation until the last integration week in Athens in 2024. As a theoretical testing of the setup in an indoor setting before would yield unusable results, due to the interface not actually overlaying useful data in that setting, we refer to D7.3 and D7.4 for the evaluation of the virtual reality user interface.

In the integration week in Athens 2024, the used environment yielded unexpectedly good tracking and alignment results (due to being mostly indoors) and was deemed helpful by many other partners. Unfortunately, the time to execute a formal study was too short to gather formal insights on that occasion.

IV. Conclusion

While we could see that the augmented reality user interface offers a big benefit for maintenance, the technology is currently not ready for usage in bright (outside) environments.

In the future, many current hardware developments will lead to higher contrast displays, that can project brighter and more readable images for the users, even in overhead sunlight. Together with hardware improvements for the performance in marker tracking we can see a successful application of the technology for the here presented use case.

For future developments, other sources of tracking could be explored that are currently not applicable for our developments. E.g., high precision GPS tracking or other external tracking systems could also be used for the alignment but were not applicable within this project due to incompatibility with the chosen hardware or availability within the testing areas.

V. Appendix

During the last integration week in Athens ’24, we collected some video material which is attached to this deliverable. It first shows the world alignment process, then the live data visualization and in the end the recorded data visualization. Note: An offset is especially visible in the yellow/red overlays over the marker. This is due to the video recording in the HoloLens 2 and is not perceivable, while using the device.

The video can be found here:

https://www.bugwright2.eu/nextcloud/index.php/s/kp7sqqrcQ6NxsnL