Semi-autonomous UVMS simulation: vehicle and manipulator pose control using artificial markers for localization

Yuri M. A. Oliveira¹, Diogo A. Martins², Leizer Schnitman³, Marco A. Reis⁴

Abstract—This work describes the developing and simulation of semi-autonomous Underwater Vehicle Manipulator System (UVMS) operation. It is the sequel from our previous paper on Remote underwater Operation Vehicle (ROV) pose control [1]. We designed a methodology that helps UVMS operators to perform its tasks. This method locates underwater structures and pre-allocate the UVMS itself and its manipulator in a pose that facilitates the operation. Every sensor and actuator used are already available in commercial UVMS. We tested our development in a virtual 3D simulation using ROS and Gazebo. We utilized Musa Manhães [2] environment and BIR* underwater structure from our previous work. The simulation presented good results with small localization and positioning error, despite underwater environment issues.

Key-words: UVMS, ROV, manipulator, control, simulation.

I. INTRODUCTION

Humanity still relies on Oil & Gas industries to produce energy. A research in 2019 showed that about 26% of consumed energy came from the Oil & Gas industries [3]. Most of Oil & Gas extraction happens onshore, but deep offshore extraction levels are rising faster [4]. Offshore operations need a floating station to store the collected fuel. This fuel flows from the well to the floating station through a vertical pipe, with its length varying from few hundreds to few thousands of meters. It is impossible for divers to go deeper than 300m, also this job is very dangerous even for smaller depths. That is why offshore exploration needs ROV to perform some operations, such as installing underwater components, oil flow controlling and retrieving dropped objects [5]. The importance of Oil and Gas in energy production, deep water extraction growth and also ROV increasing demand [6] were our motivation to work on the field.

This work uses an intervention ROV, also called work class or class III ROV. It is an ROV that has one or two manipulators attached to it and is also capable of carrying more weight than class I and II [5]. Class III ROV is a subset from UVMS. This acronym will be used in this work to refer to an ROV in conjunction with a manipulator.

Normally, UVMS have 2 manipulators. One anchor manipulator that holds UVMS still, as an underwater current countermeasure. Another main manipulator, as the actual operation manipulator. An umbilical cable connects the UVMS to the auxiliary vessel, in which operators stay. Also, every UVMS uses cameras and lights, so operators can see underwater. All of these components can be seen in the SF-30K ROV from Fig. 1. That is also the ROV used in the simulation.

Industrial functional UVMS still uses operators to execute underwater tasks. UVMS operators need to work in a very stressful environment and the success of the operation highly depends on their skill and concentration. Despite the lack of underwater operation repeatability, humans are still responsible for controlling intervention UVMS due to underwater robot complexity and high cost involved in the operation. The difficulty of controlling these systems is increased by some factors, such as water current and water turbidity.

For these reasons, we will develop in this work an software that would help UVMS operators in their tasks, to increase repeatability and reduce operation time. In this way, we intend to develop a semi-autonomous assistant for underwater operations with a ROV. That assistant is a software that should locate the underwater structure, move UVMS towards the desired operation pose and approach its manipulator to a valve or some other feature that is going to be activated. In that manner, the need of an anchor manipulator would be removed. Additionally, we just use cameras to locate the UVMS, so there is no need to add sensors in the robot. Therefore, underwater structures geometry must be previously known by the robot.

To test the software we are going to utilize ROS and Gazebo. ROS is a robot framework that allows connecting multiple programs on simulation. Additionally, it has lots of tools that help development and debugging [8]. Gazebo is a robot simulator that enables realistic virtual 3D tests
in the computer [9]. This is essential to validate software logic virtually without additional costs and risks from the real operation.

The simulation setup is an underwater environment developed by Musa Manhães [2] [10]. This environment has a ROV called RexROV, which was used in conjunction with Oberon7 manipulator [11]. RexROV dynamic model is based on SF-30K from Fig. 1 and Oberon7 manipulator is based on Orion 7P from Schilling robotics. Additionally, a panel developed by BIR* was added. This panel has a valve, a button and a hole. These features illustrate similar ones that UVMS needs to activate. Artificial markers are also attached to the panel for robot localization. Both underwater panel and localization algorithm belong to a private repository, so they won’t be shared. The setup described in this paragraph was used in a study case that will be described in the methods section.

II. METHODS

In this section, the semi-autonomous behavior was divided in three main features, which are localization, UVMS pose control and manipulator pose control. It is possible to see how they are related in Fig. 3 flowchart, which explains how the assistant works and how previously mentioned features contribute with it.

A. Localization

The localization problem is present in almost every robotic application. To locate a robot is to know where it is in the environment. There are many sensors that could provide necessary data to estimate UVMS pose in the environment. Each sensor uses different algorithms to establish sensor’s pose and consequently robot pose. Since we chose only to use ROV camera for localization, we decided to use aruco artificial markers for pose estimation. Artificial marker is an object inserted in the camera field of view, that allows correspondence between image pixels and a 3D known model. The association of artificial markers and computer vision algorithms enables camera pose estimation [12]. Aruco is a subset of artificial markers, it is a square matrix of black and white squares with a black border [13].

Aruco marker was chosen because it was successfully used in two others underwater works. In one it had zero flaws in aruco detections with AUV [14]. In another, a underwater manipulator control was successfully made with localization based on arucos [15].

The underwater panel utilized in our simulation had 3 features, so we attached 4 arucos on it. One for each feature and another one for the panel itself. The UVMS already know the relation between aruco pose and features poses. Therefore, if it estimates aruco pose, it is also possible to calculate each feature pose in relation with UVMS. All arucos on the panel were used to estimate UVMS pose. We used an algorithm developed also by BIR* [16] to estimate pose with aruco.

B. UVMS Control

UVMS can move to reach a pose with its thrusters activation. RexROV already had an velocity control [2], which controlled thrusters to move the ROV in a specific velocity in all 6 Degrees Of Freedom (DOF). To control thrusters we utilized an association of Proportional Integral Derivative (PID) controllers, just like in the previous paper [1]. We developed a c++ program that implements digital parallel PID accordingly to Eq. 1. In which \( u_k \) is the control action, \( e_k \) is the error at \( k \) instant and \( \Delta t \) is time sample. Also, \( K_p, K_i \) and \( K_d \) are respectively proportional, integral and derivative gains.

\[
u_k = K_p e_k + K_i \sum_{i=1}^{k} e_i \Delta t + K_d \frac{(e_k - e_{k-1})}{\Delta t} \tag{1}\]

We associated a pose controller with the velocity controller, as it is in Fig. 4. Localization algorithm estimates the \( x, y, z, \) roll, pitch and yaw values from ROV coordinate frame to operation pose. However, roll and pitch normally are not manually controlled and are also stable. Therefore, we use only \( x, y, z \) and yaw values as pose controller input.
Controller output are UVMS velocities in each of these 4 DOF.

We treated the control problem as a black box, without modeling ROV kinematics or dynamics. For that reason, we used Ziegler and Nichols (ZN) closed-loop method [17] to tune PID controllers, which can be done only with experiments. Also, we applied anti-windup to prevent integral error from going to infinity and a saturation in the output to prevent thrusters overload.

![Control diagram](image)

**C. Manipulator Control**

A ROS tool called Moveit [18] was used to control the manipulator. It implements trajectory planning, joint controller, inverse and direct kinematics. With Moveit is possible to define a pose as set-point to manipulator’s end effector. Moveit uses rrt-connect algorithm to generate end effector trajectory to reach that pose avoiding obstacles [19]. Then, applies KDL library [20] to compute inverse kinematics, which define joint positions that put end effector in the determined pose. Subsequently, Moveit calculates the error between current and generated joint positions. Finally, it applies that error as an input to ros-control [21], another ROS tool that can be used to control movement of manipulator’s joints.

**III. RESULTS**

Just like in previous section, here it will be presented localization, UVMS pose control and manipulator pose control results. There is a video on [source](https://youtu.be/omsOCw7Im4) of the UVMS simulation, it is not the same test from following graphs.

**A. Localization**

We had 2 configurations for localization tests. Configuration one used only 4 artificial markers on the panel, as described in methods section. Configuration 2 had 8 markers on the panel, that was made to compare if redundancy would enhance measurement reliability. These configurations were tested in static and dynamic case. In static case, ROV was holding still in front of the panel, as pose estimation algorithm was running. In dynamic case, we moved the ROV in a specific trajectory, to observe if its movement would affect pose measurement.

Fig. 5 shows the results of configuration 1 and 2 for static ROV and Fig. 6 shows the the results of these configurations for dynamic case. We have used euclidean distance and the sum of angular distance modules to measure localization algorithm performance. All markers were utilized to estimate ROV pose, therefore we utilized the mean of each marker estimation as the final value of pose. Table 1 and Table 2 shows statistical data of estimation measurements. \( \sigma \) represents standard deviation and \( \mu \) represents means in these tables.

![Distance between real localization and estimated for configuration 1 and 2 in static case](image)

![Distance between real localization and estimated for configuration 1 and 2 in dynamic case](image)

**TABLE I**

<table>
<thead>
<tr>
<th>Case</th>
<th>Static</th>
<th>Distance</th>
<th>Euclidian (m)</th>
<th>Angular (rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>( \mu )</td>
<td>( \sigma )</td>
<td>( \mu )</td>
<td>( \sigma )</td>
</tr>
<tr>
<td>Config. 1</td>
<td>0.00482</td>
<td>0.00236</td>
<td>0.00741</td>
<td>0.00268</td>
</tr>
<tr>
<td>Config. 2</td>
<td>0.00186</td>
<td>0.00017</td>
<td>0.00592</td>
<td>0.00104</td>
</tr>
</tbody>
</table>
TABLE II
STATISTICAL DATA OF POSE ESTIMATION DYNAMIC CASE

<table>
<thead>
<tr>
<th>Case</th>
<th>Dynamic</th>
<th>Distance Euclidian (m)</th>
<th>Angular (rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>µ</td>
<td>σ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>α</td>
<td>µ</td>
</tr>
<tr>
<td>Config. 1</td>
<td></td>
<td>0.03431</td>
<td>0.03294</td>
</tr>
<tr>
<td>Config. 2</td>
<td></td>
<td>0.03294</td>
<td>0.02794</td>
</tr>
</tbody>
</table>

From what is shown in pose estimation graphs, it is possible confirm that this algorithm successfully works for this application. The utilized underwater environment has an exponential light attenuation and fog effect utilized in camera image, so it can simulate an real underwater environment turbidity and low luminosity. Even with that effect in the camera image, pose estimation error was not bigger than 1cm in static case at any moment. It is also possible to notice in the graph, that ROV movement diminish estimation precision, increasing measurement errors almost ten times. From the statistical table we can conclude that 8 markers are more effective then 4, since it standard deviation and error mean were smaller in both static and dynamic cases.

B. UVMS pose control

In this subsection ZN closed loop tuning will be displayed as well as tuned PID control results. Fig. 7 shows the tuning process for x position process variable. A position unit step input is given to a proportional controller with proportional gain $K_p$. This process is repeated with different $K_p$ values each iteration, until the systems continuously oscillates. $K_p$ value that maintain this oscillation is called critical gain. For x position controller, the critical gain was 8, it is possible to see in Fig. 7 this oscillating behavior.

This process was repeated for y, z and yaw. The critical gain $K_u$ and oscillation period $T_u$ for each of these 4 DOF is listed on Table III. After applying ZN, we obtained $K_i$, $K_d$ and $K_p$, these values are also shown in Table III.

With tuning process finished, the ROV was placed near the underwater panel in simulation and the operation pose was given as a set point. Fig. 8 represent ROV movement towards it operation pose. The second graph in Fig. 8 displays position controller output, which was then used as input for velocity controller. As it is shown in the graph, pose error was close to zero. It is important to remember that localization already has a measurement error, therefore distance from ROV to setpoint includes control and localization errors. Also, there is a saturation of 0.8 and -0.8 in the controller. At this point, ROV software assistant already located underwater panel and moved the UVMS to its operation point. This simulation test didn’t include underwater current disturbances.

C. Manipulator Control

The last step for the assistant to do is to move manipulator end-effector next to the valve, thus it will be easier for the operator to turn it. Fig. 9 shows the manipulator movement towards its goal. These graphs demonstrate that manipulator’s end-effector almost reached its goal pose. Again, it is important to remember that this is the last step of the process, thus includes localization errors, ROV pose control errors and manipulator pose control errors.

IV. CONCLUSION

Semi-autonomous UVMS behavior was achieved. In Gazebo simulation, our algorithm was able to ease operator job and reduce operation time. With just a ROS command, RexROV autonomously searches for artificial markers and move to ideal operation pose. Then, Oberon7 manipulator
moves towards the valve and gets close enough so the operator can finish turning valve operation. Even if numerically Oberon7 end-effector position error did not went to zero, this is a positive result. This whole process aimed to move the manipulator’s end-effector close to panel’s valve and it was successfully achieved.

Nevertheless, there is room for improvement. There are others control methods that could perform better than PID. Also, ROV pose control based on kinematics and dynamic model is being developed. Finally, to reduce operation time even more, manipulator trajectory planning to execute the whole turning valve operation is also being developed.

REFERENCES

[10] https://uuvsimulator.github.io/

---

**Fig. 9.** Linear and angular errors for manipulator go to pose operation