Wet Whale: Rescue mode engaged

1st Hellwig Jonathan	2nd Adamanov Asan	3rd Dierfeld Sven	4rd Dirikgil Tolga
Wet Whale	*Wet Whale*	*Wet Whale*	*Wet Whale*
UH	*TUHH*	*TUHH*	*TUHH*
7381194	11861429	21595967	21158403
jonathan.hellwig@studium.uni-hamburg.de	asan.adamanov@tuhh.de	sven.dierfeld@tuhh.de	tolga.dirikgil@tuhh.de

***Abstract*—To speed-up underwater rescue, autonomous ROVs can be used, which need to be able to detect objects, find a possible path and then detect the person in need of rescue. This paper proposes an approach for a ROV to autonomously navigate through a hole and around obstacles from one side of a tank to the other while simultaneously being able to detect faces and update the map based on detected obstacles. For that, a controller based on multiple SISO PID-Controller, an object detection algorithm using Apriltags, a path planning algorithm, and a human detection is implemented. After that, the results from a simulation and an experiment are presented.**

***Index Terms*—Underwater ROV, PID-Controller, Object detection, Path planning, Face recognition.**

I. Introduction

In order to use ROVs for underwater rescue, developers have to solve many challenging problems. The ROVs need to locate themselves in the specific body of water, map their surroundings, and detect drowning persons. All that while detecting other moving obstacles. This paper tries to solve these problems in a simplified setting and is a continuation of the paper [10]. It is part of our effort to develop an autonomous ROV for underwater rescue. The outline of the paper is structured as follows: Experimental design, Methods, Result, Project schedule, and Conclusion.

II. Experimental design

To test the approaches proposed the authors used the model BlueROV2 manufactured by Bluerobotics. It is equipped with a front and a downwards camera, a pressure sensor, an IMU sensor, and lights. The tests were conducted in Gazebo and in the laboratory using a tank with the dimensions 3.25 x 1.67 x 1.4 m. To enable robust localization the authors attached 36 AprilTags to the bottom of the tank. The goal is to follow a list of set destination points, which are alternately on both ends of the tank, while diving through a hole in a wall in the middle of the tank and also avoid an obstacle. Every time the ROV reaches a destination point he is to first rotate until he detected the wall. Then it rotates in the direction of the destination point and dives to the destination point.

1) Simulation: To test path planning and object detection the authors added a wall with one AprilTag on each side and a hole in the middle which can be seen in Figure 2a. An additional tag represents an object in the tank. Due to the nature of the simulation, a very simple model of the front camera was chosen. To simulate AprilTag detection the authors created a Gazebo plugin that generates a displacement vector and a rotation to the detected AprilTag in the coordinate frame of the ROV.

2) Laboratory Experiment: Further, to emulate the conditions in the simulation the authors inserted a wooden wall in the middle of the tank with a hole in the middle. An additional object is not used for the experiment. In total, the authors set out to conduct three different experiments in the laboratory to test the capabilities of the ROV. They tested three hypotheses:

- ROV manages to follow a path five consecutive times without hitting a virtual wall inserted into the path planning map.
- ROV correctly detects the tag on the wall at the position we measured for five different angles.
- ROV successfully navigates from the side of the tank to the other without bumping against the walls on the sides or the wall inserted into the middle of the tank.
- Detect a person on a printed and laminated 2D picture as a person.

This approach allowed the authors to test the different components of the system separately to identify behavior not observed in the simulation.

III. Methods

A. Localization

In previous work, we developed our approach for a solution of localization task [10]. However, in this work, we do not use our localization but get the pose and the position from the given localization, which uses apriltags on the floor of the tank and a Kalman filter.

B. Controller

For this project, we want to implement a controller that is able to apply the necessary thrusts independently of the yaw-angle γ_{yaw} of the ROV relative to the inertial coordinate system. For that, we modify the controller we use in [10] for the x- and y-direction. Instead of the global instances in x- and y-direction we plug-in the relative distances of the ROV to the setpoint in the ROV coordinate system respectively as seen in the Figure 3. This approach, which uses multiple SISO PID-Controller, has low implementation effort and is easy to tune. Using the position from the localization p_{ROV} and the setpoint p_{set} from the path planning, we can calculate the distance vector p_{dis} of the ROV to the setpoint in the two-dimensional

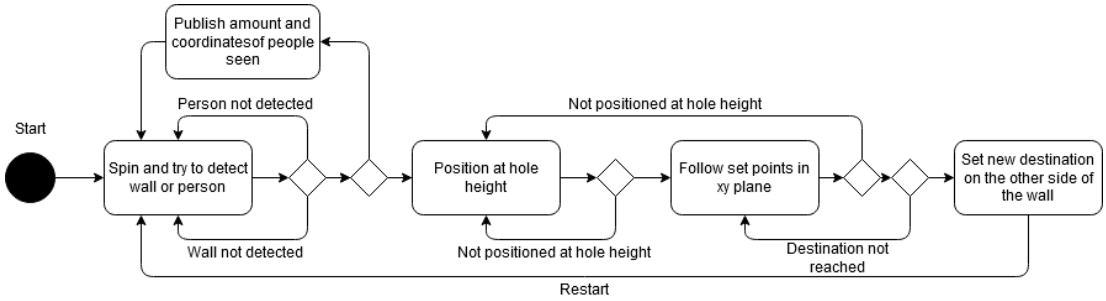
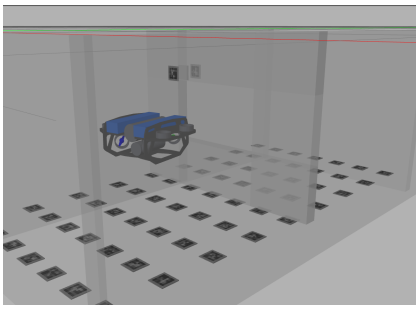
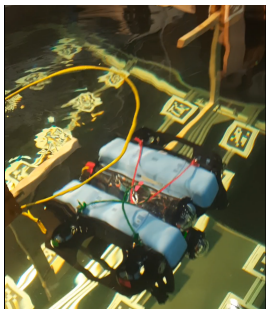


Fig. 1: The ROV operation scheme.



(a)



(b)

Fig. 2: Depiction of the (a) simulation environment with the wall and (b) experiment tank and the wooden construction.

xy-plane and than transform it into the ROV coordinate system using the transformation matrix T_{yaw}:

$$T_{yaw} = \begin{pmatrix} \cos(\gamma_{yaw}) & \sin(\gamma_{yaw}) \\ \sin(\gamma_{yaw}) & \cos(\gamma_{yaw}) \end{pmatrix} \tag{1}$$

$$p_{dis} = T_{yaw} \cdot (p_{set} - p_{ROV}) \tag{2}$$

We then send the x- and y-component of p_{dis} to the PID-controller for thrust and lateral thrust respectively. The PID-Controller for yaw also was modified by putting a ceiling on the distance between setpoint and state.

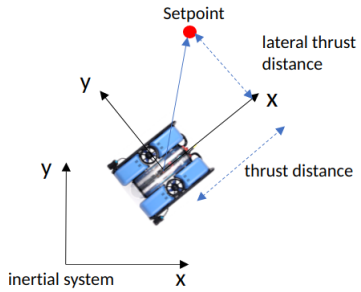


Fig. 3: A visualization of the controller.

C. Object detection

Using the data from the front camera the authors use the AprilTag detection package [3] [4] to obtain a displacement vector $v_{camera}^{tag} \in \mathbb{R}^3$ from the camera to the detected tag in the coordinate frame of the camera. They use this vector in combination with the current position p_{ROV}, rotation matrix $R_{ROV}^{tank} \in \mathbb{R}^{3x3}$ obtained from the orientation of the ROV and the base to camera vector $v_{base}^{camera} \in \mathbb{R}^3$ to derive the position of the wall

$$p_{\text{wall}} = p_{\text{ROV}} + R_{ROV}^{tank} v_{base}^{camera} + R_{ROV}^{global} v_{camera}^{tag} \tag{3}$$

Since the dimensions and the position of the tag relative to the wall are known the authors update the map accordingly. To note here is that the orientation of the camera frame and the ROV frame are assumed to be equal.

D. Path planning

The planning can be divided into two sub-tasks: global route planning and local path planning. The global planner is responsible for finding the route on the road network from origin to the final destination. After the user defines the final destination, the global route subsystem computes a set of Paths, $P = \{P_1, P_2, \ldots, P_n\}$, considering the current route, BluROV2 State. A Path $P_j = \{p_1, p_2, \ldots, p_m\}$ is a sequence of poses $p_i = (x_i, y_i)$. The most common graph search-based techniques for path planning of autonomous robots are Dijkstra and A-star. The Dijkstra algorithm [1] finds the shortest path $g(v)$ between the initial node and goal node of a graph. The A-star algorithm [2] is an extension of Dijkstra, which performs fast graph search by assigning weights to nodes based on an additionally a heuristic-estimated cost $h(v)$ to the goal node, and the total cost becomes $f(v)$. The heuristic function is selected as L_2 norm. For the computation of local path-planning is usually chosen a faster approach which is Sampling-based and called Rapidly-exploring Random Trees (RRT), RRT*. However, due to the time limit, the A-star Algorithm is implemented for both cases: global and local path planning. Furthermore, in order to smooth the calculated path from a grid map, the path is fitted to a Bézier Curve [6].

$$f(v) = g(v) + h(v) \tag{4}$$

E. Human detection

To detect people in danger of drowning underwater it is mandatory to detect people visually at first. For this purpose, the front camera of the BlueROV2 is used to observe its environment. The obtained data is then processed with OpenCV [9]

and the 'you only look once - YOLOv3' algorithm [7] [8]. A neural network as used with 'YOLOv3' expects preprocessed frames, Binary Large Objects, hereinafter stated as 'BLOB', as input. Usually, this preprocessing includes mean value subtraction for the R, B, G Channels followed by scaling those channels but YOLO only requires scaling by the scaling factor $\sigma = \frac{1}{255}$ to normalize the pixel values between zero and one. For 'YOLOv3' to be able to detect objects, it needs to use a trained model. For simplicity the pre trained model 'YOLOv3-320' is used. Once a person is detected as seen in figure 4 the amount of people seen as well as the ROVs yaw angle and current position are being published. By that rescue divers can determine at which position they need to get into the water to rescue drowning people. To get the actual 3-dimensional position of the detected person, a stereo camera is implemented without using the BlueROV2. It is calibrated using the standard hand-eye calibration using a checkerboard. Once a person is detected the actual position of the person is calculated by using the depth from the generated depth map.

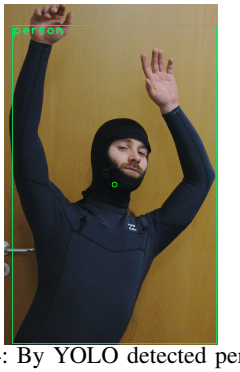


Fig. 4: By YOLO detected person

F. Integration of the components

In the global path planning solution, the path is usually calculated in advance and after that send to the controller. This can result in undesirable behavior of the robot. For example, if the controller is not perfectly tuned, the robot will diverge from the path and is not able to follow the path which is calculated in an advance. To avoid this the authors combine global and local path planning, i.e. the path is calculated online simultaneously and updated each time. To find a setpoint we iterate backwards through the path, starting at the destination and choose the first point that is closer to the ROV then a set distance. Instead of sending the setpoint to the controller after each time the path is calculated, which is slow as the path planning is computational expensive. We store the latest path and continuously calculate and send the setpoint in parallel. This helps to avoid unnecessary waiting until the calculation of the whole path is finished.

IV. Results

A. Simulation

1) Object detection: Figure 7a shows the position of the detected AprilTag on wall in the simulation. All values remain constant even when the ROV is moving. Therefore, all calculations to convert the displacement vector to the position in the coordinate system are correct.

2) Controller: The setpoint is calculated to be 15 cm in front of the ROV which enables a more aggressive PID Controller for the lateral thrust and thrust as overshoot is limited due to a limited distance between state and setpoint. The ceiling for the yaw-controller is at 20°. In Table I the values for the proportional gain k_p, the derivative gain k_d and the integral gain k_i for each PID-Controller can be seen. In Figure 8a the position of the ROV and the setpoints in x- and y-direction are shown.

3) Path planning and mapping: The time complexity of the A-star algorithm depends on the heuristic, which is in our case is the L_2 norm. In the worst case of an unbounded search space, the number of nodes expanded is exponential in the depth of the solution (the shortest path) d: $O(b^d)$, where b is the branching factor (the average number of successors per state) [5]. As we calculate the path independently of the controller we can keep the grid size small and run the A-star algorithm simultaneously, even though it results in a costly computation time.

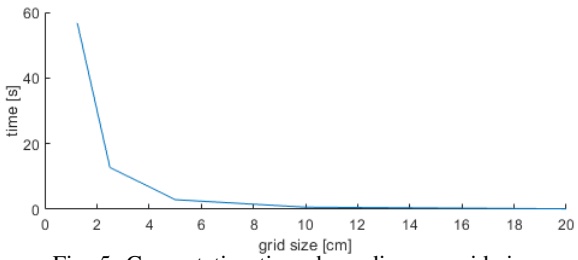
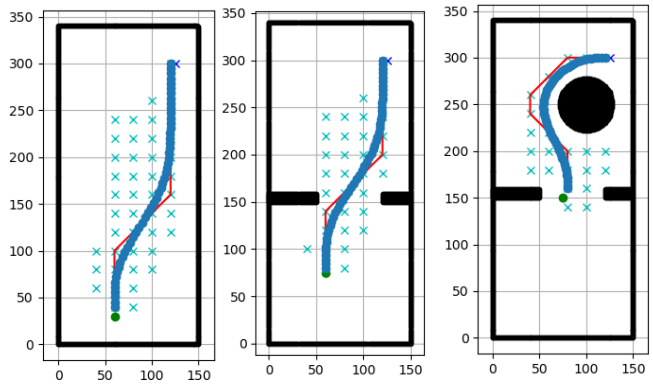


Fig. 5: Computation time depending on grid size

Figure 5 shows how the computation time increases with smaller grid sizes. Therefore, the grid size is selected to be 5 cm as a trade-off not being too slow on a local host computer. Furthermore, to avoid unexpected collisions with the wall or box, the buffer is set to 30 cm. Figure 6 shows the 2D paths using the A-star Algorithm for three different maps. Initially, the planner is only aware of the static map shown in Figure 6a. As soon as the obstacle detection stack detects AprilTags that are attached to the wall/box, it calculates their coordinates, the path planning receives the position of the obstacles and updates the map with the detected obstacles. Immediately after that, the new path is recalculated as it is working in parallel but this time with a new map Figure 6b, 6c. To simplify the complexity of the orientation consideration during the mapping on the grid, the box is mapped as a circle onto the grid map, Figure 6c. Another simplification, the authors assume that the wall has only one AprilTag on its edge, therefore, the length of the hole is encoded. This could be avoided by attaching multiply AprilTags on each corner of the wall. Also, the box dimension is encoded.

The update map stack is heavily dependent on correct received obstacle coordinates which are calculated in the obstacle detection stack. The results of path planning are independent of everything since it finds always a solution

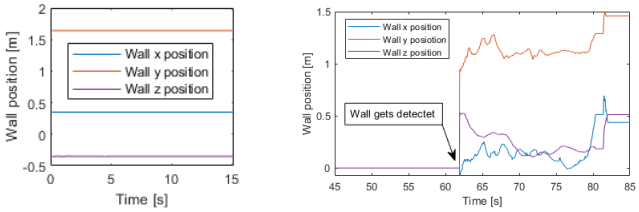


(a) initial map (b) with wall (c) with box

Fig. 6: Three depictions of the path planning in the xy-plane with A-Star (red) with fitted Bezier Curve paths (blue).

based on the given grid map. For further details of the A-star Algorithm and its optimality of finding a least-cost solution, please take a look in [5].

B. Laboratory Experiment



(a) Simulation (b) Laboratory

Fig. 7: Detected wall position

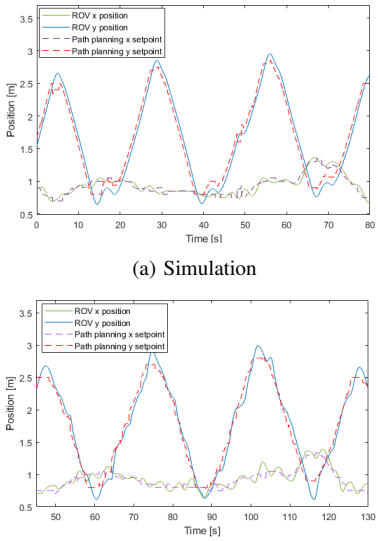
1) Object detection: Figure 7b shows the position of the tag attached to the wall inside the tank. The wall is firmly fixed while the ROV is moved to different positions. The expected result is a constant value in x, y, z. However, the values oscillate rather strongly and do not assume a constant value. This error likely occurred due to an incorrect transformation between coordinate frames. The coordinate frame of the camera does not coincide with the coordinate frame of the ROV. In the simulation experiments, the authors did not observe this error because the camera frame and ROV frame were the same. This fact was overlooked and lead to erroneous results.

2) Controller: In table I the values for the proportional gain k_p, the derivative gain k_d and the integral gain k_i for each PID-Controller can be seen.

	Vertical thrust		lateral thrust		thrust		yaw	
	sim	exp	sim	exp	sim	exp	sim	exp
k_p	4.0	5.0	2.0	3.0	2.0	3.0	0.3	0.5
k_d	0.018	0.02	0.0	0.007	0.0	0.007	0.0	0.005
k_i	0.006	0.006	0.0	0.002	0.0	0.002	0.0	0.003

TABLE I: Parameters for the PID-Controllers from the experiment and the simulation.

In Figure 8b the position of the ROV and the setpoints in x- and y-direction are shown. Due to time constraints during the experiment and the infeasible tunning compared to a step input, the PID-controllers are not fully tuned. However, it can be seen, that the ROV follows the setpoint quite well with some overshoot and acts quite similar to how it works in the simulation.



(a) Simulation

(b) Laboratory

Fig. 8: Setpoint following for the x- and y-direction

3) Path planning and mapping: The path planning algorithm depends on the position of the detected wall and the current position and destination point. Since the results in the simulation show that the planning and mapping work as intended, no changes were observed in the laboratory experiments.

4) Human detection: While the ROV was supposed to stop at the position where it detected a person underwater this didn't work due to the slow image computing using a cpu. While the ROV was already past the person the image processing still didn't process the image containing the person. Nonetheless the detection of a person using OpenCV and YOLO is working as seen in image 4.

V. Conclusion

The scope of this project was narrow due to constraints on available sensors and testing capabilities. However, the approaches discussed in this paper are promising in a more general setting. Further testing of face detection capabilities and 3D mapping using a stereo camera could provide useful insights to develop automatic rescue robots for open water. Another topic of interest is extending the 2D grid to a 3D grid and calculate a path in 3D. This would allow for more rapid navigation to a target object. However, additional computational cost and modification of the current control scheme have to be considered.

VI. Project schedule

Name	Work packages
Tolga Dirikgil	human detection, real wall construction, video
Asan Adamanov	path planning, map update, integration of components
Sven Dierfeld	contoller design, apriltag plug-in, object detection, gazebo, integration of components
Jonathan Hellwig	Path planning, apriltag plug-in, object detection, integration of components

References

[1] Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Numerische mathematik 1.1 (1959): 269-271.

[2] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4.2 (1968): 100-107.

[3] Wang, John, and Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection." 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016.

[4] Malyuta, Danylo. Guidance, navigation, control and mission logic for quadrotor full-cycle autonomy. MS thesis. ETH Zurich, 2018.

[5] Russell, Stuart; Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (2nd ed.)

[6] Farin, Gerald. Curves and surfaces for computer-aided geometric design: a practical guide. Elsevier, 2014.

[7] Redmon, J.; Divvala, S.; Girshick, R. and Farhadi, A. "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[8] Redmon, Joseph & Farhadi, Ali. (2018). YOLOv3: An Incremental Improvement.

[9] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[10] Hellwig J., Adamanov A., Diefeld S., Dirikgil T., (2020). Wet Whale: Hey, where am I??