sensors

Article

Design and Implementation of a Low-Cost Intelligent Unmanned Surface Vehicle

Piyabhum Chaysri, Christos Spatharis, Kostas Vlachos and Konstantinos Blekas *

Department of Computer Science & Engineering, University of Ioannina, 45110 Ioannina, Greece; p.chaysri@uoi.gr (P.C.); cspatharis@cs.uoi.gr (C.S.); kostaswl@cs.uoi.gr (K.V.)
* Correspondence: kblekas@uoi.gr

Abstract: This article describes the design and construction journey of a self-developed unmanned surface vehicle (USV). In order to increase the accessibility and lower the barrier of entry we propose a low-cost (under EUR 1000) approach to the vessel construction with great adaptability and customizability. This design prioritizes minimal power consumption as a key objective. It focuses on elucidating the intricacies of both the design and assembly processes involved in creating an economical USV. Utilizing easily accessible components, the boat outlined in this study has been already participated in the 1st Aegean Ro-boat Race 2023 competition and is tailored for entry into similar robotic competitions. Its primary functionalities encompass autonomous sea navigation coupled with sophisticated collision avoidance capabilities. Finally, we studied reinforcement learning strategies for constructing a robust intelligent controller for the task of USV navigation under disturbances and we show some preliminary simulation results we have obtained.

Keywords: USV; autonomous navigation; low-cost USV platform; reinforcement learning

check for updates

Citation: Chaysri, P.; Spatharis, C.; Vlachos, K.; Blekas, K. Design and Implementation of a Low-Cost Intelligent Unmanned Surface Vehicle. *Sensors* **2024**, *24*, 3254. https://doi.org/10.3390/s24103254

Academic Editor: Sergio Toral Marín

Received: 29 March 2024
Revised: 10 May 2024
Accepted: 15 May 2024
Published: 20 May 2024

1. Introduction

As the era of artificial intelligence (AI) continues to impact several sectors of the economy and autonomous technologies continue to mature, a new era of marine industry, maritime innovation and sustainability is dawning. The growing field of small surface vehicle technology promises innovative solutions to revolutionize shipping and transport systems, with extensive research focused on unlocking benefits such as improved safety and efficiency. Within this landscape, autonomous systems have emerged as key elements in transportation infrastructure and the creation of intelligent solutions [1]. Researchers are increasingly directing their efforts towards developing control techniques, leveraging AI, in pioneering autonomous marine vehicles. The ability of AI to navigate uncertain and highly constrained dynamical systems is crucial, enabling adaptation to environmental changes and noisy conditions, and facilitating the implementation of optimized decisions.

There are plenty of benefits of making surface vehicles unmanned such as shipping flexibility, reducing costs and minimizing the impact, limitation and cost of human operators [2]. Unmanned surface vehicles (USVs) have been involved in military, research and commercial applications, including surveillance, data collection and sea, surface and space communication hubs [3–6].

However, the majority of USVs are typically large and expensive, as well as being custom-designed and built for their target application and are not generally available for purchase commercially. Existing commercial USVs are expensive and not fully based on open-source hardware, making it difficult to purchase them and to modify their designs and programming to suit various environments. Moreover, there are no available open-source USV platforms that would allow customization for varied purposes. In general, the above factors make the cost of purchasing commercially available USVs prohibitive for many laboratories and researchers.

In the past few years, the price of the components has decreased and microcontrollers, such as Arduino, are easily obtainable, which makes the construction of such vessels more accessible. There were attempts on low-cost USV [7], such as the small USV for mapping and monitoring coral presented in [8]. The cost without the camera is not so different than our proposed vessel but the operation time is limited to approximately 40 min. The use of low-cost components to enhance a manual USV with autonomous functions has also been explored in [9] to access the precision for performing bathymetric measurements. Another example of a low-cost approach for bathymetric measurements was completed in [10]. The aforementioned works utilized ArduPilot Pixhawk 1 and ArduRover 2.43 respectively for navigation purposes. Although the software is open-source, flexible and easy to use, the integration of more advanced sensors and navigation functions are still limited.

In contrast, the USV platform proposed by our team is completely open-source, from hardware to software. Some parts of the platform were 3D-printed and they can be easily modified and upgraded in terms of both design and programming. Our primary goal was to design and build a low-cost USV, where the parts can be sourced from various online and local vendors and can be replaced easily and at a low cost. This facilitates exploration of new scientific avenues and provides a flexible framework for investigating diverse and interesting research projects. Additionally, to build an easy to carry and deploy USV, we based our design on a small and light floating vessel that is widely available and inexpensive to buy. Because of this design choice, the proposed USV is suitable for applications in confined environments, like lakes or ports, where the wind and wave disturbances are limited. Possible applications could be the water quality monitoring, object localization and identification on a lake surface, seabed inspection in ports and marine life monitoring. The development of an USV for the open sea or the ocean is out of the scope of this paper.

Given our primary focus on machine learning, reinforcement learning (RL), control of autonomous marine platforms, and intelligent agents [11–13], we have already delved into the study and implementation of ship navigation algorithms utilizing RL within simulated environments [14,15], and control of autonomous robotic marine platforms [16,17]. Encouragingly, we have achieved promising results, attaining high accuracy and self-learning capabilities. We aim to continue studying and potentially enhancing these algorithms while operating in real-world settings. Additionally, we aspire to develop tailored intelligent agents to address navigation challenges specific to USVs.

Gathering together all the previous findings, we can emphasize the advantages of the proposed USV platform as follows:

- It is designed with cost-effectiveness in mind, offering a lower initial purchase cost compared to many existing models in the market. Through innovative design features and efficient energy management systems, the proposed USV minimizes operational expenses, resulting in long-term cost savings.
- It has the flexibility to incorporate cutting-edge technologies such as an autonomous navigation system and remote sensing capabilities, offering enhanced situational awareness and mission effectiveness, expanding its applicability to covering multiple needs.
- Although the system has some limitations mostly due to its size, it demonstrates adaptability and reliability across a spectrum of operational scenarios, and it covers the capability to perform significant applications, such as environmental monitoring, water quality and maritime surveillance.
- The proposed USV has lightweight construction materials allowing extended endurance, prolonged missions without the need for frequent refueling or recharging, as well as the easy integration and customization of various sensor systems.

The structure of our manuscript is as follows. Sections 2–4 present a meticulous analytical description of the fundamental buildings blocks constituting the proposed USV platform. We delve into comprehensive presentations of each component, clarifying their roles as well as their interactions with the system. Following this, we give details about the procedures required for the assembly and rigorous testing of the platform, ensuring precision and reliability in its construction. Then, Section 5 illustrates the reinforcement earning agent we have created for automatically controlling and navigating the proposed USV system. To showcase the platform's basic functionalities, Section 6 demonstrates its main capabilities in various simulated and real-world scenarios. In addition, we illustrate the efficiency of the decision support system integrated into the platform, highlighting its role in enhancing the system's autonomy behavior and adaptability in dynamic marine environments. Finally, Section 7 concludes the manuscript with findings and future directions for further study.

2. Hardware Architecture

This section outlines the hardware and components that were used in the construction of the vessel, starting with a brief outline of the components, and then discusses each of them in detail in the following subsections.

We utilized a small kayak as a base for the floating platform to simplify the building process. Subsequently, we integrated an electric motor for propulsion and mounted a servo for directional control atop it. The propulsion was an electric trolling motor for kayak to skip the waterproofing process for the thruster and simplify the building process even further. Furthermore, the control unit was a microcontroller, specifically an Arduino Mega, chosen for its versatility in communicating with both sensors and actuators. The electronic speed controller for the motor and the servo receive commands from the microcontroller.

2.1. Hull

The hull, or the main body of the vessel, has been selected within the rules of the Ro-boat Race competition which stated that the boats must be under 2.5 m in length. Our primary consideration was ensuring that the vessel was sufficiently robust to withstand the sea and waves while also adhering to the competition's regulations. After careful consideration, we opted for the Lifetime Wave, a small kayak designed for children, as it met our requirements (see Figure 1). Additionally, this kayak offers ample space to accommodate all the necessary equipment. Table 1 presents the specifications of the selected kayak.

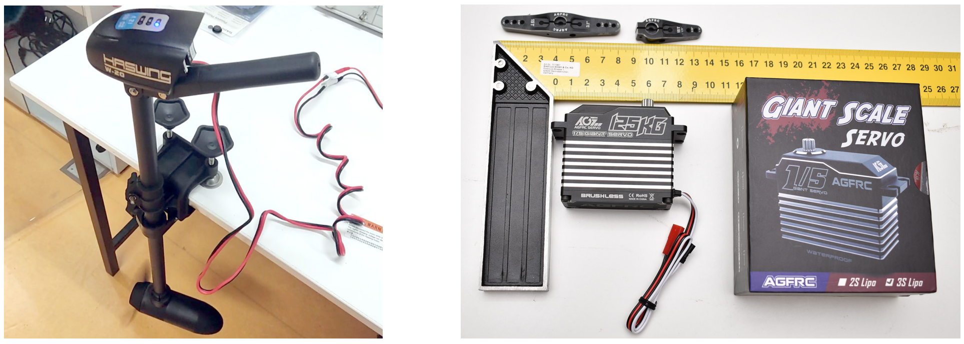
Table 1. The specifications of the kayak Lifetime Wave used in this work.

Parameter	Value
Width	61 cm
Height	22 cm
Length	183 cm
Weight	8.2 kg

Figure 1. The vehicle chassis we utilized was a simple yet durable Lifetime Wave kayak, chosen for its affordability and adaptability to our demands.

2.2. Propulsion

The propulsion is provided by an electric trolling motor and is normally powered by a 12 V deep cycle battery. The design of such motor is a thruster motor with propeller on a pole that can be attached to the back of a kayak or a small boat. We chose the Haswing W20 trolling motor manufactured by Ya Tai Electric Appliance Co., Ltd. Guangdong, China. for this work shown in Figure 2a, due to its low price and availability.



(**a**) Haswing W20 electric trolling motor; (**b**) AGFRC A280BVSW high-speed servo

Figure 2. Electric parts of the platform concerning propulsion and steering.

The steering is controlled by a high-voltage, hi-speed servo, AGFRC A280BVSW, which is shown in Figure 2b. This servo was provided to us by AGF-RC Electronic Technology Co., Ltd., Guangdong, China. at no cost. It was originally designed for 1/5 scale R/C cars and it was chosen for being waterproof (IP67 certified) and having high torque with immediate response.

The steering range was limited to 120° overall or 60° to each side. The thrust and steering range are depicted in Figure 3. The specifications of the motor and servo can be found in Table 2. By using a single thruster and a servo to propel and steer the vessel we can greatly reduce the power consumption.

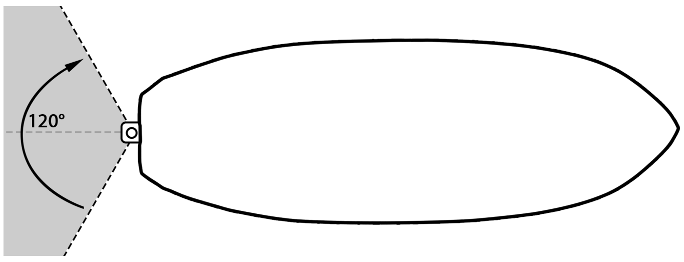


Figure 3. Steering control range of the vessel.

Table 2. Specifications of the propulsion system, consisting of a Haswing W20 electric trolling motor and AGFRC A280BVSW servo.

Parameter	Value
Motor voltage	12 V
Max thrust	20 Lbs
Power-Max	200 W
Model	W20
Prop type	2 blade prop/5.9 inch diameter
Shaft length	60 cm
Pro speed at full power	Max. 1200 rpm underwater
Servo dimensions	65.8 × 30 × 59 mm
Servo weight	284.3 g
Operating voltage	7.4 V–16.8 V
Operating travel	180° ± 10°
Operating speed	0.08 s/60° @16 V
Stall torque	125.0 kg-cm @16 V

2.3. Sensors, Computing Unit and Power

The vessel obtains its position and orientation data from GPS, gyroscope, accelerometer and magnetometer (compass). Arduino was utilized to receive and process the sensor data, which includes GPS position, angular orientation, angular velocity and linear acceleration. In addition to the positioning data, the vessel can also perceive its surroundings using a forward-facing camera, which is connected to the Raspberry Pi for image processing. A radio control transmitter was used to remotely operate the vessel as well as turning the autonomous functions on and off. The overall connection between sensors, computing unit and propulsion is depicted in Figure 4. The sensor unit consisted of a GPS, an integrated IMU sensor and a Micro SD card writer for recording the sensor data. It must be noted that the thruster was an electric DC motor that can be controlled by an electronic speed controller (ESC) from radio control vehicles. This can subsequently be controlled via PWM signal. The steering servo was also a part of radio control vehicle. Once again, this can also be controlled via PWM signal. This was to simplify the autonomous agent to control the vessel as both thruster and steering can be controlled through Arduino as well as the GPIO library on the Raspberry Pi. We chose to utilize one thruster and steering along with a low-powered computing unit in consideration of power consumption. While sacrificing speed and maneuverability, having only one engine reduced the power consumption significantly.

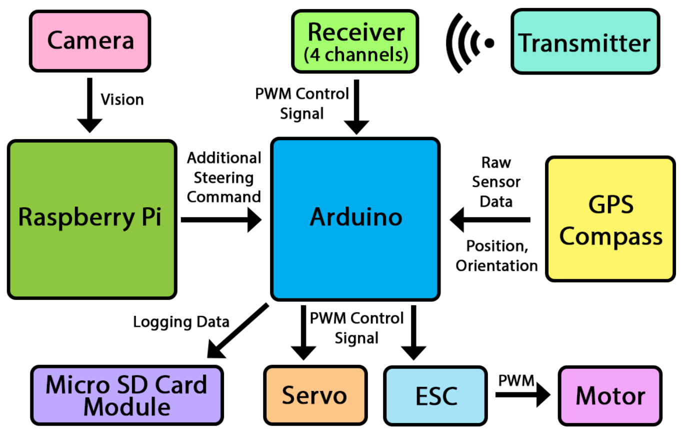


Figure 4. The connection schematic between sensors (GPS, accelerometer, gyroscope, magnetometer), Arduino, Raspberry Pi, electronic speed controller and servo.

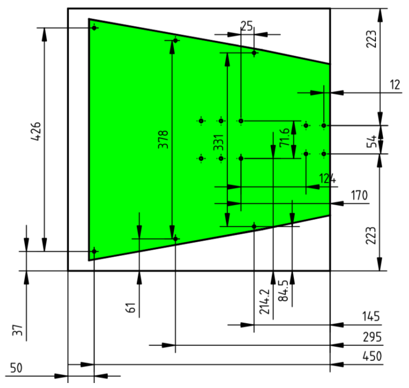
To power the propulsion and computing unit, 4-cell lithium polymer (4S LiPO) batteries were used. Each battery had the nominal voltage of 14.8 V and all batteries were connected in parallel, which allowed us to have different battery configurations between 2 and 4 batteries depending on the required operation time. We chose to increase the battery in pair because of the battery placement in order to maintain the weight distribution. LiPO was chosen due to its low cost, low weight and high energy density (6000 mAh 4S LiPO batteries). The power was provided to the computing unit through a DC-to-DC converter, a step down to 5 V 5 A.

3. Building Process

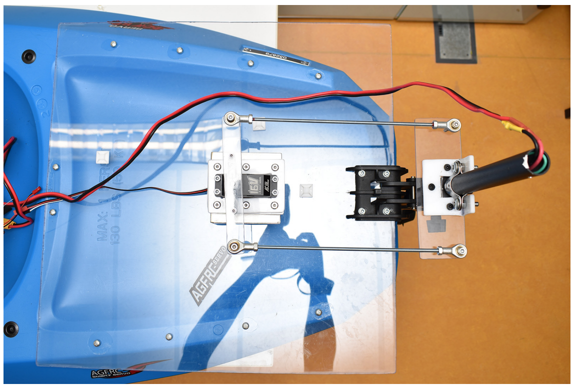
In this section we discuss the process of converting a kayak into an autonomous vehicle. To achieve this, it was necessary to securely mount the propulsion system on the hull. Additionally, the steering system required conversion from manual lever operation to a servo steering mechanism. Finally, all electronics and batteries needed to be securely attached to the hull with waterproofing measures in place.

3.1. Motor Mounting Solution

The trolling motor, or thruster motor, needed to be mounted on a flat surface. However, the area at the back of our kayak was curved, making it impossible to attach the bracket for the thruster motor. The solution wass to use a 50 × 50 cm Plexiglass sheet with a thickness of 5 mm, securely screwed onto the hull to provide a flat surface for mounting the motor bracket. We then modified the trolling motor mount by sawing off a portion to create mounting points, enabling us to attach it to the Plexiglass sheet. The design of the mounting points and the resulted motor mounting solution can be seen in Figure 5.



(**a**) Motor mounting plate design



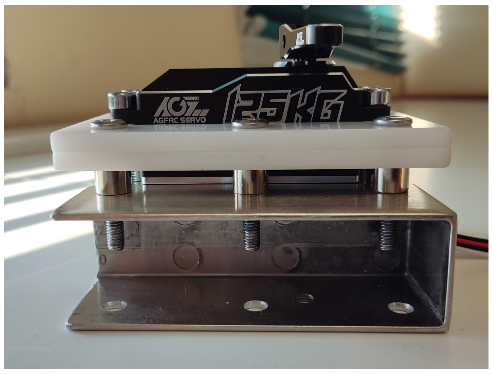
(**b**) Final result

Figure 5. The designs of the motor mounting plate with screw placements template for 50 × 50 cm plexiglass sheet (**a**) and the final result from the design (**b**).

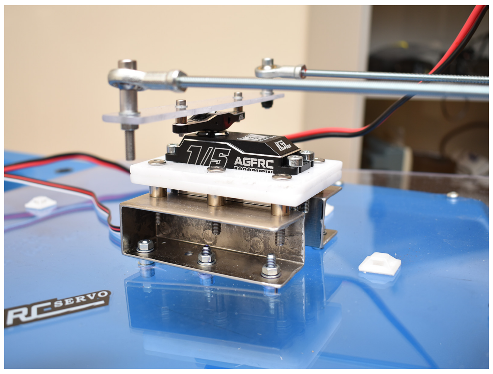
3.2. Steering Mechanism

The reason we utilized the original motor mount was to keep its pivot point intact in order to use it as a part of steering mechanism. The original steering mechanism was a shaft with locking collar to stop the motor from dropping down. This shaft can be rotated for steering using the steering handle located on the top, as shown in Figure 2a.

We converted this action from the driver physically moving the handle to using a servo to steer. We started by making the servo mounting bracket to make it sit upright. This is needed in order to hold the steering mechanism down to the plate with absolute rigidity. The picture of the servo bracket is shown in Figure 6.



(**a**)

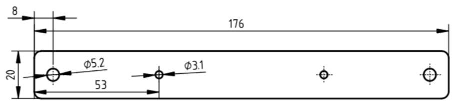


(**b**)

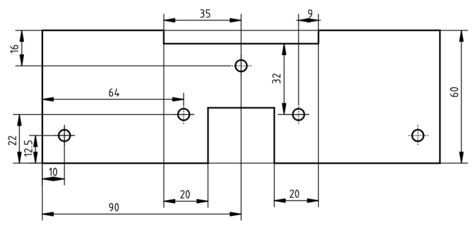
Figure 6. Servo mounting bracket (**a**) for mounting the steering system to the flat surface at the back of the vessel (**b**).

We applied the steering mechanism using two pull rods (of size M5), which were connected directly from the servo to the motor shaft. The design of the servo arm extension and the steering bracket are depicted in the Figure 7. We found that the Plexiglass or acrylic sheets provided a great solution whenever a flat surface was needed. Furthermore, once the heat was applied, the sheet could be bent to the desired angle. We used this technique for the steering part, as depicted in Figure 8a. It was then placed on the trolling motor shaft. This was later substituted with 3D printing for more durability and ease of construction.

We limited the steering range to 120° to prevent the steering mechanism from locking up at the maximum steering angle. With the motor and steering system mounted on the kayak, it was been converted to a remotely operated vehicle. The dimensions of the vessel with its engine are depicted in Figure 8b.

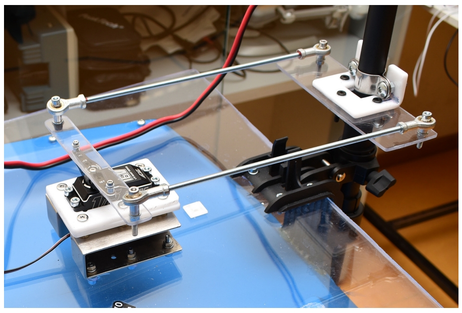


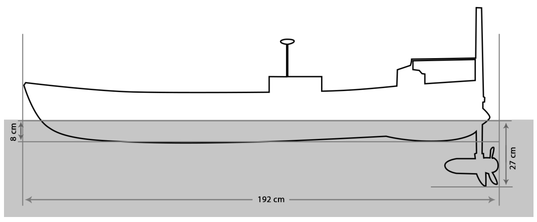
(**a**) arm extension



(**b**) steering bracket

Figure 7. The designs of the servo arm extension (**a**) and steering bracket on the motor shaft (**b**).


(**a**) The steering assembly

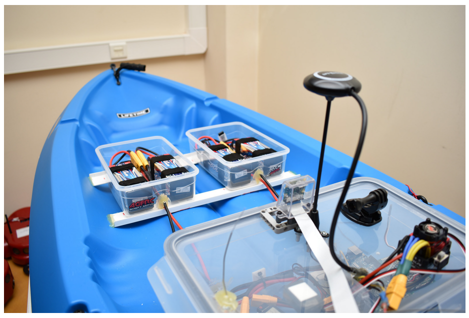


(**b**) Vessel's dimension with engine and submerged

Figure 8. The final version of the steering assembly using ball joints and the dimensions of the vessel.

3.3. Electronics Placement

The electronics and batteries were placed in water-proof boxes mounted along the craft with the consideration of the weight distribution. Their placement is depicted in the Figure 9. The boxes at the front contain batteries (see Figure 9a), while the box shown in Figure 9b contains speed control, microcontroller and sensors. As the batteries constituted the second heaviest component in the vessel, following the motor and steering mechanism, we positioned them at the front to offset the weight of the heavy rear section. Moreover, we drilled several holes on the boxes to thread the power wires in, and then we used hot glue to form the seal around the holes where the wires come out. The fully assembled vessel, along with our team members, is depicted in Figure 10.


(**a**) Battery boxes

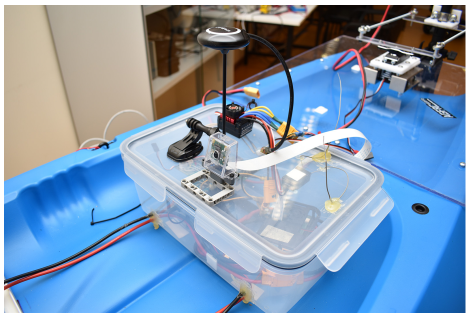

(**b**) Control unit box

Figure 9. Placing the electronics parts of the vessel.

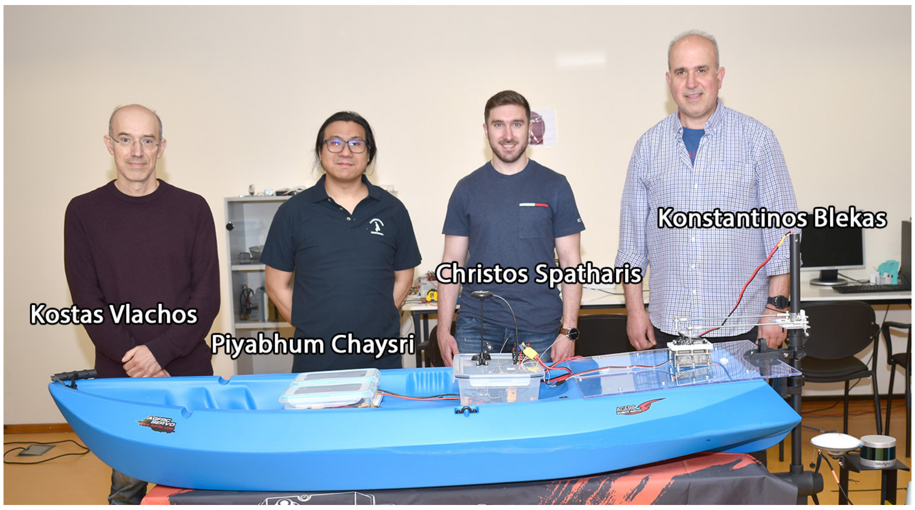


Figure 10. The fully assembled USV with our team members.

3.4. Budget

Cost was a significant concern for us, as this project did not receive support from any grants. Instead, funding for the project was provided by our own team members. Additionally, time constraints were a key consideration, necessitating the timely arrival of every component. To optimize our budget, we meticulously selected each component based on its price-to-performance ratio, taking into account factors such as its origin and estimated delivery time.

Our procurement process began by sourcing all the necessary parts from a diverse range of local and overseas vendors. In addition to the hull and propulsion system, mechanical components like ball joints were also essential to complete the build. While the kayak, trolling motor and the electronic speed controller (ESC) could be purchased within Greece, we could not find the battery with high capacity at an affordable price. Furthermore, the ball joints were also not available within Greece, hence these parts were purchased from overseas vendors to keep the spending low.

The waiting time for the parts to arrive varied depending on the vendor. Generally, products from local vendors typically arrived within 3 days, whereas orders from overseas vendors ranged between 12 to 26 days for delivery. The total cost, inclusive of shipping fees, is detailed in Table 3. It is important to note that the cost of the servo is not included in this calculation as it was generously provided to us at no cost. The current market price of the servo stands at EUR 315.86. However, for those aiming to maintain a lower budget, a substitute servo can be acquired for approximately EUR 64.

Table 3. List of the components and the total budget of the vessel.

Item	Manufacturer	Cost (EUR)
Kayak Lifetime Wave	Lifetime Products, Inc. Clearfield, UT, United States	148.8
Trolling Motor Haswing W20	Ya Tai Electric Appliance Co., Ltd., Guangdong, China	178.20
Hobbywing Quicrun WP 880 ESC	Hobbywing Technology Co., Ltd., Shenzhen, China	50
6000 mAh 4S Lipo Battery $\times 4$	Xiamen 3-circles Sports Technology Co.,Ltd., Xiamen, China	163
Plexiglass Sheets	ACRILIX AE, Thessaloniki, Greece	33.12
Nuts, bolts, ball joints, wiring, plugs and misc.	Shenzhen, China	115
Raspberry Pi 4B 8GB	Sony UK Technology Centre, Pencoed, Wales, UK	118.9
Pi Camera	Sony UK Technology Centre, Pencoed, Wales, UK	28.23
Arduino Mega	System Electronica, Scarmagno, Italy	52
Ublox NEO-M8N GPS Module	U-blox, Thalwil, Switzerland	47.03
MPU9250 sensor	SparkFun Electronics, Niwot, CO, United States	14.80
Adafruit SD Card Unit	Adafruit Industries, New York, NY, United States	11.20
Total		960.10

4. Software Architecture

The software functionalities of the proposed platform are as follows:

- Autonomous waypoint following using GPS and compass.
- Visual-based obstacle avoidance using camera.
- Remotely operated (teleop) via R/C transmitter.
- Remotely switch between manual control, waypoint following, waypoint following with obstacle avoidance and a function to skip a waypoint in case it is out of reach.

The remote control was a four-channel R/C transmitter. For the teleop function, channel 1 and 2 were used for steering and throttle respectively. channel 3 was used to activate the autonomous waypoint-following algorithm. The next waypoint could be skipped by alternating channel 3 between high and low. Finally, channel 4 was used to activate the visual-based obstacle avoidance. Next, we describe the navigation algorithm for the following waypoints.

Navigation Algorithm

The autonomous waypoint following algorithm is depicted in Figure 11, where

- ψ: the orientation of the vessel.
- $\theta_{desired}$: the desired orientation.
- θ_{error}: the difference between the vessel orientation and the desired orientation.

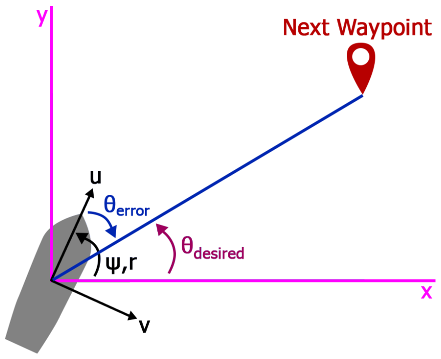


Figure 11. Navigation algorithm used in this vessel.

The vessel's latitude and longitude V_{lat}, V_{lon} were acquired via GPS, while the orientation ψ was obtained via compass. The waypoints' latitude and longitude WP_{lat}, WP_{lon} were manually entered into the program; therefore, the Arduino needed to be flashed each time a waypoint position was changed. The vessel navigated from any position to the next waypoint in a straight line and the autonomous function cycled through all waypoints and then repeated the process until it was deactivated. Each waypoint was considered completed if the distance between the vessel and the waypoint was under a threshold of 2.2 m. Algorithm 1 describes the framework for waypoint following.

Algorithm 1: Waypoint following (update rate: 2 Hz)

```
Input: Distance from the waypoint
Output: Steering and throttle command
while $Distance > Threshold$ do
    Calculate $\theta_{desired}$;
    Calculate Steering angle command with PI controller;
    Throttle = 80%;
    Send command to steering servo and speed controller;
end
Throttle = 0%;
Go to next waypoint;
```

It must be noted that in the above algorithmic structure, *Threshold* denotes the target distance from the waypoint coordinates. On the other hand, the distance function to follow the waypoints is calculated using the Haversine Formula [18]:

$$\begin{aligned} \Delta_{lat} &= WP_{lat} - V_{lat} \\ \Delta_{lon} &= WP_{lon} - V_{lon} \\ a &= \sin^2\left(\frac{\Delta_{lat}}{2}\right) + \cos(V_{lat}) \cdot \cos(WP_{lat}) \cdot \sin^2\left(\frac{\Delta_{lon}}{2}\right) \\ c &= 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{1-a}\right) \\ R &= Radius\ of\ the\ Earth \\ Distance &= R \cdot c \end{aligned} \tag{1}$$

Moreover, the desired orientation is obtained as follows:

$$\begin{aligned} y &= \sin(\Delta_{lon}) \cdot \cos(WP_{lat}) \\ x &= \cos(V_{lat}) \cdot \sin(WP_{lat}) - \sin(V_{lat}) \cdot \cos(WP_{lat}) \cdot \cos(\Delta_{lon}) \\ \theta_{desired} &= \operatorname{atan2}(x, y) \end{aligned} \tag{2}$$

Finally, the *Steering angle* is controlled by a PI controller:

$$\begin{aligned} \theta_{error} &= \theta_{desired} - \psi \\ Steering\ angle &= K_p \cdot \theta_{error} + K_i \int_0^t \theta(t)_{error}\, dt \end{aligned} \tag{3}$$

The collision avoidance utilized the SSD MobileNet V3 with COCO dataset (https://github.com/zafarRehan/object_detection_COCO (accessed on 28 March 2024)) for object detection using a camera running on Raspberry Pi4 Model B.(https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (accessed on 28 March 2024)) We chose the MobileNet for this work because we found that it was able run on Raspberry Pi at an adequate speed with good detection. It must be mentioned that we did not train our own model, but instead we utilized a pre-trained one to speed up the building process. In order to mitigate the false detection of the object, we experimented with different classes that the buoys would be recognized as. We concluded that the collision detection will only recognize three classes or categories, as follows: boat, person and airplane. The detection was executed every 2 s so that in case an obstacle was not recognized the first time, it would be in the next step. The objects detected using this method are depicted in Figure 12. The camera resolution was set to 640 × 480 and the boat was cropped out before the collision avoidance calculation process. At that stage, it was only used for the horizontal position and perceived size (occupying pixel count) of the object. Algorithm 2 presents the structure of the collision avoidance steering command framework. It has to be noted that the algorithm only considers the largest detected object in view.

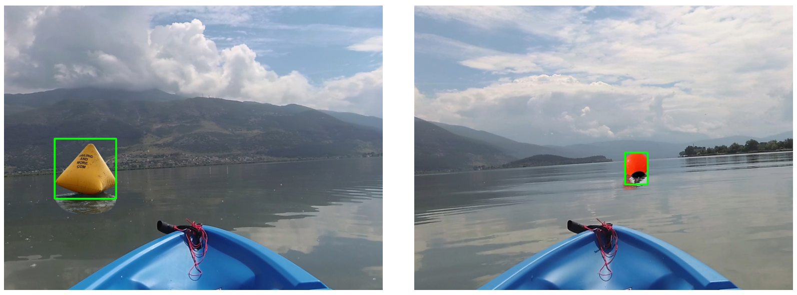


Figure 12. The view from the forward-facing camera with object detection.

The Raspberry Pi was responsible for sending the aforementioned steering command to the Arduino, which was then combined with the waypoint following algorithm, resulting in the final steering command. The diagram of the autopilot system is depicted in the Figure 13. The codes used in this work are available in the provided link (https://github.com/PiyabhumC/Ro-Boat_2023 (accessed on 28 March 2024)).

Algorithm 2: Collision avoidance (camera running at 2 FPS)

```
Input: Largest detected object's position $O_x$ and width $O_w$
Output: Steering command $[-60, 60]$
$Direction = \begin{cases} 1, & \text{if } O_x < 320 \text{ /* right turn */} \\ -1, & \text{Otherwise /* left turn */} \end{cases}$
$Steering\ angle = \max(O_w, 60)$;
return $Steering\ angle \times Direction$
```

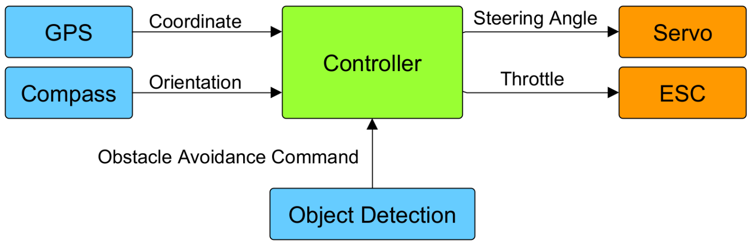


Figure 13. Autopilot system utilizes GPS, compass and extra steering signal from the object detection.

5. Reinforcement Learning Agent for USV Navigation and Control

The recent literature has introduced various methods for marine platform navigation utilizing reinforcement learning (RL) techniques [14,19]. In the realm of USVs, deep RL (DRL) methodologies have gained a lot of attention. For example, Ref. [20] used the deep deterministic policy gradients (DDPG) [21] algorithm to determine the desired heading with an adaptive sliding mechanism regulating both the heading and velocity to ensure effective path-following for the USV. Additionally, Ref. [22] introduced a DRL algorithm tailored for collision avoidance among multiple USVs operating in complex scenarios, Ref. [23] proposed a novel collision avoidance method using DRL, and [24] developed evasion strategies for USVs in multi-obstacle environments.

Generally, in RL [13], the formulation of problems often relies on the framework of Markov Decision Processes (MDPs) (Figure 14), which offers a way to model sequential decision-making problems. For instance, the navigation and control scenario examined in this study can be effectively characterized as an MDP. Specifically, the agent interacts with the environment by collecting observations (states) and performing appropriate actions for the current state. Subsequently, upon executing these actions, the environment transitions into a new state, s_{t+1}, and receives a reward, r_t. This process iterates until a terminal state is reached, and the standard RL objective is to maximize the sum of expected rewards:

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t)] \,, \tag{4}$$

where ρ_π denotes the joint distribution over states and actions based on policy $\pi(a_t|s_t)$.

Within the realm of RL, value-based methods [25] represent a framework that emphasizes on the estimation of action values over different states. Optimal policy π^* is derived by iteratively improving the value function according to the Bellman equations. Learning algorithms such as Deep Q-Network (DQN) [26] and its variations operate within this paradigm, trying to converge towards an optimal value function and using it to construct an optimal policy. Nevertheless, these approaches struggle in adapting to scenarios featuring high-dimensional or continuous action spaces. In contrast, policy-based methods focus on directly optimizing the policy function $\pi(a|s)$ to maximize the expected rewards. Learning schemes like Proximal Policy Optimization (PPO) [27], Trust Region Policy Optimization (TRPO) [28] and their variants fall under this category, leveraging gradient ascent to enhance policy parameters. Unlike value-based methods, which concentrate on policy evaluation and improvement separately, policy-based approaches continuously refine the policy through iterative updates.

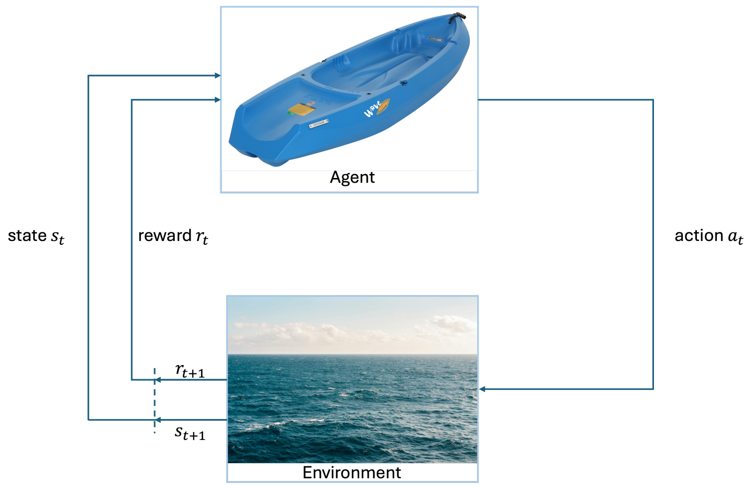


Figure 14. Markov decision process.

However, both approaches face some limitations; hence, actor-critic methods [29] have emerged as hybrid solutions in RL. These approaches combine the strengths of both paradigms to facilitate more robust learning. In such a framework, the actor network selects actions under a stochastic manner, while the critic network provides value estimates based on the actions chosen by the actor. Subsequently, the actor network updates its parameters based on the feedback from the critic network, thereby enhancing the policy over time. A state-of-the-art advancement in this domain is the Soft Actor–Critic (SAC) [30,31] algorithm, which further refines the actor–critic architecture to address issues related to sample efficiency and exploration in continuous action spaces. Leveraging an off-policy update mechanism and an experience replay, SAC addresses the challenge of sample inefficiency commonly encountered in RL tasks. Moreover, SAC introduces entropy maximization as an additional objective, promoting exploration that aids in discovering diverse strategies and contributes to the stability of the learning processes in complex environments. Formally, SAC optimizes the following objective:

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))] \,, \tag{5}$$

where α is a temperature weight of the entropy $\mathcal{H}$ of the policy that is given by

$$\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}_\pi[log\pi(\cdot|s_t)] \,. \tag{6}$$

The primary goal of SAC is to discover the optimal policy, represented as π^*, which aims to maximize the entropy-augmented objective function. To achieve this, it requires the calculation of soft Q and value functions, which satisfy the following soft Bellman equations:

$$Q(s_t,a_t) = r(s_t,a_t) + \gamma\mathbb{E}_{(s_{t+1})\sim p(s_{t+1}|s_t,a_t)}[V(s_{t+1})] \,, \tag{7}$$

where value function is obtained by

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}[Q(s_t,a_t) - \alpha\; log\pi(a_t|s_t)] \,. \tag{8}$$

We can evaluate the soft Q value of a fixed policy π by applying the Bellman equation at each time step, which is called soft policy evaluation in the soft policy iteration. This is achieved by minimizing the expected Kullback–Leibler (KL) divergence between the policy and the soft Q-function. However, applying this obective directly to large continuous state–action spaces is infeasible.

Rather than executing the policy iteration until convergence, SAC employs parameterized neural networks as function approximators for both the Q-function and the policy. Specifically, SAC comprises three networks: (a) actor network, (b) critic network and (c) value function network. The critic (or soft Q-function) network $Q_{\phi}(s,a)$, parameterized by ϕ, estimates the action-value function, while the value network $V_{\psi}(s)$, parameterized by ψ, calculates the state-value function. The soft Q-function parameters ϕ can be optimized by minimizing the squared soft Bellman residual given by

$$J_Q(\phi) = \mathbb{E}_{(s_t,a_t)\sim D}[\frac{1}{2}(Q_{\phi}(s_t,a_t) - (r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}} V_{\psi}(s_{t+1})))^2] \,. \tag{9}$$

Furthermore, SAC employs two target networks similar to the previous ones to ensure algorithm convergence.

The actor (or policy) network $\pi_{\theta}(a|s)$, with weights θ, selects actions in a stochastic manner using the reparameterization trick and updates its weights by minimizing the expected KL divergence:

$$J_{\pi}(\theta) = \mathbb{E}_{s_t\sim D}[\mathbb{E}_{a_t\sim\pi_{\theta}}[\alpha log \pi_{\theta}(a_t|s_t) - Q_{\phi}(s_t,a_t)]] \,. \tag{10}$$

First, in our study of integrating an RL agent into a navigation and control task, we defined the state space, which encompasses several crucial factors, including the linear velocity of the USV and its orientation. Additionally, the distance between the USV and the goal, along with the angle of the goal relative to the USV's orientation, provide essential spatial context for the navigation task. Moreover, we incorporate two proximity sensors indicating the distance to the closest obstacle and the angle of the obstacle in relation to the USV's orientation, which are crucial for collision avoidance. Furthermore, environmental factors like wind velocity and its direction relative to the USV's orientation are integrated into the state space to account for external noise sources on navigation.

On the other hand, the action set represents the choices available to the USV to navigate its environment. These actions include specifying the desired velocity (in m/s), enabling the agent to regulate its speed dynamically based on the circumstances and the desired steering angle of the USV within a range of ± 45 degrees. It is worth noting that the actions are assumed to be processed by a low-level controller, ensuring precise execution of the commands issued by the reinforcement learning agent. This integration of high-level decision making with low-level control mechanisms is crucial for achieving robust and efficient autonomous navigation in complex environments.

The reward function evaluates the chosen action's effectiveness based on the current USV state. We have designed an efficient reward function that prioritizes two objectives: eliminating collisions between our USV and static obstacles, and minimizing travel time. Calculating the reward function first checks if the USV reaches a terminal state, meaning that it either arrives at its destination or collides with a static obstacle. In these cases, the agent receives a constant reward: positive for successfully reaching the goal, or negative as a penalty for collision.

In any other case, the reward function encourages the USV to reach its goal quickly, while avoiding collisions with static obstacles. If there is no obstacle ahead, the agent receives a reward illustrated by the following equation:

$$r_t = -1 + velocity - distance(s_{t+1} \longmapsto goal) \,, \tag{11}$$

that is a combination of three factors: a small penalty (-1) to discourage multiple steps, a bonus for higher velocity and a relative penalty based on the distance to the goal. If there is a static obstacle detected, the base reward remains the same, however, a punishment factor is added to the overall reward, when a static obstacle is very close. This scheme discourages the agent from getting too close. For obstacles farther away, the punishment is proportional to the obstacle's distance and the agent's velocity. The closer the obstacle and the faster the agent, the higher the penalty. This encourages cautious navigation around obstacles.

6. Experimental Results

The proposed USV and developed algorithms are extensively tested in both simulated and real-world environments. The results are presented and discussed in this section. We aim to showcase the efficacy and adaptability of our approach across various scenarios, comparing its performance in both real-world settings and simulated conditions.

6.1. Real-World Experiments

The primary experimental tests of the proposed platform were conducted in Lake Pamvotis, situated in Ioannina, in north-west Greece. More specifically, the results of our initial tests has proven that the vessel functions properly under calm weather condition. The draft measured from the tests is 8 cm and the top speed recorded from the GPS is 6.4 km/h. It is to be emphasized that the results were obtained on a fresh water lake and under calm weather condition with low wind. A photo of the test environment, in Lake Pamvotis, is shown in Figure 15a. Through the test, we discovered that the battery and electronic compartments were prone to overheating when exposed to direct sunlight. This issue was addressed by implementing heat shields to protect the heat-sensitive areas. Furthermore, the waypoint skipping function proved to be a valuable safety feature which can be served to divert the vessel somewhere else or returning it to the origin point. The main conclusions from these initial trials are the following:

- The USV was able to follow the waypoints successfully.
- Due to the pondweed presence, some waypoints had to be skipped because they were unreachable. But all the systems worked as intended.
- The achieved top speed is equal to 1.64 m/s (3.19 knots) in autonomous mode and 1.78 m/s (3.46 knots) with teleop.
- The operation time with four batteries was approximately 160 min with enough left to return to base safely.

The GPS log from the test is depicted in the Figure 15b.

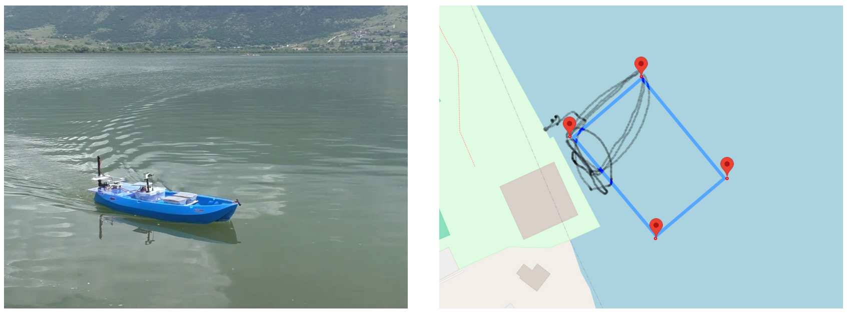
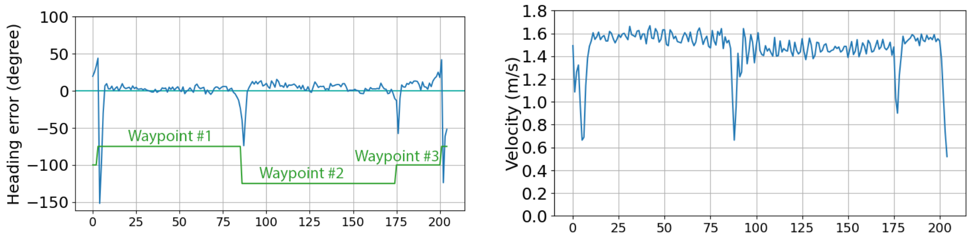

(**a**) (**b**)

Figure 15. (**a**) Trial at the lake Pamvotis, Ioannina. (**b**) GPS log from the autopilot system test at the lake Pamvotis, Ioannina. On the day of the test there were some pondweed present. Hence, other two waypoints were unreachable. This presented us with the opportunity to test the waypoint skip function.

Additionally, we participated in the 1st Ro-boat Race (Autonomous Robotic Vessels Competition) aiming to rigorously test the capabilities of our USV under various challenging conditions to assess its performance and reliability. The race took place on 12 July 2023 in Syros, Greece (https://smartmove.aegean.gr/roboat-race/ (accessed on 28 March 2024)), see Figure 16. The competition was divided into three races. The first one was the speed race. Each USV was required to traverse from the starting position then go around other two buoys and back to the starting position to complete a timed lap. The competitors went on the course one at a time. The GPS coordinates of all the buoys were given at the time of the race. We managed to achieve second place in this category. The collected data, during the race, provide us with an insight of the actions and performance of the vessel (see Figure 17). We managed to achieve a top speed of 1.67 m/s or 3.24 knots and the average speed was 1.47 m/s or 2.86 knots with little to no deviation from the shortest path available, which in this case was a straight line.

(**a**) (**b**)

Figure 16. (**a**) Sea trial in Syros, Greece before the race day. (**b**) Ro-boat race competitors on the race day, 12 July 2023.



(**a**) Heading error (**b**) Velocity

Figure 17. Data collected during the speed race competition: (**a**) heading error of the vessel, and (**b**) velocity recorded during the race in m/s.

The second race was the collision avoidance race, where all the competitors were on the course at the same time. The race was between two lines, starting and u-turn points, while the four GPS coordinates were given beforehand. The obstacle buoys were floating randomly between the lines without any prior positioning information. Each competitor had to navigate through the buoys as well as other competitors within the course. The race was one lap from the starting line to the u-turn line and back to the starting line again. The winner was the one that completed the lap the fastest with minimum contact with the obstacles. We took the first place in this category by being the fastest and without making any contact with the obstacles. The performance of this race can be seen in the log depicted in Figure 18. The steering command provided by the collision detection seemed to be erratic in the collected data. Even though the vessel managed to avoid all obstacles, this behavior was not ideal and could be improved further.

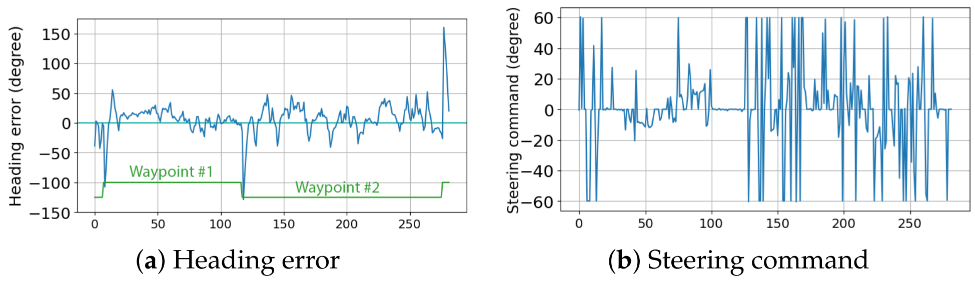


Figure 18. Data collected from the collision avoidance race category: (**a**) heading error of the vessel, and (**b**) steering command received by the collision avoidance algorithm.

The final race was the endurance race, where all the competitors were traversing between two buoys. The one that completed the most laps within the time limit would be the winner. In this category, we secured the second position by completing 19 laps within 26 min. In the data provided by log (Figure 19) the vessel managed to traverse between two waypoints repeatedly and at a constant pace. The average velocity was 1.13 m/s or 2.2 knots. This was slower than the speed race because we decreased the maximum throttle output in order to conserve the battery. All the GPS logs from the races are depicted in Figure 20.

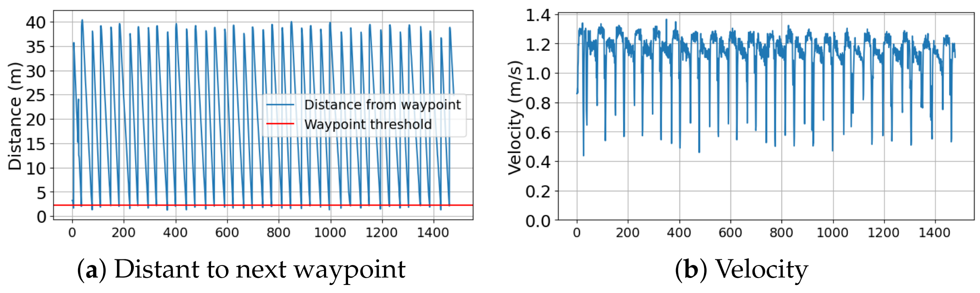


Figure 19. Data collected from the endurance race category: (**a**) distance between the vessel and the next waypoint, and (**b**) velocity of the vessel.

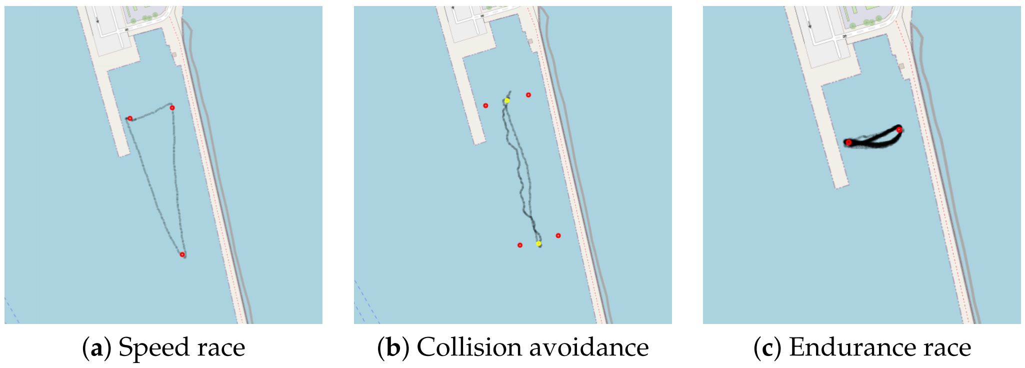


Figure 20. During the 1st Ro-boat race, we collected valuable GPS log data that provide insights into our performance and route throughout the event. The red dots in (**a**–**c**) represent the GPS coordinates given by the race organizer. The yellow dots in (**b**) are the waypoints we entered for the collision avoidance race because the provided GPS coordinates were used as markers for the start/finish and u-turn lines.

More details about the race results and the competition are published on the official website (https://smartmove.aegean.gr/events/1st-aegean-roboat-race/ (accessed on 28 March 2024)). Our final ranking was second place. Additional materials from our participation in this competition, such as videos and photos, can be found in the provided link (https://drive.google.com/drive/folders/18uuBA1z184QwKC_u4UmGBHdoCrk3HvhV?usp=sharing (accessed on 28 March 2024)).

6.2. Simulation Results

We evaluated the performance of the proposed reinforcement learning framework by conducting simulations in an environment closely resembling the one utilized in our previous research [15]. The simulation incorporates both the kinematic and dynamic model of the USV, along with simulated wind and wind-generated wave disturbances. In this study, we constrained the capabilities of the simulated vessel to closely match those of its real-world counterpart in terms of linear velocity. Subsequently, we replicated the geographic area of the island of Syros, Greece within the simulator, shown in Figure 21. The starting point is in the Ermoupoli's (capital of Syros island) swimming basin and the goal is at the main port area. The prevailing wind direction is Northeast or 225 degrees.

Experiments were made following two scenarios concerning alternative levels of wind velocity: low-wind condition with wind velocity equal to 4 knots and high-wind condition with 7 knots. The purpose of these experiments was to train the USV to navigate out of the designated swimming area and traverse the open sea to reach a specified goal on the opposite side, all while contending with various environmental disturbances.

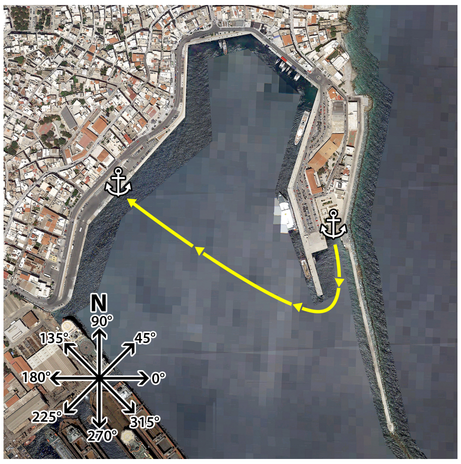
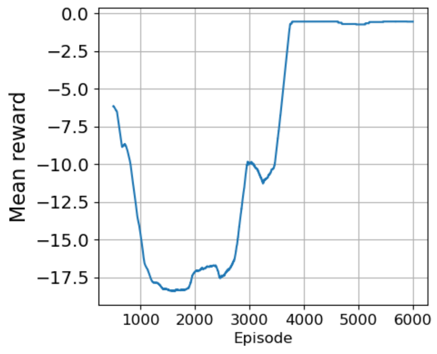


Figure 21. Map of the selected area from Syros, Greece complemented with compass angle marker, starting position and goal used in simulated environment.

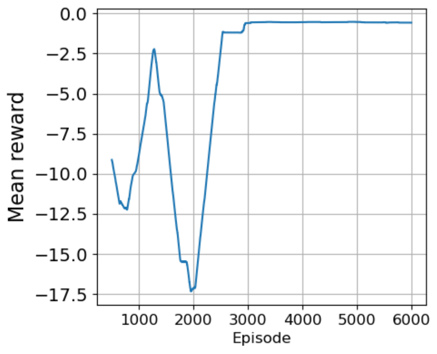
The learning curves of the reward in terms of the mean reward during navigation paths as obtained by the agent are presented in Figure 22; Figure 23 illustrates some typical navigation paths obtained following the learned policy of the RL agent in both experimental scenarios. One can observe that the generated trajectories are optimal in terms of shortened distance, proving not only the ability and efficiency of the trained agent's policy, but also its fidelity and stability.

Additionally, Table 4 presents preliminary findings from the agent operating within our simulated environment, detailing statistics gathered from 1000 executions of the agent's policy in both scenarios. The results are promising, indicating the effectiveness of our approach in enabling autonomous and secure navigation for the USV across a range of environmental conditions. The observed behavior of the agent showcases its ability for thorough exploration, adaptation to real-time fluctuations in environmental dynamics and avoidance of static obstacles.

Lastly, it is important to note that in this study, the reinforcement learning (RL) agent has not been deployed onto the physical hardware. This decision stemmed from the absence of a computing system capable of managing the computational demands necessary for the agent's operation during the competition, which was the primary purpose for constructing this Unmanned Surface Vehicle (USV).

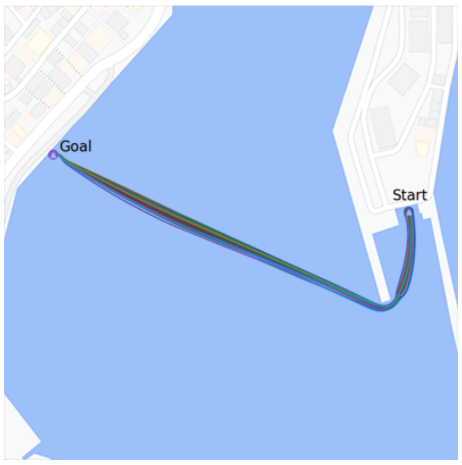


(**a**)

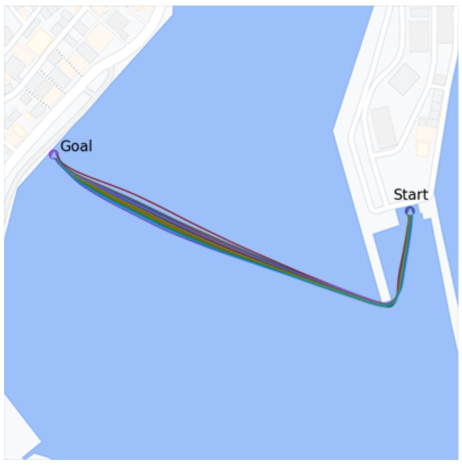


(**b**)

Figure 22. Depiction of the learning curve through the reward in rolling window of 500 episodes. (**a**) Simulated environment with 4 knots wind velocity. (**b**) Simulated environment with 7 knots wind velocity.



(**a**)



(**b**)

Figure 23. Examples of navigation paths obtained from the learned agent using several environmental disturbances. (**a**) Wind velocity 4 knots, northeast direction. (**b**) Wind velocity 7 knots, northeast direction.

Table 4. Evaluation results (mean values and stds) calculated from 1000 executions of the agent's learned policies in the simulated environment, considering two levels of wind velocity.

Wind Velocity (knots)	Success Rate (%)	Average Velocity (m/s)	Average Distance (m)
4	100 ± 0	1.71 ± 0.00	493.55 ± 2.31
7	100 ± 0	1.71 ± 0.01	491.82 ± 1.96

7. Conclusions

This manuscript presents an integrated USV system, meticulously developed within our laboratory at the Department of Computer Science and Engineering, University of Ioannina, Greece. Representing an economical solution, this platform emerges as an ideal instrument for validating concepts and algorithms in marine robotics research. Its innovative design meets the demand for cost-effective marine robotics platforms, finely tuned to the requirements of research endeavors in robotics. Notably, the system's exceptional flexibility empowers researchers with the ability to swiftly and effortlessly modify, enhance and reprogram its functionalities as required. There is plenty of space left on the hull for future expansion such as lidar and radar as well as vacant pins on the Arduino for more sensors. The USV can also take more payload which can be utilized for more batteries and better computing system for RL agent integration. This adaptability ensures that researchers can seamlessly tailor the platform to suit evolving research needs, allowing for efficient experimentation and innovation in the field of marine robotics. Furthermore, a RL agent has been developed for USV navigation and control under disturbances, which can be integrated into the presented USV. Its simulated performance proves to be reliable with excellent static obstacle avoidance capability. This could be utilized as a marine navigation algorithm for better efficiency and safety.

Finally, many concepts and methodologies employed in the proposed results, such as the reinforcement learning framework and the simulation environment, can be adapted for Networked Marine Surface Vehicles (NMSVs) [32,33], since the the underlying principles of autonomous navigation and adaptation to environmental disturbances in the case of individual USVs remain relevant. For this purpose, we can extend the proposed framework in building collaborative multi-agent systems with sharing data and coordinating actions to achieve common objectives [34,35]. Also, we can possibly consider issues about scalability and adaptability for operating in dynamic marine environments, exploring the capability of NMSVs to dynamically adjust their behavior and strategies based on real-time information exchange within the network. The above framework constitutes a possible direction for further research and a way of providing promising results.

Author Contributions: Conceptualization, K.B. and K.V.; methodology, all; software, P.C. and C.S.; validation, P.C. and C.S.; formal analysis, all; investigation, P.C. and C.S.; resources, K.B., P.C. and C.S.; data curation, P.C. and C.S.; writing—original draft preparation, all; writing—review and editing, all; visualization, P.C. and C.S.; supervision, K.B. and K.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The Arduino sketch and Python script that contain the codes for Algorithms 1 and 2, along with the logs from Ro-boat race portrayed in Figures 17–19, can be found in the GitHub repository https://github.com/PiyabhumC/Ro-Boat_2023/ (accessed on 5 March 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Rahman, A.; Hamid, U.Z.; Chin, T.A. Emerging technologies with disruptive effects: A review. *Perintis e-J.* **2017**, *7*, 111–128.
2. Qiao, Y.; Yin, J.; Wang, W.; Duarte, F.; Yang, J.; Ratti, C. Survey of Deep Learning for Autonomous Surface Vehicles in the Marine Environment. *arXiv* **2022**, arXiv:2210.08487.
3. Liu, Z.; Zhang, Y.; Yu, X.; Yuan, C. Unmanned surface vehicles: An overview of developments and challenges. *Annu. Rev. Control* **2016**, *41*, 71–93. [CrossRef]
4. Zhan, W.; Xiao, C.; Wen, Y.; Zhou, C.; Yuan, H.; Xiu, S.; Zhang, Y.; Zou, X.; Liu, X.; Li, Q. Autonomous Visual Perception for Unmanned Surface Vehicle Navigation in an Unknown Environment. *Sensors* **2019**, *19*, 2216. [CrossRef] [PubMed]
5. Bae, I.; Hong, J. Survey on the Developments of Unmanned Marine Vehicles: Intelligence and Cooperation. *Sensors* **2023**, *23*, 4643. [CrossRef]
6. Din, M.U.; Humais, A.; Akram, W.; Alblooshi, M.; Saoud, L.S.; Alblooshi, A.; Seneviratne, L.; Hussain, I. Marine$\mathcal{X}$: Design and Implementation of Unmanned Surface Vessel for Vision Guided Navigation. *arXiv* **2023**, arXiv:2311.17197.
7. Kolev, G.; Tayarani Bathaie, S.N.; Rybin, V.; Kulagin, M.; Karimov, T. Design of Small Unmanned Surface Vehicle with Autonomous Navigation System. *Inventions* **2021**, *6*, 91. [CrossRef]
8. Raber, G.; Schill, S. Reef Rover: A Low-Cost Small Autonomous Unmanned Surface Vehicle (USV) for Mapping and Monitoring Coral Reefs. *Drones* **2019**, *3*, 38. [CrossRef]
9. Specht, M.; Specht, C.; Lasota, H.; Cywiński, P. Assessment of the Steering Precision of a Hydrographic Unmanned Surface Vessel (USV) along Sounding Profiles Using a Low-Cost Multi-Global Navigation Satellite System (GNSS) Receiver Supported Autopilot. *Sensors* **2019**, *19*, 3939. [CrossRef]
10. Jaroslaw, W.; Kasza, D. Development of low-cost Unmanned Surface Vehicle system for bathymetric measurements. *IOP Conf. Ser. Earth Environ. Sci.* **2020**, *684*, 012033. [CrossRef]
11. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
12. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.

13. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
14. Blekas, K.; Vlachos, K. RL-based Path Planning for an Over-actuated Floating Vehicle under Disturbances. *Robot. Auton. Syst.* **2018**, *101*, 93–102. [CrossRef]
15. Chaysri, P.; Spatharis, C.; Blekas, K.; Vlachos, K. Unmanned surface vehicle navigation through generative adversarial imitation learning. *Ocean Eng.* **2023**, *282*, 114989. [CrossRef]
16. Tsopelakos, A.; Vlachos, K.; Papadopoulos, E. Backstepping control with energy reduction for an over-actuated marine platform. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 553–558.
17. Vlachos, K.; Papadopoulos, E. Modeling and control of a novel over-actuated marine floating platform. *Ocean Eng.* **2015**, *98*, 10–22. [CrossRef]
18. Gade, K. A Non-singular Horizontal Position Representation. *J. Navig.* **2010**, *63*, 395–417. [CrossRef]
19. Tziortziotis, K; Tziortziotis, N.; Vlachos, K.; Blekas, K. Motion Planning with Energy Reduction for a Floating Robotic Platform under Disturbances and Measurement Noise using Reinforcement Learning. *Int. J. Artif. Intell. Tools* **2018**, *27*, 1860005. [CrossRef]
20. Gonzalez, A.; Castañeda, H.; Garrido, L. USV Path-Following Control Based On Deep Reinforcement Learning and Adaptive Control. In Proceedings of the Global Oceans 2020: Singapore—U.S. Gulf Coast, Biloxi, MS, USA, 5–30 October 2020.
21. Lillicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
22. Ma, Y.; Zhao, Y.; Wang, Y.; Gan, L.; Zheng, Y. Collision-avoidance under COLREGS for unmanned surface vehicles via deep reinforcement learning. *Marit. Policy Manag.* **2020**, *47*, 665–686. [CrossRef]
23. Fan, Y.; Sun, Z.; Wang, G. A novel intelligent collision avoidance algorithm based on deep reinforcement learning approach for USV. *Ocean Eng.* **2023**, *287*, 115649. [CrossRef]
24. Qu, X.; Gan, W.; Song, D.; Zhou, L. Pursuit-evasion game strategy of USV based on deep reinforcement learning in complex multi-obstacle environment. *Ocean Eng.* **2023**, *273*, 114016. [CrossRef]
25. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
26. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; Volume 48, pp. 1928–1937.
27. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
28. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.I.; Moritz, P. Trust Region Policy Optimization. *arXiv* **2015**, arXiv:1502.05477.
29. Bhatnagar, S.; Sutton, R.; Ghavamzadeh, M.; Lee, M. Natural actor-critic algorithms. *Automatica* **2009**, *45*, 2471–2482. [CrossRef]
30. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm Sweden, 10–15 July 2018.
31. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft Actor-Critic Algorithms and Applications. *arXiv* **2018**, arXiv:1812.05905.
32. Gu, N.; Wang, D.; Peng, Z.; Liu, L. Distributed containment maneuvering of uncertain under-actuated unmanned surface vehicles guided by multiple virtual leaders with a formation. *Ocean Eng.* **2019**, *187*, 105996. [CrossRef]
33. Peng, Z.; Wang, J.; Wang, D.; Han, Q.-L. An overview of recent advances in coordinated control of multiple autonomous surface vehicles. *IEEE Trans. Ind. Inform.* **2021**, *17*, 732–745. [CrossRef]
34. Chen, Q.; Ge, M.F.; Liang, C.D.; Gu, Z.W.; Liu, J. Distributed optimization of networked marine surface vehicles: A fixed-time estimator-based approach. *Ocean Eng.* **2023**, *284*, 115275. [CrossRef]
35. Huang, K.-L.; Liang, C.-D.; Ge, M.-F.; Liu, X.; Chen, Q.; Cai, W. Finite-time multi-surrounding control of networked marine surface vehicles: A three-layer-hierarchical approach. *Ocean Eng.* **2024**, *294*, 116702. [CrossRef]