2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Duckiepond: An Open Education and Research Environment for a Fleet of Autonomous Maritime Vehicles

Ni-Ching Lin[1], Yu-Chieh Hsiao[1], Yi-Wei Huang[1], Ching-Tung Hung[2], Tzu-Kuan Chuang[1], Pin-Wei Chen[1], Jui-Te Huang[1], Chao-Chun Hsu[1], Andrea Censi[4], Michael Benjamin[3], Chi-Fang Chen[2], and Hsueh-Cheng Wang[1,*]

Abstract— **Duckiepond is an education and research development environment that includes software systems, educational materials, and of a fleet of autonomous surface vehicles Duckieboat. Duckieboats are designed to be easily reproducible with parts from a 3D printer and other commercially available parts, with flexible software that leverages several open source packages. The Duckiepond environment is modeled after Duckietown and AI Driving Olympics environments: Duckieboats rely only on one monocular camera, IMU, and GPS, and perform all ML processing using onboard embedded computers. Duckiepond coordinates commonly used middlewares (ROS and MOOS) and containerized software packages in Docker, making it easy to deploy. The combination of learning-based methods together with classic methods enables important maritime missions: track and trail, navigation, and coordinate among Duckieboats to avoid collisions. Duckieboats have been operating in a man-made lake, reservoir and river environments. All software, hardware, and educational materials are openly available (`https://robotx-nctu.github.io/duckiepond`), with the goal of supporting research and education communities across related domains.**

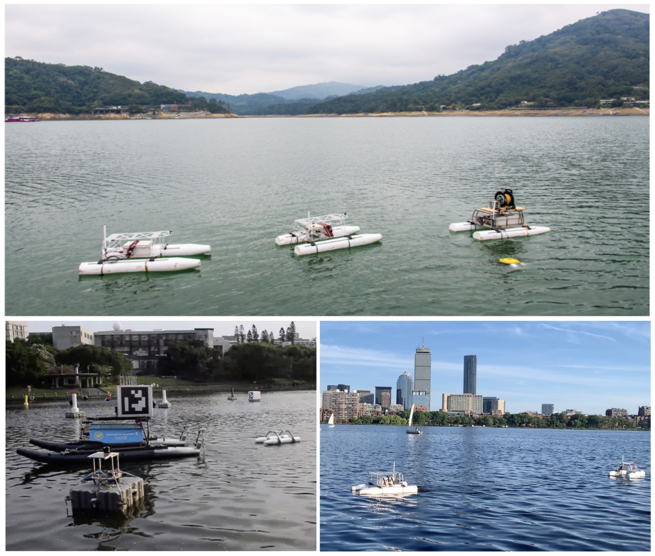


Fig. 1: Duckiepond: an open education development environment for a fleet of autonomous maritime vehicles operating in outdoor fields. The development environment includes: the hardware Duckieboat, software systems, and education materials for autonomy education. Duckieboats have been tested in a reservoir, artificial lake, and river environments.

I. INTRODUCTION

According to the US Coast Guard 4,291 accidents were reported in 2017 among 11,961,568 registered recreational vessels. Those accidents involved 658 deaths, 2,629 injuries and approximately 46 million US dollars of damage to property, and over 1,500 of reported accidents were caused by collisions to other vessels or fixed objects [1]. Self-driving vehicles have become one of the most influential applications, either on the road or on maritime domains such as harbor security. Developing such a technology may have the potential to reduce accidents due to operator inattention, inexperience, violation of navigation rules, or lack of a proper lookout - the major factors causing the reported accidents. A testbed is very much needed to make progress for the safety considerations, and is involving how to deploy the state-of-the-art technologies in a fleet of real robots to avoid collisions.

A. Development Environments for Autonomy Education and Research

Duckiepond is inspired by the *open* and *low-cost* platform Duckietown [2], [3]. The recent success of the Duckietown platform was one of the miniaturized testbed to develop autonomy education and research [2], [4]. The inexpensive platform includes a team of vehicles built upon ROS, and each vehicle includes only an onboard monocular camera and an embedded computer. A miniaturized city (Duckietown) with roads, signage, and obstacles is designed to tackle the problems of autonomous driving. The Duckietown platform started to adopt Docker and deep learning, such as Convolution Neural Network (CNN) imitation learning and deep reinforcement learning for lane following tasks, and was used in the competition of the AI Driving Olympics (AI-DO) [3], [5]. The hands-on materials have been adopted at over 10 universities globally. The materials include solutions for lane following, vehicle following, intersection coordination tasks. The materials include hardware assembly, softwares (Docker, ROS, Python, OpenCV), state estimation (Bayes' filter), computer vision and deep learning, shown in Table I.

Nevertheless, autonomous maritime missions have different challenges from ground vehicles, such as vehicle controls under unpredictable currents and winds, no road lane, and unable to apply stopping distance like cars while braking. Therefore, we wish to adapt some of the principles for the domains of the autonomous surface vehicles, and establishing

[1]Department of Electrical and Computer Engineering, National Chiao Tung University, Taiwan. [2]Department of Engineering Science and Ocean Engineering, National Taiwan University, Taiwan. [3]Department of Mechanical Engineering, Massachusetts Institute of Technology, USA. [4] Department of Mechanical and Process Engineering, ETH Zürich, Switzerland. *Corresponding author email: `hchengwang@g2.nctu.edu.tw`

TABLE I: Automony Education Materials in Duckietown and Duckiepond.

Modules	**Duckietown** [2], [3]	**Duckiepond**
Tasks	Lane Following Vehicle Following Intersection Coordination	Waypoint Navigation (M) Track & Trail Obstacle Avoidance (M)
Hardware	Duckiebot Camera RPi3 + NCS	Duckieboat Camera, GPS, & IMU RPi3 + Jetson TX2/Nano
Software	Docker, ROS, TF/PyTorch, OpenAI Gym & Pyglet [9]	Docker, **ROS & MOOS**, TF/PyTorch, **Gazebo & VRX**
State Est. CV & Learning	Bayes' Filter Edge/Color Detection Ground Projection Imitation Learning (Sim/Real) Deep RL [11]	EKF SSD [10] Imitation Learning (Sim)

(M) represents existing behaviors in MOOS-IvP.

a safe testbed called Duckiepond. Our goal is to provide the Duckiepond development environment as a further option for autonomy education and research for maritime tasks, such as waypoint navigation, track and trail, and obstacle avoidance.

For the maritime autonomy education, MIT 2.680 [6] covers software and algorithms for autonomy involving decision making. In particular the MOOS-IvP [7] autonomy software infrastructure is introduced for developing integrated sensing, modeling and control solutions. The simulation environments is complemented be a field testbed with small autonomous surface craft and underwater vehicles operated on the Charles River. There are also educational certificate programs [8] for unmanned maritime vehicles that provide students with the knowledge operating of a variety of autonomous marine survey vehicles. The program aims at covering the decision making and mission planning based on the environment factors gathered from sensor data.

B. Robotic Software and Middlewares

Mission Oriented Operating Suite (MOOS) [12] is a commonly used middleware for marine robotics, and MOOS-IvP (Interval Programming) [7] is an open source project for multi-objective optimization between competing behaviors. It is well-suited in the fields of unmanned surface vehicle and underwater acoustics, and has profound influence in marine robotics for years.

Recently the rise of Robot Operation System (ROS) [13] facilitates the development across academia, industry, and government organizations. Since there are majority of users and thousands of modules have been developed, ROS has become one of the most commonly-used middleware for robotics. Lots of robots such as PR2, Atlas, UR5, and Turtlebot use it as their software framework. There have been other middlewares developed, such as LCM and others; we refer the readers to [14].

Gazebo, a 3D simulation for general robots applications, is integrated and available to ROS users. Simulations in Gazebo have also been widely adopted in robotics education and competitions [15]–[17]. In particular the Virtual RobotX (VRX, https://bitbucket.org/osrf/vrx/src/default/) includes rich sensors, objects, and robot models, and is an open source to support developments and competitions of autonomous surface vehicles. Some tutorials are available including the qualification task in the RobotX competition [18], and a baseline solution using given GPS locations and PID controllers.

Our goal is to support two of the commonly used middlewares: ROS and MOOS (mission-oriented operating system IvP helm) [7]. In particular, we consider two major reasons to contain *both* middlewares. 1) The development environment of the education efforts should be inclusive and supporting wider communities. 2) We wish Duckiepond to well-support international competition VRX, which only provides Gazebo and ROS protocols. Although MOOS-ROS Bridge [19] has been develop to communicate these two robotics middleware, the compatibility issues remain due to various software dependencies in the *fast evolving Ubuntu and ROS distributions*. Therefore, we propose a flexible software architecture to support multiple middlewares (currently only ROS and MOOS), and is able to deploy algorithms to simulation and real robots.

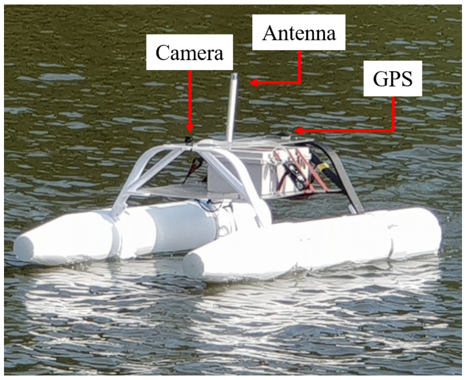
C. Developments of ASV and USV

Recent work for autonomous surface vehicles (ASV) or unmanned surface vehicles (USV) have been proposed in various applications. The OASIS platform [20] is designed to enable ocean surface study with solar power for long term operation. The Roboat platform [21]–[23] designs latching system for efficiently assemble/disassemble floating structures to quickly form floating infrastructure from multiple-vessel reconfigurable surface platform. [24] develop multiple networked mid-sized autonomous surface vehicle (ASV) for environmental monitoring. Some are extended from commercial crafts such as fishing trawler [25] and kayak [26] by mounting sensors, computing units, and propulsions. Depending on the application scenarios, the buoyance, size, sensor choices, and strength of mechanical structure of the vehicles are designed [27]–[35], including: kayak-style [26], [29], speedboat [36], catamaran [27], [31], [34], [35], small waterplane twin hull (SWATH) [30], [32]. There are also maritime robotics competitions to design various ASVs, such as the RoboBoat [37]. We refer the reader to see surveys in [38]–[42]

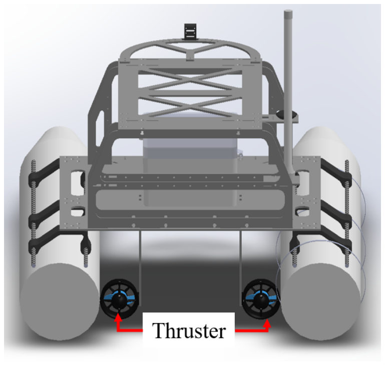
As summerized in Table II the designs of Duckieboat is after the commercially available platform Heron [43], which is a surface vehicle used for environmental monitoring (water sampling) and other tasks. Although the commercial USV have been developed, the high cost of these systems (usually several thousand dollars) has been a challenge to be widely adopted. [44] developed a low-cost platform Cooperative Robotic Watercraft (CRW), which took advantage of all sensors on a commercial smartphone, and only cost under $800. Nevertheless, the size is relatively small and not easy to reconfigure the sensors, computation, and network architecture.

D. Contributions

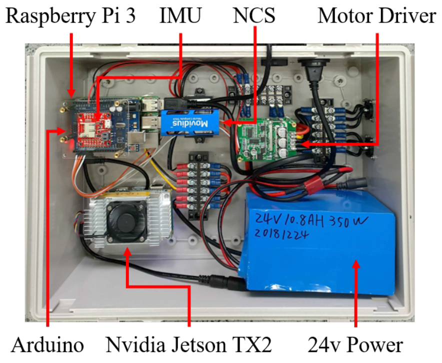
This work aims to develop an *open* education and research development environment for a fleet of ASV with essential functionalities of autonomous navigation in real-world maritime environments. The design principles include:



(a) Duckieboat



(b) Rear view of Duckieboat.



(c) Embedded computers and battery.

Fig. 2: Duckieboat designs. (a) Duckieboat includes a camera, IMU, and GPS sensors; (b) differintial drive propulsion; (c) the computing units include a Raspberry Pi3, neural compute stick, and NVidia Jetson TX2 inside a waterproof container.

TABLE II: Autonomous/Unmanned Surface Vehicles Comparisons.

Vehicles	Duckieboat	Heron [43]	CRW [44]
Dimension (m)	$1.5\times0.7\times0.5$	$1.35\times0.98\times0.32$	$0.7\times0.4\times0.4$
Weight (kg)	20	28	5.9
Payload (kg)	20	10	1
Propulsion	2 to 4 motors	2 motors	1 motor
Speed (m/s)	1.8 to 3	1.7	2.78
Power	10.8Ah Li-ion	29 Ah NiMh	10 Ah NiMh
Cost (USD)	1,600 to 2,800	>10,000	Approx. 800

- *Open Development Environment.* The Duckiboat design will be open source in hardware and software, and we wish the developed functionalities and research could be reproducible. Currently there are 4 universities (National Chiao Tung University, National Taiwan University, Seoul National University, and MIT) have adopted Duckiepond.
- *Flexible in Software and Middlewares.* With the principles of containerization (Docker), we designed and built an autonomous unmanned marine system with the software that is compatible for two commonly-used middlewares: ROS and MOOS. Enable both middlewares allow to develop under different middlewares and include establish work from wider communities, and is able to deploy on real-world and simulation environments, in particular the VRX. The flexible framework allows further integrations of other middlewares, such as LCM and ROS2.
- *Educational Materials and Benchmark.* We provide step by step tutorials include basic knowledge about autonomous surface vehicles and introduction to deep learning image processing algorithms. In the experiments we demonstrate and benchmark the performance of track and trail task that the learning-based algorithms from vision together with classic approaches via GPS.

All software, hardware, and education materials are openly available at `https://robotx-nctu.github.io/duckiepond`.

II. The Duckieboat

The Duckieboat is designed to support a fleet of homogeneous maritime vehicles. We consider the proposed Duckieboat (details in II) is capable of carrying out waypoint navigation, track and trail, and obstacle avoidance. Nevertheless, building such an autonomous unmanned marine platform for outdoor fields is challenging in many aspects. The hardware design considerations should include waterproofing, compact electronics, as well as a cooling system to handle a variety of weather conditions including heavy rain, strong winds and blistering sun. Due to the resource constrained of power and computation onboard, realtime algorithms should be adapted to *support power-hungry, computation-intensive deep neural networks* for overall performance. The design considerations should also include easily reproducible parts, such as from a 3D printer and other commercially available ones, and ideally low-cost to be widely adopted.

Table III.

TABLE III: Duckieboat is designed for low-cost.

Subsystem	**Item**	**Cost**
Computation	Jetson TX2, RPi 3B, & Arduino	\$650
Sensing	Camera, GPS & IMU	\$90
Power	Battery (4 hours operation)	\$105
Communication	2 routers & 2 antennas (Wifi: 100m; LoRa: 1km)	\$115
Propulsion	2 motors & 2 drivers	\$430
Backbone	Aluminum extrusion & lasercut acrylic board	\$1,140
Safety	E-stop, fuse & relay	\$70
Misc.	Cables, SD card, screws, nuts, insulated terminals, styrofoam ski, & waterproof connector	\$200
Total		**\$2,800**

We expect reduce the cost to **1,600** by replacing Jetson TX2 (\$599) with Jetson Nano (\$99), re-designing the backbone, and other minor improvements. Prices rounded to the closest \$5. Exact items are available at the website.

A. Sensors

The Duckieboat is designed as a low-cost autonomous surface vehicle with a camera, GPS and an IMU. The GPS antenna is set on the top plate to avoid self-occlusion and the IMU in the waterproof box. By fusing the heading data from the IMU and position data from the GPS, we obtain rough navigation information, around 1 to 4 meters precision error depending on the environments. Duckieboat uses a Raspberry Pi Camera for gaining local/short range (within

10 meters) information, in order to enable object tracking or avoid collision (see Fig. 2a).

B. Waterproof

The Duckieboat uses a watertight box (see Fig. 2a) with a waterproof connector to protect electronics from limited dust ingress and from water, except the camera outside the waterproof container. The GPS, and antenna are waterproof as well. Our design aims at IP56, and allows a Duckieboat to work in several maritime environments, such as lakes and rivers.

C. Propulsions

The Duckieboat uses two brushless DC motors in a differential drive configuration (see Fig. 2b). It could be also reconfigured using 4 motors for omnidirectional. Brushless motors are the most commonly used motor for marine robotics because of their high torque, small size, and its low electrical noise. The motors are mounted under the floating ski to provide a small radius of curvature (approximately 0.75 meter) while rotating. The Duckieboat max speed is about $1.8m/s$ come from two thrusters (from the SeaDrone https://seadronepro.com/accessories/thruster) or faster from the BlueRobotics T200 thrusters.

D. Computing Units

We use a Raspberry Pi3 embedded computer for processing GPS and IMU signals and controls, and to a Jetson TX2/Nano GPU for vision-based computing. Several deep learning algorithms have been tested such as the single shot multi-box detector (SSD) [10] and the MobileNet [45]. The Jetson TX2 is connected to Raspberry Pi3 via ethernet router. The previous version included a Neural Computer Stick (NCS) connected to the Raspberry Pi3. The NCS is a neural network hardware accelerator with USB interface designed to run the prediction/inference of deep learning algorithms. It is compatible with Caffe or Tensorflow by converting deep learning models to graph files. The Jetson TX2/Nano provide more flexibilities than the solution of NCS, including model size, specific framework, memory, and computing time.

The Duckieboat system is built on Ubuntu 16.04 and ROS Kinetic as host, and packaging software into standardized units in Docker containers [46] for development (laptop) and deployment (NVIDIA TX2, and Raspberry Pi3). Such settings ensure the software integration of required dependent libraries, machine learning tools, and the robot hardware. Based on the underlying system, we could take advantage of the packages such as Caffe, Tensorflow (TF), PyTorch to achieve learning-based capabilities.

E. Communications

The Wifi (2.4G) antenna solution provides up to 100 meters in wireless communication range, and therefore the testing environment is set to approximate $80m \times 80m$ away from the shoreside. The shoreside station which includes a computation device, a joystick, a router, and an antenna. The shoreside station serves as a master device and connects to the clients (Duckieboats). The Duckieboat has its antenna wired to an onboard client router which can communicate to the shoreside

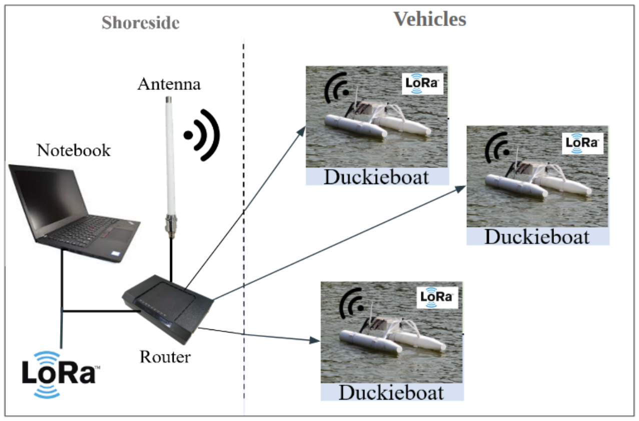


Fig. 3: Duckiepond Architecture.

router (master). It is possible to use the LoRa communication to extend the communication range to 1 kilometer with low bandwidth, shown in Fig. 3.

III. Maritime Navigation Tasks in Duckiepond

Duckiepond includes maritime navigation tasks: waypoint navigation, track and trail, and obstacle avoidance. We choose to use track and trail to demonstrate the capabilities of classic and vision-based learning approaches. The lead vehicle carried out waypoint navigation via given GPS positions, and obstacle avoidance to avoid known static obstacles such as shoreside or other floating structure in the testing environments. The follower vehicle perform track and trail and keep a assigned distance 5 meters, which is determined by vehicle speed (1.8 m/s), curvature (0.75 meter), and vehicle size (1.5 meters long).

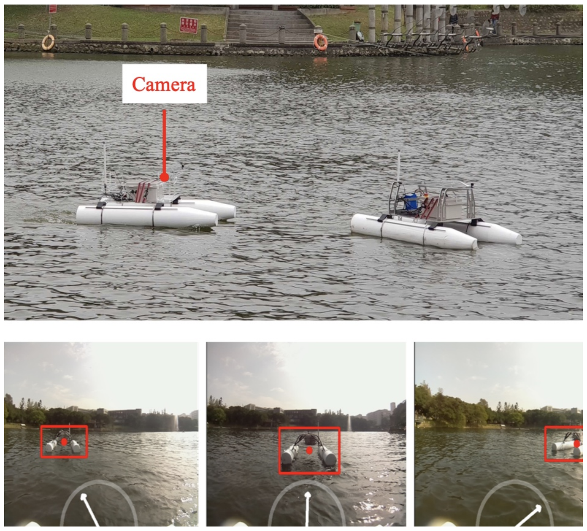
A. State Estimation and Classic Approaches

Duckiepond tutorials include navigation using PID controller and pure pursuit algorithms, as well as existing waypoint navigation using MOOS-IvP. Extended Kalman Filter (EKF) is used to fuse GPS and IMU data to filtering odometry for localization, estimating the position and orientation of egomotion. Both vehicles must communicate with each other through the shoreside station, in order to update the positions for the track and trail mission.

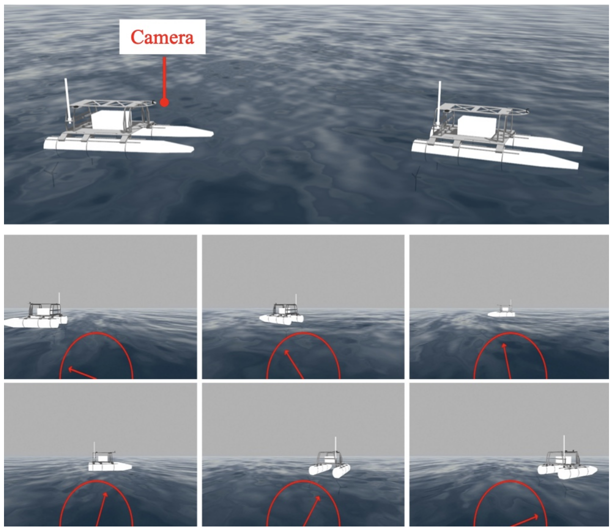
B. Object Tracking with Single-Shot Multibox Detector (SSD)

Although track and trail can be done by GPS, both vehicles require to maintain communications with a sufficient updating rate. The most intuitive way is to using the onboard camera to follow the lead vehicle. We choose the MobileNet [45] Single-Shot Multibox Detector for its efficiency. MobileNet-SSD is modified based on MobileNet referred to VGG-SSD. To train the model, we collected 3,183 images from a follower vehicle in the real environment, and manually labeled the bounding boxes of the lead vehicle in PASCAL-VOC format. We use Caffe to train our model for 30,000 iterations with a pre-trained model based on VOC datasets VOC2007 and VOC2012 [47].

The trained model is then deployed to the follower vehicle, enabling the detection of the bounding box of the lead vehicle through the camera inputs. The tracking control system is

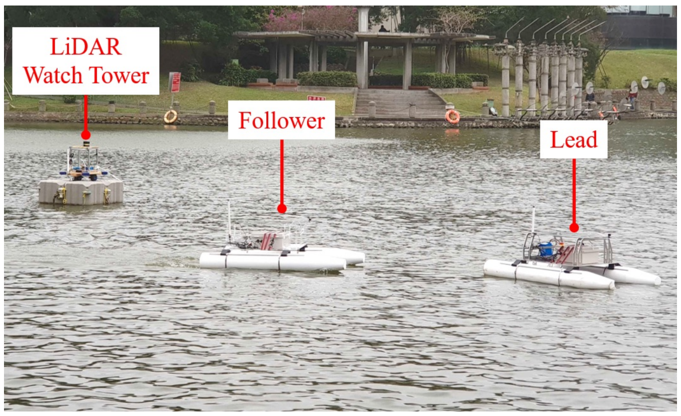


(a) Tracking the lead vehicle from SSD.

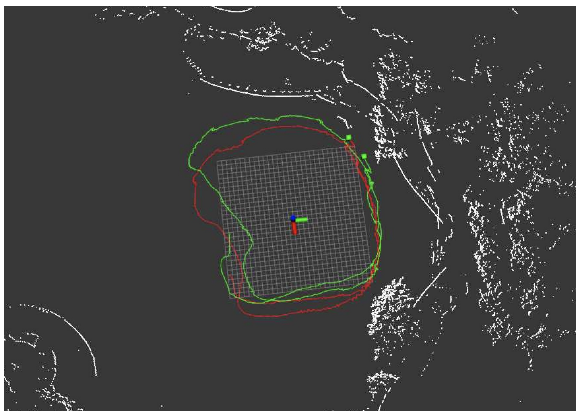


(b) Imitation learning from human demonstraction.

Fig. 4: The vision-based learning approach for track and trail task. (a) Tracking the lead vehicle using the SSD and perform control commands based on the detected bounding box. (b) Imitation learning from human demonstration in a end-to-end fasion to map input images to control commands in simulation.



(a) Experiment environments and the vehicles.



(b) Visualization of the vehicle trajectories.

Fig. 5: Experiment setup. (a) Experiment environment is around 60 $\times$ 80 meters with a LiDAR watchtower at center for localization groundtruth. (b) Visualizations of the trajectories from LiDAR watchtower for the lead (green) and the follower vehicles (red).

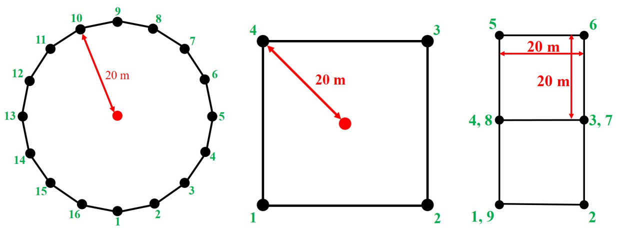
divided into two parts: angular and distance control. We define the angular error as $\frac{C_x - W_{img}/2}{W_{img}/2}$, and the distance error is defined as $\frac{K - H_{bbx}}{H_{img}}$, where (C_x, C_y) and H_{bbx} are the center point and the height of the detected bounding box, H_{img} and W_{img} as the height and width of input image, and K as the desired height of the bounding box to be pursued. The PID controller on these two systems are then used (see Fig. 4a).

C. Combined Learning and Classic Approaches

It is expected that the classic approach would cause an unstable tracking trajectory due to the sensor error of localization or updating rate, but it should complete the task as long as the GPS positions of the two vehicles are aware by both vehicle in a timely manner. On the other hand, the vision-based learning approach would smoothly track the lead vehicle, but it may fail while incorrect prediction (false positives) or the lead vehicle is out of the field of view or range (about 10 meters). Therefore, we combine both approaches and assign the weights based on the distance between the two vehicle. While the two vehicles are close to each other, the weight of learning approach increases. Otherwise the weight of classic approach increases.

D. Imitation Learning

One desired ability of robot is capable of learning from demonstration. Such approach does not require labeling bounding boxes and tracking the lead vehicle in all frames. It intends to clone the the reflex behavior and imitate human human policy for an input image, known as the end-to-end learning. Duckiepond includes educational materials of imitation learning based on the one of the AI Driving Olympics [3], [5]. In the Gazebo simulation, the "demonstration" can be done using either manual joystick control or classic approach, in order to collect the mapping for input images and control



(a) Circle-shaped. (b) Rectangle-shaped. (c) 8-shaped.

Fig. 6: Routes for track and trail experiments.

commands. By training the model, the follower vehicle can predict the steering command by input images (see Fig. 4b). We also attempt to deploy the models trained in simulation to the real robot, but we found the sim-to-real did not succeed. We plan to further train the model in a more complex/realistic simulation environment.

IV. Experiments

Track and trail task is selected for evaluations, shown in Fig. 5a, to demonstrate the capabilities: 1) the lead vehicle runs waypoint navigation or is tele-operated; 2) the follower vehicle keeps a certain distance from the lead; 3) both vehicles should avoid collisions to each other or fixed objects in the experiment site.

A. Experiment Site and Settings

The experiments took place at Bamboo Lake, an artificial reservoir in the campus of the National Chiao Tung University (NCTU), Taiwan. The experiment site is relatively still compared to river and ocean environments, but with occasional strong winds. The site is around 60 $m \times 80$ m meters. We designed 3 routes, including different numbers of waypoints and angular difference in each route, shown in Fig. 6. (a) Circle-shaped trail is based on 16 waypoints. This trail is the easiest task for track and trail because it has less angular difference and the speed of motors output should be stable. (b) Rectangle-shaped has 4 waypoints with 90° turn, considering moderate in difficulty. (c) 8-shaped has 8 90° turn waypoints and have both turns right and turn left with shorter straight route (20m), which is the most difficult route.

B. Methods

The lead vehicle is automatically operated around $1.3m/s$ through each waypoint, followed by the follower vehicle using the proposed methods: 1) a vision-based learning approach via SSD and an arbitrary controller, 2) classic approach developed in MOOS-IvP using GPS and IMU fusion, and 3) a combined method of 1) and 2), in which the control commands (velocity and angular velocity) were combined. For the safety of track and trail. We consider the accuracy of our GPS is 3 meters, and the curvature of rotation of the duckieboat is 1 meter. The safety distance for Duckieboat to go around another boat is about 2 times of the length of the Duckieboat (1.5 meters long). And the distance to generate the ideal detection bounding box by SSD is under 10 meters. Therefore, we set the track and trail distance to be 5 meters.

C. Evaluation Matrics

We record the positions of two vehicles from 1) the estimated locations from GPS and IMU fusion in each vehicle, and 2) a watchtower on an anchored development environment with a 3D LiDAR Velodyne VLP-16 is set at the center of experiment (see Fig. 5a). The point clouds from two vehicles are then clustered to estimate the positions of two vehicles. LiDAR-based evaluation is more accurate than GPS-based, although the former might be difficult in other experiment sites without an anchored floating platform.

The *trailing distances* between the lead and the follower vehicles are then measured according to the position and trajectory data, shown in Fig. 7. The ideal distance between two vehicles should maintain 5 meters but must not be shorter than 1.5 meter. We further record the number of collisions as *safety* measure, and the number of human interventions as *autonomy* measure.

D. Results and Discussions

The overall results are shown in Table IV.

1) Trailing Distance: We demonstrate that vision-based learning is well-suited for the short distance (5 meters) track and trail, given the average distance between two vehicles is the closest to 5 meters compared to the other two methods. The distance observed using the GPS-based method is larger due to the sensor error of the low-cost GPS unit. Combined vision-based learning and classic method shows a way in between: we set the follower with a higher weight to use vision-based learning method in short trailing distance, whereas a higher weight of classic method while the trailing distance is longer.

2) Safety: The learning approach is in general good to maintain a certain trailing distance, but misclassification (false detection of leader vehicle) leads to collisions. Currently the SSD method does not include the watchtower, and therefore is not able to perform collision avoidance around unrecognized objects. In classic method the GPS error and the low GPS update rate (1 update per second) may result in the follower vehicle getting too close to the lead one. More importantly, we observed unreliable network communications that caused missing GPS data, resulting in 1 collision.

3) Automony: For a more challenging 8-shaped route, we observed the need for four human interventions of the vision-based learning method, due to the lead vehicle moving out of the field of view (FOV) of the follower vehicle. The classic method from GPS can mostly maintain the autonomous. The combined method seems to be a good way for auto-recovering from vision-based learning method when there are lapses due to a limited FOV.

V. Conclusions and Future Works

We propose a development environment "Duckiepond," which include Duckieboat, software, and education materials for a fleet of autonomous maritime vehicles for education and research. The Duckieboat is designed as open and low-cost, to be widely adopted. The openly available educational materials , including Duckieboat specification, hands-on modules, and tutorials are available for hardware assembly, middleware, computer vision, state estimation, and machine learning. Our

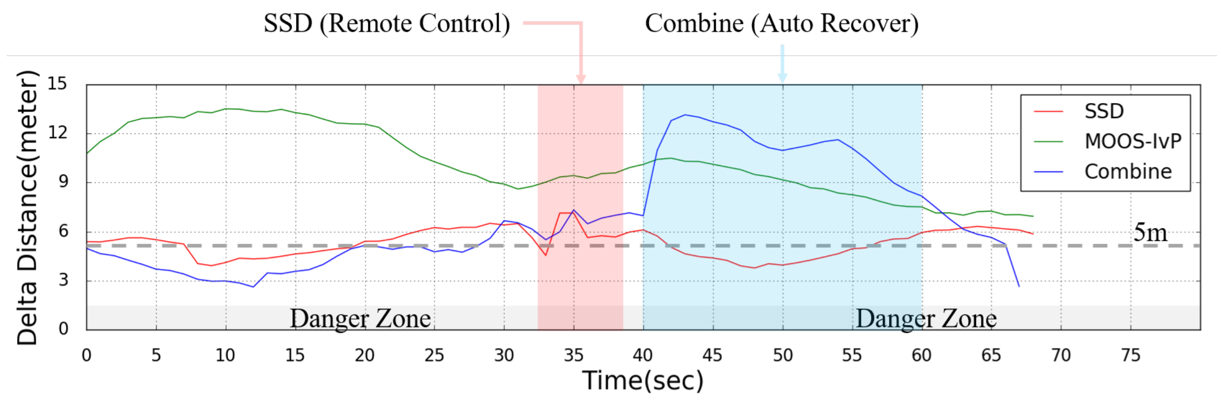


Fig. 7: The measurement of three methods trailing distances between two vehicles. Classic approach is a way far to the following distances (5 meters). SSD approach has the best trailing distance. However, there is a switch from auto to remote control because the follower vehicle can't see the lead vehicle shown in red. Combined approach could recover while the follower couldn't see the lead vehicle shown in blue.

TABLE IV: Track and trail performance in a navigation task.

Trail Routes	Learning	Classic	Combined
Circle-shaped			
No. of Trials	6	6	6
Distance (avg.)(m) GPS	**5.268**	9.999	9.309
Distance (std.)(m) GPS	**0.986**	2.972	4.485
Distance (avg.)(m) LiDAR	5.05	-	7.607
Distance (std.)(m) LiDAR	1.235	-	2.998
N of Collision	0	0	**0**
N of Manual Intervention	0	0	**0**
Rect-Shaped			
No. of trails	6	6	6
Distance (avg.)(m)	6.089	11.216	7.62
Distance (std.)(m)	4.137	3.634	1.171
Collision	0	1	**0**
Manual	1	2	**0**
8-Shaped			
No. of Trials	6	6	6
Distance (avg.)(m)	5.161	5.87	6.967
Distance (std.)(m)	2.005	1.972	2.554
Collision	1	0	**1**
Manual	4	0	**1**

experiments show the baseline performance of track and trail task using vision-based learning, classic, and combined methods. There have been 4 universities adopting Duckiepond as education or research development environment.

The Duckieboats have been tested in artificial lake, reservoir, and river. We envision that Duckieboat will work together with hetergeneous vehicle teams, such as underwater vehicles or a larger vehicle WAM-V equipped more sensors. Future work will include: 1) track and trail with underwater vehicle and more multi-vehicle behaviors with machine-learning approaches; 2) a "super Duckieboat" equipped with more sensors and advanced waterproof and self-sustained for a couple of months for river and ocean scenarios for climate observatory or bio-acoustic detection platform; 3) we plan to use Duckieboat for a large-scale environment on navigation, mapping and coordination with multiple vehicles such as harbor security. Finally, we wish that Duckiepond could be widely adopted in the communities to facilitate basic scientific research and investigate complex phenomena in nature and engineering in fields.

carry out homogeneous multi-vehicle research as well as

ACKNOWLEDGMENT

We are thankful for the inspiration by Liam Paull and Duckietown community. We are also grateful for the help of Chih-Ping Lin, Hsiao-Sheng Chen and Lim Sin Kiat. The research was supported by Ministry of Science and Technology, Taiwan (grant 107-2221-E009-156, 107-2218-E007-051, 105-2511-S-009-017-MY3).

REFERENCES

[1] (2017) Recreational boating statistics. [Online]. Available: https://www.uscgboating.org/library/accident-statistics/
[2] 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 *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1497–1504.
[3] The AI Driving Olympics. [Online]. Available: https://nips.cc/Conferences/2018/CompetitionTrack
[4] Duckietown MIT. [Online]. Available: http://duckietown.mit.edu/
[5] The AI Driving Olympics. [Online]. Available: https://AI-DO.duckietown.org
[6] Mit 2.680 - unmanned marine vehicle autonomy, sensing and communications. [Online]. Available: http://oceanai.mit.edu/2.680/pmwiki/pmwiki.php?n=Main.HomePage
[7] M. R. Benjamin, J. J. Leonard, H. Schmidt, and P. M. Newman, "An overview of MOOS and a brief users guide to the ivp helm autonomy software," 2009.
[8] Unmanned maritime systems (ums) certificate program. [Online]. Available: https://www.usm.edu/school-ocean-science-and-technology/unmanned-maritime-systems-ums-certification
[9] M. Chevalier-Boisvert, F. Golemo, Y. Cao, B. Mehta, and L. Paull, "Duckietown environments for openai gym," https://github.com/duckietown/gym-duckietown, 2018.
[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.
[11] (2019) The ai driving olympics - reinforcement learning. [Online]. Available: https://docs.duckietown.org/DT19/AIDO/out/embodied_rl.html
[12] P. M. Newman, "Moos-mission orientated operating suite," 2008.
[13] Robot Operation System. [Online]. Available: http://www.ros.org/
[14] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 4057–4062.
[15] DARPA Robotics Challenge simulator. [Online]. Available: https://bitbucket.org/osrf/drcsim

[16] DARPA Subterranean challenge (subt). [Online]. Available: https://bitbucket.org/osrf/subt/wiki/Home

[17] Virtual RobotX Challenge (VRX). [Online]. Available: https://bitbucket.org/osrf/vrx

[18] Maritime RobotX Challenge. [Online]. Available: https://www.robotx.org/

[19] 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.

[20] G. W. Podnar, J. M. Dolan, A. Elfes, S. Stancliff, E. Lin, J. C. Hosler, T. J. Ames, J. Moisan, T. A. Moisan, J. Higinbotham, and E. A. Kulczycki, "Operation of robotic science boats using the telesupervised adaptive ocean sensor fleet system," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 1061–1068.

[21] W. Wang, L. A. Mateos, S. Park, P. Leoni, B. Gheneti, F. Duarte, C. Ratti, and D. Rus, "Design, modeling, and nonlinear model predictive tracking control of a novel autonomous surface vehicle," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6189–6196.

[22] L. A. Mateos, W. Wang, B. Gheneti, F. Duarte, C. Ratti, and D. Rus, "Autonomous latching system for robotic boats," 2019.

[23] S. Park, E. Kayacan, C. Ratti, and D. Rus, "Coordinated control of a reconfigurable multi-vessel platform: Robust control approach," in *IEEE Int. Conf. Robot. Autom*, 2019.

[24] M. Dunbabin, "Autonomous greenhouse gas sampling using multiple robotic boats," in *Field and Service Robotics*. Springer, 2016, pp. 17–30.

[25] T. W. Vaneck, C. D. RODRIGUEZ-ORTIZ, M. C. Schmidt, and J. E. Manley, "Automated bathymetry using an autonomous surface craft," *Navigation*, vol. 43, no. 4, pp. 407–419, 1996.

[26] J. Curcio, J. Leonard, and A. Patrikalakis, "Scout-a low cost autonomous surface platform for research in cooperative autonomy," in *OCEANS, 2005. Proceedings of MTS/IEEE*. IEEE, 2005, pp. 725–729.

[27] H. Ferreira, R. Martins, E. Marques, J. Pinto, A. Martins, J. Almeida, J. Sousa, and E. Silva, "Swordfish: an autonomous surface vehicle for network centric operations," in *Oceans 2007-Europe*. IEEE, 2007, pp. 1–6.

[28] M. Dunbabin, A. Grinham, and J. Udy, "An autonomous surface vehicle for water quality monitoring," in *Australasian Conference on Robotics and Automation (ACRA)*, 2009, pp. 2–4.

[29] H. K. Heidarsson and G. S. Sukhatme, "Obstacle detection and avoidance for an autonomous surface vehicle using a profiling sonar," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 731–736.

[30] M. Dunbabin and A. Grinham, "Experimental evaluation of an autonomous surface vehicle for water quality and greenhouse gas emission monitoring," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5268–5274.

[31] M. Fahad, N. Saul, Y. Guo, and B. Bingham, "Robotic simulation of dynamic plume tracking by unmanned surface vessels," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2654–2659.

[32] M. Dunbabin, B. Lang, and B. Wood, "Vision-based docking using an autonomous surface vehicle," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 26–32.

[33] M. Breivik and J.-E. Loberg, "A virtual target-based underway docking procedure for unmanned surface vehicles," in *Proceedings of the 18th IFAC World Congress*, vol. 18, 2011, pp. 13 630–13 635.

[34] A. Martins, H. Ferreira, C. Almeida, H. Silva, J. M. Almeida, and E. Silva, "Roaz and roaz ii autonomous surface vehicle design and implementation," in *International Lifesaving Congress 2007*, 2007.

[35] J. E. Manley, A. Marsh, W. Cornforth, and C. Wiseman, "Evolution of the autonomous surface craft autocat," in *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No. 00CH37158)*, vol. 1. IEEE, 2000, pp. 403–408.

[36] [Online]. Available: https://maritimerobotics.com/mariner-usv/

[37] RoboBoat Challenge. [Online]. Available: https://www.auvsifoundation.org/competition/roboboat

[38] V. Bertram, "Unmanned surface vehicles-a survey," *Skibsteknisk Selskab, Copenhagen, Denmark*, vol. 1, pp. 1–14, 2008.

[39] R.-j. Yan, S. Pang, H.-b. Sun, and Y.-j. Pang, "Development and missions of unmanned surface vehicle," *Journal of Marine Science and Application*, vol. 9, no. 4, pp. 451–457, 2010.

[40] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[41] J. E. Manley, "Unmanned surface vehicles, 15 years of development," in *OCEANS 2008*. IEEE, 2008, pp. 1–4.

[42] J. Yuh, G. Marani, and D. R. Blidberg, "Applications of marine robotic vehicles," *Intelligent service robotics*, vol. 4, no. 4, p. 221, 2011.

[43] [Online]. Available: https://www.clearpathrobotics.com/heron-unmanned-surface-vessel/

[44] A. Valada, P. Velagapudi, B. Kannan, C. Tomaszewski, G. Kantor, and P. Scerri, *Development of a Low Cost Multi-Robot Autonomous Marine Surface Platform*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 643–658. [Online]. Available: https://doi.org/10.1007/978-3-642-40686-7_43

[45] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[46] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[47] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.