A Learning-based Modular Heterogeneous USV and UAV Team in the Maritime RobotX 2022 Competition

Po-Jui Huang[1,†], Ching-I Huang[1,†], Sin Kiat Lim[1], Po-Jui 'Bory' Huang[1], Ming-Fong Hsieh[1], Lai Sum Yim[1], Yu-Ting Ko[1], Hao-Yang Hung[1], Yi Chen[1], Jie-Xin Liu[1], Li-Wei Liou[1], Sun-Fu Chou[1], Yi-Chen Teng[1], Kai-Jui Weng[1], Wang-Chuan Lu[1], Hsueh-Cheng Wang[1,*]

***Abstract*—This paper describes the architecture and implementation of a heterogeneous team comprising of unmanned surface vehcile (USV) and aerial vehicles (UAV) for the RobotX Challenge 2022. The modularity designs of sensor tower, autonomy box, and communication modules provide hardware-in-the-loop (HIL) developments. This work tackles the sim-to-real gap of learning-based models. We generate a virtual RobotX perception dataset with realistic appearances and automatic labelling. The Gazebo simulation provided by Virtual RobotX (VRX) consists of physical engines of high fidelitous current, wind and sensors streams, which are used to develop deep reinforcement learning algorithms of collision avoidance and navigation. We show that a deep RL policy of USV learns the vehicle dynamics in maritime environments implicitly and outperforms other models only trained on UGV. For fleet management, we consider situational awareness of human supervisor at base station by including a virtual reality (VR) interface. We develop a behavior tree for each RobotX 2022 task with reusable subtrees and nodes for the USV and UAV. Quantitative Evaluations of collision avoidance and RobotX tasks are carried out in simulation, and the proposed approaches are deployed and tested in a real wave adaptive modular vehicle (WAM-V) platform.**
For a supplementary video visit: https://vimeo.com/758819296
For team webpage visit: https://arg-nctu.github.io/robotx-2022/

***Index Terms*—Deep Reinforcement Learning, Heterogeneous Robot Team, Sim-to-real, Fleet Management, WAM-V, Behavior Tree**

I. Introduction

RobotX competition has brought together universities and research teams around the world since 2014. Each team have developed both hardware designs and software algorithms for the unmanned surface vehicle (USV) platform. Perceptions and autonomous navigation support real-time decision-making to perform well-defined tasks such as path following, docking, etc., and avoid collisions at the same time.

A. Autonomy Challenge

Nevertheless, Building an autonomous unmanned marine system is challenging in many aspects. The hardware should include waterproof design, compact electronics, together with cooling system are crucial for weather conditions. The navigation and control system should consider wind and wave conditions and cope with dynamics and uncertainties of the marine environments. Those challenges remain without baselines and standardized evaluations because benchmarking in marine environments is hard. Recently, The open-source Virtual RobotX (VRX) simulator [1] is proposed to support the development and evaluation of USVs operating in such environments. More importantly, there are still big questions arisen from the previous RobotX competitions.

Fig. 1: We proposed an autonomous surface/aerial system with modular design. The system is capable of running in real robots and simulations.

1) What are the gaps between real and virtual environments and how to bring the gaps closer?
2) How could the developed system/techniques be generalized to other robots or in other environments?
3) What are the basic principles to manage a team of heterogeneous robots in multiple tasks and to advance the developments?

Our approaches in the RobotX Competition 2022 are built upon our recent developments. The Duckiepond [2] is motivated by the Duckietown [3] and include a fleet of USVs with low-cost, modular designs of sensing, autonomy, and control components. Subsequently, a heterogeneous team of unmanned ground vehicle and blimp robot is designed for search and rescue missions [4]. For potentially foggy or dark environemnts, we develop a sensor tower of millimeter-wave (mmWave) radar, which is lightweight and inexpensive, for learning-based autonomous navigation. A cross-modal contrastive learning for representation (CM-CLR) method is proposed maximizes the agreement between mmWave radar data and LiDAR data [5]. We also leverage the use of ultra-wideband (UWB) for centermeter accuracy localization for autonomous navigation [6]. Our recent work on an immersive VR interface and simulation environment [7] allow the monitoring and control of real robots remotely. Many of the techniques developed over the years have been adopted for RobotX 2022.

B. Autonomy Objectives

We summarize our objectives as follows:

1) **Modular designs of sensing, autonomy, control, and communication.** Our WAM-V system include 4 sensor towers, 3 autonomy boxes, control and communication, and 6 localization anchors. We also include an quadrotor UAV with a vacuum-based soft gripper in the heterogeneous team. All of the modular components have been developed for several iterations since 2018.
2) **Sim-to-real Learning-based Perception and Autonomy.** Our team relies heavily on Unity to render realistic training data with automated ground truth labels for perception tasks. We leverage the use of VRX Gazebo simulators [1] to train deep reinforcement learning (RL) for collision-free navigation policy. In particular, we tackle the problems of a) sim-to-real gap of virtual and real visual appearances, b) curriculum transfer learning from UGV to USV in maritime environments.
3) **Heterogeneous robot fleet management and interface.** Our system adopts the recent efforts of the Duckietown platform for flexible fleet managements using the Duckietown Shell (DTS). The fleet can be discovered with heartbeat (1Hz) health status of sensors and computing units. We also include an immersive VR interface for fleet monitoring.

II. Engineering Design Decisions

A. Problem Statements and Challenges

- **Autonomous Navigation** Our USV is assumed to operate in the environments with some static floating objects, signs, and natural interference, such as wind and wave. We aim at developing collision avoidance control policies for goal navigation, whose goal is assigned by GPS or the target from learning-based perceptions.
- **Learning-based Perception** Due to the light conditions, viewing angles and distances in the images, we utilize deep-learning-based approach to recognize the targets, such as the dock with the color blocks, and the light buoy either on or off.
- **Fleet Localization** The key localization challenge for a heterogeneous robot team is to estimate precise relative positions, in order to fuse the observations obtained form the USV and the UAV. The technology we utilize is SBL-UWB localization for precise relative positions between the vehicles within a 20m range.

B. System Architecture

As shown in Fig. 2, the proposed system is designed to tackle the challenges of heterogeneous robot team, perception & autonomy, and localization & communication. For this challenge, our heterogeneous robot team consists of an USV and an UAV. Both of them are capable of autonomous navigation to a target position. They are able to recognize the specific objects in the environment through training via EfficientDet in the simulator. As to autonomous operations, we rely on deep reinforcement learning (RL) for goal navigation and collision avoidance, serving as the local planner in Fig. 3.

For fleet management, we designed behavior tree involving control flow, condition, and action nodes for each task. Even though all tasks need to be performed autonomously, we value the functionality of situational awareness of the human supervisor at the base station. VR interface provides immersive information from the sensors and transmitted in real time via our high bandwidth customized WiFi module (TVL). Xbee module is for heartbeat and system monitoring. Furthermore, we also apply our developed localization technology, SBL-UWB, at the anchors, so the localization and even short-range goal point navigation can be also realized through measuring the distance to other anchors nearby in real time. We list different kinds of our developed sensors for each task in Table. I.

TABLE I: The application of our developed sensor module

	ST1 (Lidar)	ST2-4 (ZED)	Anchor (UWB)	GPS & IMU	HSI
Obstacle avoidance	v	v		v	
T2: Entrance exit gate	v	v		v	
T3: Follow the path	v	v		v	
T4: Wildlife encounter	v	v		v	v
T5: Scan the code	v	v		v	
T6: Detect and dock	v	v		v	
T7: Find and fling	v	v		v	
T8: UAV Replenishment		v	v	v	
T9: UAV search and report		v	v	v	

III. WAM-V and Hardware Developments

Wave Adaptive Modular Vessel (WAM-V) is an ultra-light, modular vessel. We design a two-layer payload with 4 semi-spherical-shaped *sensor towers*, shown in Fig. 4. The upper layer is landing platform for UAV with a composite AprilTag markers. The lower layer houses the sensing, computing, and control units.

A. Design Considerations and Objectives

The hardware and functional requirements are:

- On-board computation over wide field-of-view (FOV) video streams for task-relevant object detection;
- Navigation toward a given goal (subgoal);
- High frame rate, low latency, and high reliability for localization and mapping;
- Manipulator with sufficient payloads for the launcher and HSI camera;
- Communication to enable heartbeat and emergency stop;
- Waterproof and prevention from overheat.

B. Propulsion Components

1) Propulsion Design Selection: Our propulsion system is tightly coupled with the autonomy function, in which a deep RL model outputs angular and linear velocities. We includes a Roboteq controller connecting to two Minn Kota thrusts for heading and two Torqeedo Cruise 6.0TS for linear velocity, shown in Fig. 4. We install two motors at front for heading

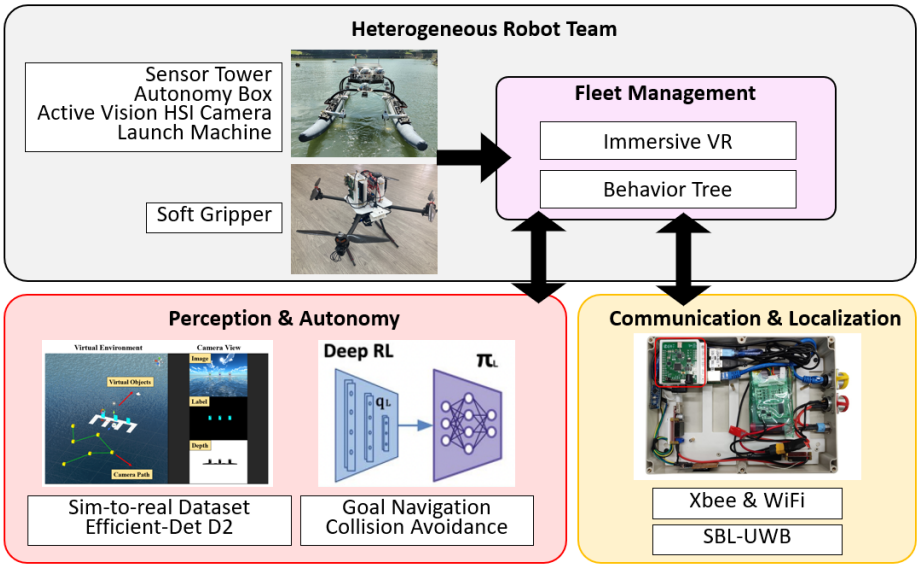


Fig. 2: System architecture: (1) Proposed heterogeneous team of UAV and USV; (2) algorithms used for perception and autonomy; and (3) fleet managements.

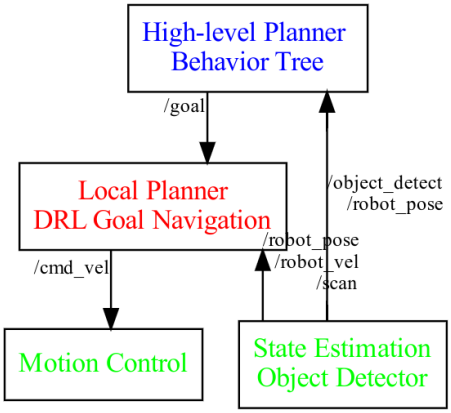
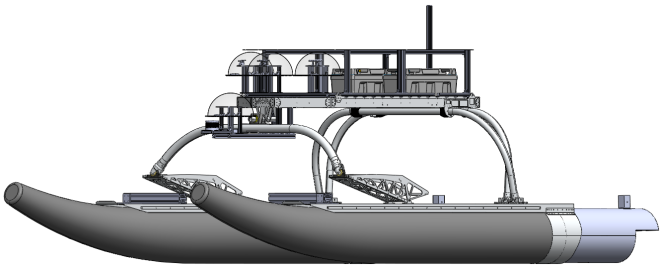
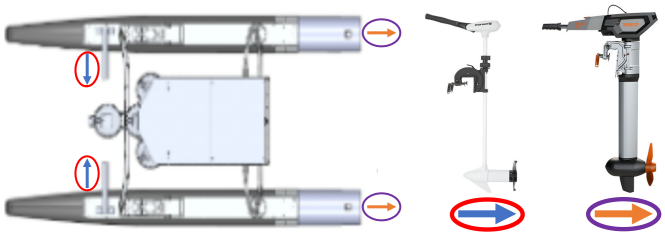


Fig. 3: We consider the deep RL policy a local planner, whereas the task-relevant planner is implemented by behavior tree.



(a) WAM-V Overview



(b) Propulsion Designs

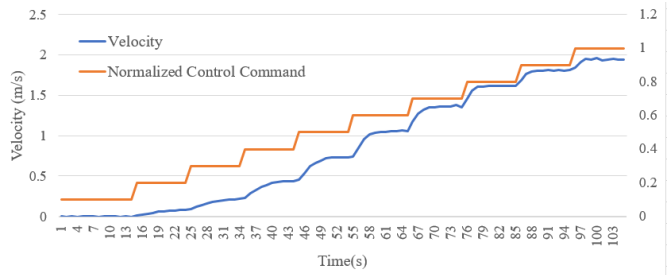
Fig. 4: WAM-V payload and propulsion design.

control instead of one motor to prevent frequent forward and reverse operations. We have considered other propulsion designs, but each has some drawbacks:

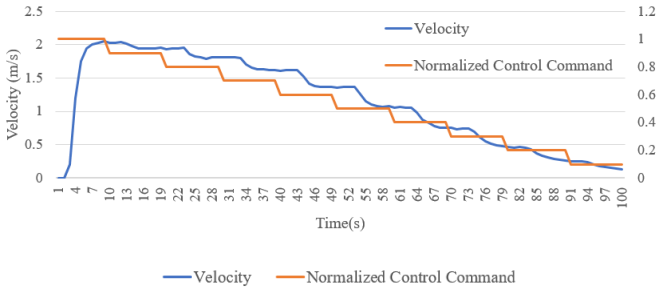
- **Differential drive** was used in our team for RobotX 2018 competition. However, such design is insufficient to provide agile movements, in particular for detect and dock task. We also found the motors might be damaged under the uses of frequent forward and reverse operations.
- **Omni-directional drive** is considered and implemented given the 4 motors installed. However, we found the sim-to-real capability less effective due to the weight distributions on the payload. An extra calibration may be required.

2) Power Supplies and E-Stop: An emergency stop can be activated by a remote control to shut the power to the propulsion system immediately. The control and communication system is packed in waterproof case, and consumes 24 volts as input through a connector to support Roboteq controller and the cooling module. A power converter is used to step down to 12 volts for wireless communication system.

3) Analysis of USV Acceleration/Deceleration in Maritime Environments: As shown in Fig. 5(a), we increase the control commands (normalized between 0 and 1) and observe the



(a) Control commands are gradually increased.



(b) Control commands are set to max (dash line) then gradually decreased.

Fig. 5: Control commands vs. vehicle velocity in maritime environments.

responses of the vehicle velocities. We found that there have been significant delays while the vehicle is in low speed, but the delays decrease when the velocity is above 1.3 m/s. In Fig. 5(b) we set the control commands to max, and the vehicle reach to 2.1 m/s after 9 seconds. There exist a few seconds delay of responses to decreased control commands, but the delay is shorter in low velocity. We found similar effects in real and simulation environments.

C. Sensor Tower

Sensor tower is a *standalone* multi-modal perception module containing the sensors, and computing units, and power supply (a Li-Fe battery). We have developed two variants of sensor tower based on stereo vision and LiDAR ranging, shown in Fig. 6. We use three stereo vision sensor tower at the forefront of the payload for object detection, and one LiDAR tower at a lower level of WAM-V for navigation and obstacle detection. Note that sensor tower is only responsible for providing visual perceptions, but not the computations of the autonomy and control, i.e., the deep RL algorithm in our system.

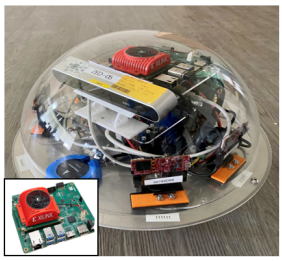
The implementation to such isolation of distributed, multi-machine system is motivated by overcoming the challenges we faced in the RobotX 2018 competition. Compared to our "all-in-one" design in 2018, where all sensors were plugged into a centralized industrial PC, there exists several significant drawbacks:

- **Occasional latency issue** is observed due to the high bandwidth data streams from multiple sensors.
- **Computing resource requirement** is different for each sensor. For example, stereo cameras usually need to compute depth map via semi-global matching algorithms, which may be accelerated by FPGA or GPU.
- **Power consumption** for each sensor via USB port or external sources need to be carefully considered to guarantee stable operations.

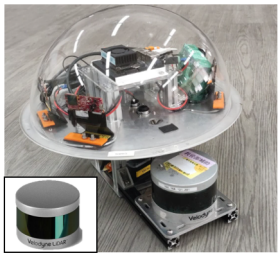
1) Stereo Vision with FPGA and GPU: Stereo camera provides both RGB and depth images for outdoor environments, which is ideal for many tasks. We include a ZED stereo camera with precise depth sensing within 15m. However, the field of view (FOV) $90°(H)\times60°(V)$ is limited and and computing requirements is high. ZED camera comes with NVidia Jetson device for depth computation on GPU . We also include an FPGA (Field Programmable Gate Arrays) board (AMD Xilinx KV260) for accelerating parallel processing with lower power consumption. However, we found a decreased frame rate on carrying out both depth computation and object detection on one device. Currently the depth computation is performed on GPU, and object detection is on FPGA device.

2) Range Sensing with LiDAR and mmWave Radar: .

We utilize a VLP-16 LiDAR (Velodyne's Puck) of 16 beams with 905-nm wavelength for detection obstacles within 50 meters. The LiDAR inputs are used in our deep RL network for autonomous navigation and collision avoidance. An alternative ranging sensing in adverse (such as foggy) environment is the mmWave radar, which uses the electromagnetic wave that can penetrate the small particles. We arranged 4 single-chip mmWave radars reaching 270 degrees FOV. The sparse and noisy point clouds from mmWave radar can be reconstructed to more dense LiDAR-like signals via a conditional generative model (cGAN) developed by our previous work [5].



(a) Stereo vision



(b) LiDAR

Fig. 6: Variants of sensor tower designs. (a) a stereo camera with FPGA and GPU computing units; (b) LiDAR and mmWave.

D. Active Vision HSI Camera

The Hyperspectral Imaging (HSI) camera provided by RoboNation OpenHSI is a line scan camera with a 10.7° FOV. We reason that given such a narrow line-scaned FOV, the HSI camera should be mounted either on a fast moving UAV, or on a robot arm on WAM-V with precise active control of the

sensing platform (active vision). We decide to implement the latter solution by considering the followings:

- The former requires POE electronics with too much payload and power consumption on UAV.
- A precise active control pointing at target on UAV is challenging.

We design a robotic arm (Trossen Robtics ViperX 300) with the HSI camera mounted on the end effector with a laser rangefinder to measure the distance to the target object, shown in Fig. 7. We also include A a 48V Power Over Ethernet (POE) to supply power. The HSI camera captures light of wavelength of multiple spectral bands from visible light to near infrared light (430 to 900 nm). The HSI camera sealed in a waterproof enclosure prevents water failure. We used the open source OpenHSI software for detecting color blocks, shown in Fig. 8.

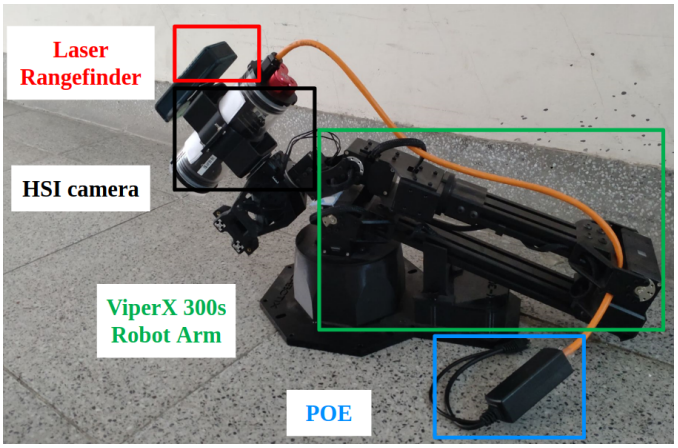


Fig. 7: HSI camera mounted on a robotic arm with precise active control.

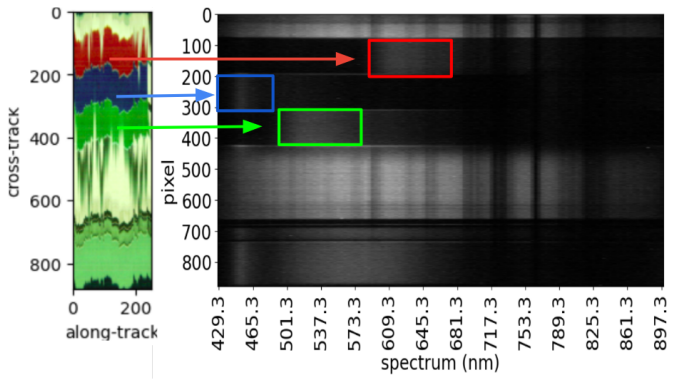


Fig. 8: Spectrum distribution of RGB color block

E. Launch Machine

The launch machine consists of two main parts (1) the feeding part, which is for projectile storage and feeding, and (2) the launching part, which is for launching the projectile to the target area. The pipeline for auto-launching, the launch system will be integrated with the perception system. Once the result comes from the perception system, the feeding part will rotate and the projectile will be free-falling, with a poking mechanism to ensure the projectile will touch the launching spot.

F. Autonomy Box

1) Hardware in the Loop (HIL): The separation of perception and autonomy functionalities in hardware is to facilitate hardware in the loop (HIL) development and testing. The Autonomy boxes consist of a NVidia Jetson Nano computer for deep RL inference. We set up a simulation environment in workstation, where each autonomy box represents one simulated robot. Each receives the observations of sensory data, performs autonomy algorithms, and transmits control commands as actions in the simulator.

2) Hardware Components for Health Status ans Safety: We include a Raspberry Pi 3B and Arduino UNO to support the health status.

- **Temperature and humidity sensors**. Monitoring the temperature within the closed waterproof box and the CPU temperature is crucial for system robustness or in long-term operations (several days). The within-box temperature may reach 60 to 70°C under direct sunlight.
- **XBee module for heartbeat**. XBee is a communication module of mesh network that uses ZigBee protocol of 900 MHz. The device has a low bandwidth and low power consumption, which is well-suited for heartbeat.
- **Electronics Safety and Protection.** The autonomy box is also designed to directly control Blue Robotics thruster with a hardware protection by a disposable components (fuse). We include two electric transducers: one is 15-40V for thruster and the other is 12-24V to 5V for the rest of the components. The function to cutoff the circuit is for protection the low-voltage hardware and emergency stop.

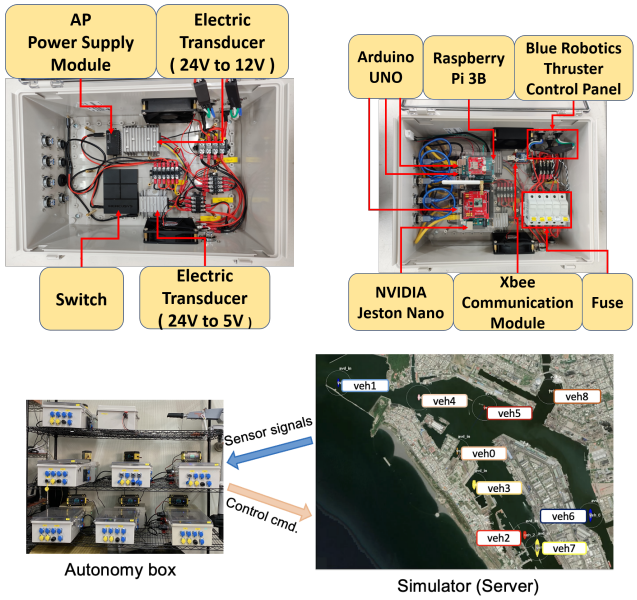


Fig. 9: Autonomy box (top) and hardware in the loop developments (bottom)

We used a software package, Duckietown Shell (dts), designed by the Duckietown [3] community to monitor the health status of the robot fleet.

G. High Bandwidth Communication for Situational Awareness

We include a high bandwidth customized Wi-Fi module (TVL, provided by our industrial partner K-Best Inc.) TVL includes adjustable 2.2 to 2.4 GHz frequency bands to minimize the interference by other Wi-Fi users. TVL is used as our major communication among the shore-side base station, WAM-V, and the UAV. The end-to-end effective range is up to 6 km with an amplifier on each end. Such high bandwidth communication is capable of VGA quality video streams to enable monitoring and situational awareness of the human supervisor at the shore side.

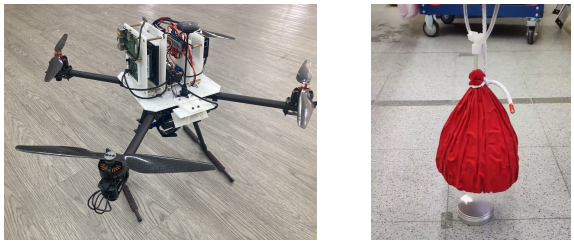
H. UAV with Soft Gripper

To accomplish UAV replenishment and search and report tasks, we use a standard quadcopter. Although our team has developed a robotic blimp in previous work [4], we consider that aerial manipulation requires highly precise controls. Blimp has a longer operation time, but is more challenging to control under aerial dynamics and winds outdoors. Our focuses are on:

- **A flying "sensor tower"**. We consider the UAV as an extended sensor tower, which is capable of scanning a large area while cruising in the sky. Similar to our sensor tower design, we include a ZED mini stereo camera mounted toward the heading direction and tilted down 60 degrees. The onboard computing unit is NVidia Jetson Xavier NX is connected to a commonly used flight controller PixHawk Pix32 via MAVROS protocol.
- **Aerial manipulation with a soft gripper**. Considering replenishment task requirements, we implement a vacuum-driven origami soft gripper based on the magic ball design proposed in [8]. The robust grasping capability over two-finger gripper and waterproof make it suitable for the task. The soft gripper is installed beneath the UAV with a short cable, which enable UAV to accomplish replenishment without landing on the floating dock. We include an Intel RealSense Depth Camera D435 mounted downward inside the magic ball skin for detecting target object, shown in Fig. 10.
- **Fleet management over UAV and USV**. We install an additional Intel RealSense Depth Camera D435 mounted backward and tilted down 60 degree as secondary detection sensor, which is responsible for detecting a composite marker of AprilTags on the top of WAM-V. The relative position between WAM-V and UAV is important to fuse observations of detection results.

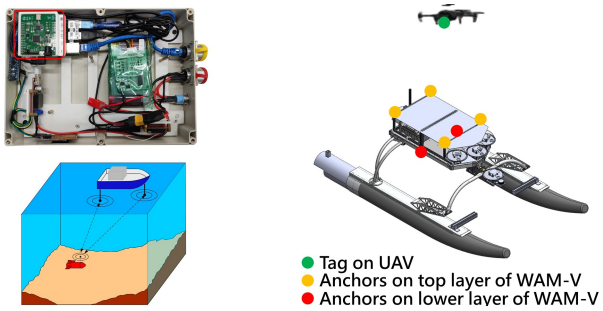
I. Short baseline (SBL) UWB Localization between UAV and USV

Given the uncertainty and insufficient precision of GPS localization, we use Ultra-wideband (UWB) localization in a short baseline (SBL) fasion. SBL has been used in the navigation and localization of autonomous underwater vehicles



(a) (b)

Fig. 10: Drone with soft gripper overview.



(a) (b)

Fig. 11: (a) Top: the anchor module. Bottom: short baseline (SBL) method has commonly used with acoustics for localizing autonomous underwater vehicle (AUV). Image courtesy of Liam Paull. (b) The configuration of 6 anchors on WAM-V and the tag on UAV.

(AUVs). In SBL, acoustic range sensors on a support vehicle with access to GPS (e.g., a boat) are used to localize the scout vehicle (AUV). In our work, we implemented SBL-UWB to obtain the relative location of the UAV and WAM-V. Such SBL-UWB method has been used in our previous work in [4]. Compared to other robotics applications, there have been several work reporting UWB tag with inertial measurement units (IMU) within an indoor environment with pre-deployed stationary UWB anchors.

UWB is a low-power ranging sensor commonly used for indoor localization. We have developed an integrated "anchor" device including Pozyx UWB module, shown in Fig. 11. We include 6 anchors as reference points on the WAM-V and one on the UAV as "tag". Pozyx UWB can range up to 30 meters and has accuracy up to 10-30 cm within 10m range through trilateration. A 3D position can be determined by performing ranging with four or more known reference points. Though four anchors satisfy the minimum requirements for 3D localization, we install six anchors on our WAM-V to improve the positioning performance, shown in Fig. 11. The four anchors on the top layer of WAM-V are enough to localize UAV, and two more anchors on the lower layer of WAM-V are added to improve the performance.

IV. Software Developments

A. Design Considerations and Objectives

The requirements of the object detection and deep RL algorithms are:

- Robust sim-to-real performance;
- Minimum human labelling;
- Multi-dimensional action space involving both navigation and arm movement commands;
- Sample efficient, requiring minimum training time.

B. Sim-to-real Perception

Data collections in real world under different weather conditions are challenging, and the generations of ground truth labels are time consuming. We develop a virtual synthesis environment automatic labelling tool in Unity simulator. The choice of using Unity over other simulation platform is that Unity provides variety of effects including realistic illumination rendering from different angle, different sky textures, and water surface reflections or waves. Those features are ideal for collecting a dataset of diverse and realistic images to train a learning-based object detection model. Fig.12 demonstrates some samples collected from different scenes.

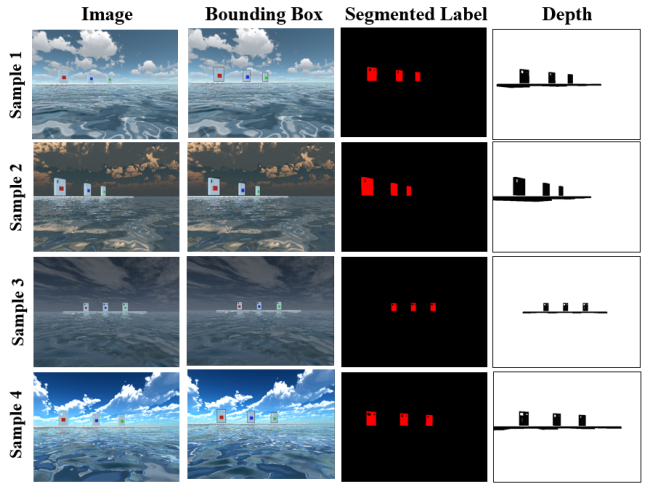


Fig. 12: Several scenes are synthesized with automatic labelling tool developed in Unity simulator.

We collect a set of virtual objects via CAD modeling or 3D reconstruction scanning, such as the docking platform. Our data generation pipeline first include 5 virtual scenes with different weathers (cloudy, sunny, and etc). We then spawn the virtual objects in the virtual environments; each object or its parts can be set independently as a segmentation label. Next, we define a camera view trajectory for collecting observations from different perspectives. Finally, a Unity script is configured to record raw image, segmented image, and depth image, respectively, shown in Fig. 13. Initially we collect 5,000 training images and train on a EfficientDet D2 model [9] similar to the settings in our previous work on underwater object detection [10].

We collect our data in the simulator for training and in the real world for testing. We set up a virtual world via Unity and Gazebo (Section IV), and a real-world testing environment similar to the Maritime RobotX Challenge 2022 in the artificial lake of NYCU campus. We split training and testing data in a ten-fold manner to investigate sim-to-real gap, and verify our perception models by deployment in real-world tasks. Taking the dock and deliver task for example, we found that 1) labelling only the color block yield poor results (mAP is around 0.2), 2) viewing angles and distances should be carefully considered, and 3) sim-to-real effectiveness is yet to discovered with real-world testing data.

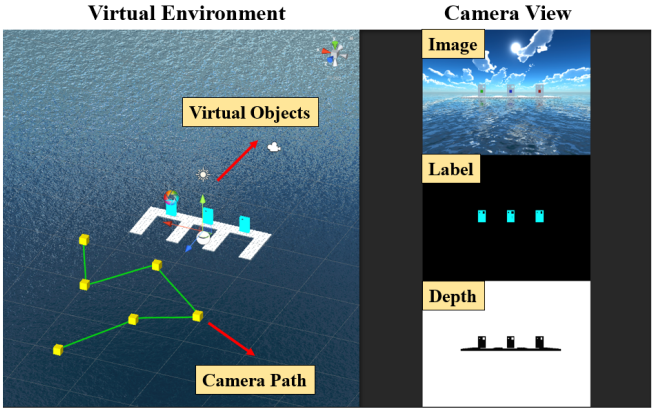


Fig. 13: Dock and deliver task setup in Unity: Left: virtual environment setting with a camera view trajectory. Right: generated raw, segmented and depth images.

C. Learning-based Collision Avoidance

Our team has developed deep RL algorithms in several of our previous research using UGV [4], [5], [11]. We wish to tackle the autonomy challenge of the model generalization from trained for UGV to the use of USV.

1) Training Environment Settings: We leverage the uses of VRX Gazebo environment [1] by selecting a forest environment environment in [12], which is an open space with standalone obstacles. We mimic the obstacle distributions and configured obstacles in VRX environment. Our goal is to compare the deep RL models trained for UGV (such as RL_{cave} and RL_{forest}) against the ones trained in the VRX environment.

2) Vehicle Dimension Settings: Given that the vehicle dimensions are different between the UGV and USV, we reshape laserscan ranges based on the length and width of the vehicle.

3) Reinforcement Learning Settings in VRX:

- Observations: We convert a three-dimensional LiDAR point cloud into a one-dimensional laser scan, with a total of 241 range values sampled from $-120°$ to $120°$. The resolution of laser scan is one degree. We concatenated 4 consecutive frames of range data and 10 frames in consist of relative position toward goal, robot velocity and robot angular velocity as a single observation space.

- Actions: Actions are designated as linear and angular of the twist commands. Linear actions were normalized to $[-0.5, 1]$ since the wave has inertia when the agent want to stop, and angular actions were normalized to $[-1, 1]$. To enable a smooth movement, linear and angular velocity was represented as a continuous rather than discrete variable.
- Reward: We established a dense reward function as follows: (1) toward the goal, (2) reaching the goal, (3) penalty for collision, and (4) keep moving

V. Innovative Developments for Heterogeneous Robot Fleet Management

A. PyIvP: Leverage the Use of MOOS-IvP for Core Functionalities

We develop PyIvP (Python binding of the IvP C++ code) as autonomy education materials for wider uses of the communities. MOOS-IvP includes two open source projects: MOOS and Ivp Helm for core of autonomy middleware and for multi-objective optimization between competing behaviors [13], respectively. MOOS-IvP has been known for the well-organized fleet managements, and widely used in the field of unmanned surface vehicle and underwater acoustics for years. Our system continues the supports of MOOS-ROS bridge [14] and we develop non-ROS WebSocket for cross-machine messaging.

B. Immersive VR Interface for Situational Awareness

In any search and rescue missions, situational awareness of a human supervisor over the robot fleet or human rescuers is the key to save lives. Virtual reality has been known to provide immersive experiences in the area of teleoperating robots over traditional 2D interface with camera views. Although all tasks need to be operated autonomously, we consider to include VR with our high bandwidth communction for task monitoring to improve situational awareness of human supervior at base station.

We employed a consumer-grade VR device (Oculus Quest 2; $420) in fabricating an easy-accessible. The Oculus Quest 2 system provides a head-mounted display which comprises a singular fast switching LCD panel with a refresh rate of 120 Hz and a resolution of 1832 × 1920 per eye. Our VR monitoring system is developed in Unity 3D engine, which is comoommly used by VR-developers. In order to utilize ROS and Unity at the same time for robot site and VR site, ROS# [15], we utilized an open-source software library for Unity to communicate with ROS, and ROS-bridge [16], providing a JSON API to access ROS functions for non-ROS programs. The WebSocket protocols allow two-way communications between ROS and Unity data transferring. Here we carry out monitoring by transferring real-time sensor data to the human supervisor. The setting is developed in our previous work [17] on a mobile manipualtor (LoCoBot) and has been applied to maritime applications demonstrated in MOOS-DAWG2022 [18].

C. Behavior Tree

To manage a team of heterogeneous robots in multiple tasks with complex state transitions, a high modularity framework is indispensable. There have been some developments using state machine, such as the SAMCH package. Although the classic state machine approach, Finite State Machine (FSM), has the upper hand in simple state transition and also easy to understand or design, the trade-offs are modularity and reactivity, which is inevitable owed to one-way control transfer structure in FSM. This cons will be out of control when the scale and complexity of the tasks increase.

Behavior Tree (BT) is a way to structure the transition between different tasks as an autonomous agent. Although BT was developed long ago in the computer game industry in order to control non-player characters, it has been applied to AI and robotics fields with tremendous success due to its *modularity* and *reactivity*. BT uses two-way control transfer governed by the different types of nodes in tree structure. BT is often compose of subtrees that are independent between the other subtrees. Each behavior in all of the subtrees can be designed in a modular way while the designer does not need to care about how it relate to subsequent behaviors in the whole BT scope. Thanks to this property, BT is able to build from a small set of simple components and all of the simple components can be designed and tested individually.

On top of that, the interaction between the nodes even provides the reactivity. The whole BT is always following the logic created by the internal nodes and leaf nodes, shown as follows:

- **Control flow nodes** define the logic and priority among all of the behaviors.
- **Condition nodes** are responsible to trigger the BT while condition changing, as well execute the corresponding actions for the purpose of triggering next condition changing. The process repeats until BT reaches a stable condition, which also meets the designer's expectation.
- **Action nodes** execute the robot actions such as navigation and pattern block exploration.

We summarize the number of nodes and list three kinds of reused subtrees in Table II. The WAM-V behavior subtrees include navigation via deep RL policy, heading, and exploration; UAV behavior subtrees include navigation and exploration. The higher values of the numbers of condition operating subtrees and nodes show more complex tasks.

VI. System Performance

A. Core Functionality: Collision Avoidance

1) Metrics: Collision avoidance was evaluated according to following metrics. *Success rate* showed that the robot successfully reached the goal. *Collisions* represented the robot collided on obstacles, we calculated the average collision times of a route. *Timeout* represented the robot got trapped and failed to reach the goal in 10 minutes. We evaluated each method 5 times for every task.

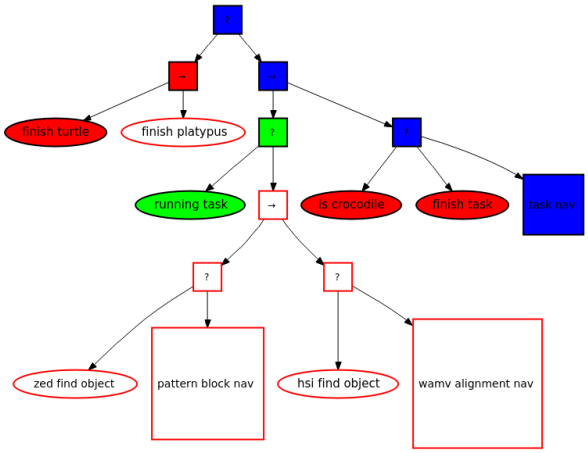


Fig. 14: Behavior tree for wildlife encounter task. Ellipse: condition nodes; small rectangle: condition flow nodes; large rectangle: action nodes.

TABLE II: Summarization of how often the subtrees are reused and numbers of nodes in all tasks.

Tasks	WAM-V subtrees	UAV subtrees	Condition operating subtrees	Control flow nodes	Condition nodes	Action nodes
T2: Entrance Exit Gate	6	-	3	16	10	8
T3: Follow the Path	1	5	4	24	14	12
T4: Wildlife Encounter	3	-	2	8	7	3
T5: Scan the Code	2	-	1	5	4	3
T6: Detect and Dock	1	-	1	1	1	1
T7: Find and Fling	2	-	1	5	4	3
T8: UAV Replenishment	-	4	8	28	17	13
T9: UAV Search and Report	-	3	5	12	7	6

2) Experiment Designs and Evaluation Environments: We evaluated policies trained in different training environments. The methods included:

- RL_{cave}: The trained from scratch in the *Cave* environment [19] for an UGV. The policy tended to follow walls while avoid obstacle.
- RL_{forest}: A two-stage trained deep RL navigation policy with RL_{cave} weights and finetued in the *Forest* environment [12] for UGV.
- RL_{vrx}: A deep RL policy trained in VRX environment considering vehicle dynamics in maritime environments for USV.

The results were shown in Table III. The proposed RL_{vrx} outperformed RL_{forest} and RL_{cave} with less collision and shorter completion time during both routes. The number of collisions of the deep RL policies trained for UGV were higher, likely due to the fact that the models were incapable of making turns early while encountered obstacles. Both UGV policies required slow down the navigation speed to complete the task, and therefore the complete time was longer.

TABLE III: Collision avoidance performance of different deep RL models.

Model	Success	Timeout	Avg. Coll.	Avg. Time.
Route1				
RL_{cave}	5/5	0/5	3.8	165.6
RL_{forest}	5/5	0/5	1.4	120
RL_{vrx} (Ours)	5/5	0/5	**0.2**	**33.4**
Route2				
RL_{cave}	5/5	0/5	2.8	177.4
RL_{forest}	5/5	0/5	2	85.4
RL_{vrx} (Ours)	5/5	0/5	**0**	**28.6**

TABLE IV: Evaluations of Path Following Tasks

Routes	Success	Timeout	Avg. Coll.	Avg. Time.
Route 1	5/5	0/5	0	131.6
Route 2	5/5	0/5	0.2	139
Route 3	5/5	0/5	0.2	125.6

B. Core Functionality: Pattern Block Exploration

We further examined a core functionality of collision avoidance with pattern block exploration, which was frequently used in several tasks. We set up a testing environment of 200m $\times$ 50m, divided into 20 regions of 10m $\times$ 50m each. Each region included 5 obstacles distributed randomly. The simulated WAM-V performed the RL_{vrx} policy similar to a lawn machine and passed through each region. The experiment was evaluated 5 rounds and the average completion time for one route was about 24 minutes (1,438 seconds). Our VRX policy was capable to avoid most of the obstacles (95.2%) and successfully reached each goal in all 5 rounds. The navigation results were shown in the supplementary video.

C. Path Following Task

Path following is a task in RobotX 2022 challenge. The missions required not only to avoid collision, but also to pass through between each pairs of red and green totems. Exploration unseen totems and potential sub-goal points were also required. We considered the UAV advantages of wider field of view. A behavior tree (BT) was designed to control the UAV actions of exploring, adjusting heading, and moving forward. The UAV cruised above WAM-V and explored the water surface nearby in order to detect totems.

We evaluated the task performances in three routes provided by VRX. We set a baseline approach of assuming known sub-goal points set between each pair of red and green totems, in order to evaluate the collision avoidance capability. The results are shown in Table IV. Our trained policy successfully avoided collisions and met the criteria for completing path following task.

D. Other RobotX Tasks

We have completed all behavior trees for each task. Due to the limited pages we did not include the behavior trees. The strategies of the tasks were described in the following sections. We will carry out more comprehensive evaluations.

1) Detect and Dock: In the detect and dock mission, WAM-V needed to detect the designated color and docked within the corresponding bay. The processing pipeline started with a trained EfficientDet model for object detection, mapped the detected position and projected to a 3D location, and finally performed a deep RL policy with the goal point while avoided collisions.

2) Scan the Code: Due to the precarious weather and changeable lighting conditions which may strongly impact the color of the object, we chosed to use the point clouds gathered from the LIDAR. First, for the preprocessing stage, we applied RANSAC and and a filter to remove the points from the sea level and random noises, leaving the objects remaining in the point cloud. After the pre-processing we appled a clustering algorithm (DBSCAN) to separate one object from another. We then projected the object's pointcloud to the 3D plane, and set it as a goal point for the RL agent to move close to the light buoy. We kept adjusting the angle between WAM-V and the light buoy until the camera faced the light buoy's LED directly. Finally, the region of interests (ROI) was cropped to further detected the color sequence.

3) Wildlife Encounter: First, WAM-V will do the pattern block navigation until zed camera detect the floating platform. Once we capture the floating platform, WAM-V will move toward it and and align USV with the floating platform making HSI has higher chance to scan the coating on the platform. Our HSI camera is installed on a robot arm. It can change the camera elevation to scan a trapezoid area. After HSI scan finish, the USV start to complete specify path correspond to platypus, turtle and crocodile.

VII. Acknowledgement

This work was supported in part by the Ministry of Science and Technology (MOST) of Taiwan under Grants MOST 110-2634-F-009-022, and MOST 109-2634-F-009-027, and in part by Qualcomm Taiwan University Research 2020 Program. We also appreciate the supports and suggestions from industrial partners or sponsors including Lungteh Inc., K-Best Inc., AMD-Xilinx, and XYZ Robotics.

References

[1] B. Bingham, C. Aguero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar, "Toward maritime robotic simulation in gazebo," in *Proceedings of MTS/IEEE OCEANS Conference*, Seattle, WA, October 2019.

[2] N.-C. Lin, Y.-C. Hsiao, Y.-W. Huang, C.-T. Hung, T.-K. Chuang, P.-W. Chen, J.-T. Huang, C.-C. Hsu, A. Censi, M. Benjamin *et al.*, "Duckiepond: An open education and research environment for a fleet of autonomous maritime vehicles," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7219–7226.

[3] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang *et al.*, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1497–1504.

[4] C. Lu, C. Huang, J. Huang, C. Hsu, P. Chang, Z. Ewe *et al.*, "A heterogeneous unmanned ground vehicle and blimp robot team for search and rescue using data-driven autonomy and communication-aware navigation," *Field Robotics*, 2022.

[5] J.-T. Huang, C.-L. Lu, P.-K. Chang, C.-I. Huang, C.-C. Hsu, P.-J. Huang, H.-C. Wang *et al.*, "Cross-modal contrastive learning of representations for navigation using lightweight, low-cost millimeter wave radar for adverse environmental conditions," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3333–3340, 2021.

[6] C.-L. Lu, Z.-Y. Liu, J.-T. Huang, C.-I. Huang, B.-H. Wang, Y. Chen, N.-H. Wu, H.-C. Wang, L. Giarré, and P.-Y. Kuo, "Assistive navigation using deep reinforcement learning guiding robot with uwb/voice beacons and semantic feedbacks for blind and visually impaired people," *Frontiers in robotics and AI*, p. 176, 2021.

[7] L. S. Yim, Q. T. Vo, C.-I. Huang, C.-R. Wang, W. McQueary, H.-C. Wang, H. Huang, and L.-F. Yu, "Wfh-vr: Teleoperating a robot arm to set a dining table across the globe via virtual reality," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[8] S. Li, J. J. Stampfli, H. J. Xu, E. Malkin, E. V. Diaz, D. Rus, and R. J. Wood, "A vacuum-driven origami "magic-ball" soft gripper," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7401–7408.

[9] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[10] F. Zocco, C.-I. Huang, H.-C. Wang, M. O. Khyam, and M. Van, "Towards more efficient efficientdets and low-light real-time marine debris detection," *arXiv preprint arXiv:2203.07155*, 2022.

[11] C.-L. Lu, Z.-Y. Liu, J.-T. Huang, C.-I. Huang, B.-H. Wang, Y. Chen, N.-H. Wu, H.-C. Wang, L. Giarré, and P.-Y. Kuo, "Assistive navigation using deep reinforcement learning guiding robot with uwb/voice beacons and semantic feedbacks for blind and visually impaired people," *Frontiers in Robotics and AI*, vol. 8, 2021.

[12] C. Cao, H. Zhu, F. Yang, Y. Xia, H. Choset, J. Oh, and J. Zhang, "Autonomous exploration development environment and the planning algorithms," 2021.

[13] M. R. Benjamin, J. J. Leonard, H. Schmidt, and P. M. Newman, "An overview of moos-ivp and a brief users guide to the ivp helm autonomy software," 2009.

[14] K. DeMarco, M. E. West, and T. R. Collins, "An implementation of ros on the yellowfin autonomous underwater vehicle (auv)," in *OCEANS 2011*. IEEE, 2011, pp. 1–7.

[15] "ROS#," https://github.com/siemens/ros-sharp.

[16] "RosbridgeServer," https://github.com/RobotWebTools/rosbridge_suite.

[17] L. S. Yim, Q. T. Vo, C.-I. Huang, C.-R. Wang, W. McQueary, H.-C. Wang, H. Huang, and L.-F. Yu, "Wfh-vr: Teleoperating a robot arm to set a dining table across the globe via virtual reality."

[18] "MOOS2022," https://oceanai.mit.edu/moos-dawg/pmwiki/pmwiki.php?n=Talk.04-Duckiepond.

[19] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments," in *Robotics: Science and Systems Conference (RSS), Virtual*, 2021.