A Novel Approach to Underwater Mobility

Cymberly Tsai

Table of Contents

Abstract

Aquatic environments currently pose a challenge to the human race. Physical limits restrict human mobility underwater, leaving the field vastly unexplored with considerable room for advancement. A submersible device capable of traversing aquatic environments would expand the human ability underwater. Based on the fundamental concept of moving underwater, the project is an underwater mobility unit designed to navigate an aquatic environment and complete a simple task: detect and land on a target. The underwater mobility unit utilizes a novel maneuvering system for sinking and machine learning algorithms for searching. The machine begins at the surface of a 120 by 56 centimeter area of water. The unit will move about the area to search for the red 10 by 10 centimeter square target in a random location at the bottom of the area. Essentially, the objective of the machine is to continue searching until the target is detected, then land and turn on an LED light to signal completion. After ten trials, the unit achieved a 70% success rate of landing on the target, with three failures caused by water disturbance. Improvements to the container and turning mechanism would make the unit more stable in water, circumventing issues of water disturbance. Future experimentation could include a larger area with varying targets to further challenge underwater movement. Features of the current machine functioned as intended and could be implemented in more extensive projects with real-world applications.

Introduction

Human physical ability severely limits underwater discovery and exploration. Lack of oxygen, high pressures, and temperature extremes hinder the human ability to access underwater environments. This leaves about 80% of the ocean unexplored, yet about 70% of Earth's surface is covered by water (NOAA). Further, the wet nature of water poses an additional challenge to electronics underwater and other alternatives for underwater mobility, as exposure to moisture can cause corrosion, dielectric loss, and degradation among various other risks (Baylakoğlu et. al). To access aquatic environments, humans require a self-contained, submersible device.

Underwater vehicles can perform tasks underwater that humans are unable to physically carry out. Marine life research, such as locating clownfish and blue tangs, can be conducted by a mobile device. Such devices can also perform underwater repairs on structures such as dams, allowing hydrologic energy to become much more maintainable and without risking human life for manual repairs. An underwater machine can also save a life by traversing water with more speed and accuracy than a human ever could. Aquatic vehicles can open the possibility of colonizing the ocean, along with other opportunities humans have yet to conceive.

Engineering Goal

Create a machine with machine learning algorithms capable of underwater movement to detect a red 10 by 10 centimeter target placed within a 120 by 56 centimeter square at the bottom of a body of water, the machine will then move toward the target until the machine makes contact with the target, then the machine will stop moving and signal an attached LED light.

Materials

- Raspberry Pi 4
- Breadboard
- Motor Drivers (L293D)
- DC Motors
- Plastic Container
- Syringes
- Nut and Bolt
- LED Light
- Webcam
- Coins
- 9v Battery
- Power Bank

Python Libraries

- OpenCV
- NumPy

Fig 1. Image of machine

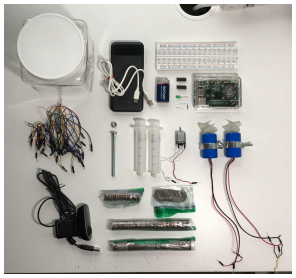


Fig 2. Image of materials

See Appendix A for circuit diagram and description.

System Overview

Overall Approach

An aquatic vehicle would ideally be autonomous and self-contained, to minimize unwanted contact with water. The machine would also need a method of descending and surfacing on demand, to be able to both sink and float. Underwater machines would also need to be able to receive and interpret some sort of input for navigation and other tasks to truly be applicable. The underwater mobility unit designed in this project aims to meet these goals.

Machine Design

All materials are contained in the container with the exception of the syringes and propellers protruding from one end. The syringes are contained in a cardboard compartment to

prevent interference with motion. The model includes coins as weight to bring the mass of the machine closer to the volume as well as aiding in balance. Protrusion openings sealed with copious amounts of ethylene vinyl acetate applied with a hot glue gun. The inside of the container is accessible through an opening and screw on cap. During experimentation, the opening is covered with three layers of plastic wrap with the cap screwed on and additional layers of duct tape to seal the container.

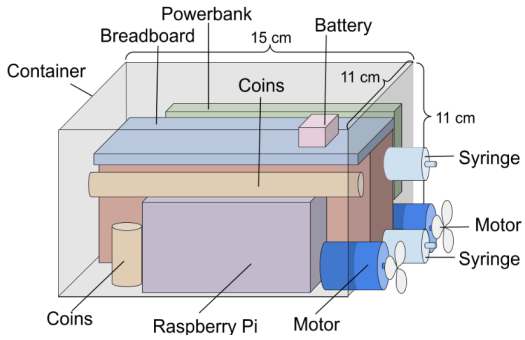


Fig. 3. Diagram of machine design

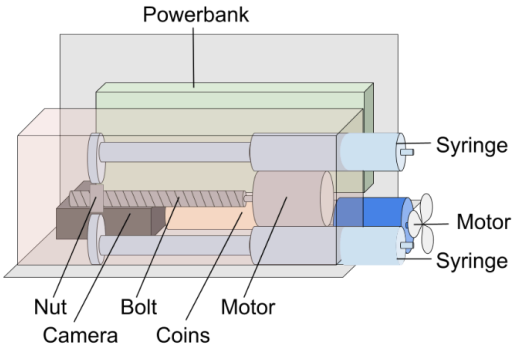


Fig. 4. Diagram of machine internal design

See Appendix B for images of the machine.

Sinking Mechanism

The machine needs to sink to achieve true three dimensional movement underwater. For an object to sink in water, the density must be greater than the density of water, which is roughly 1.0 g/ml. Given the equation $D = \frac{m}{v}$, where D equals density, m equals mass, and v equals volume, the object must lose volume or gain mass to increase the density. Rather than losing volume, the machine gains mass by intaking water available in the environment through two 20 ml syringes that protrude from the machine. The additional water increases the mass of the

machine while maintaining the volume, allowing the density to increase past 1.0 g/ml and sinking the machine. The water can then be released for the machine to resurface.

Drawing water through the syringes requires linear motion to push and pull the plungers. The rotational motion from a DC motor was translated into linear motion using a nut and bolt mechanism. The motor rotates a bolt in a fixed position with a nut attached to the plunder heads. The rotation of the bolt forces the nut up and down, and the plungers with the nut. The machine program signals for the motor to turn on for a set amount of time to fill the syringes and sink the machine.

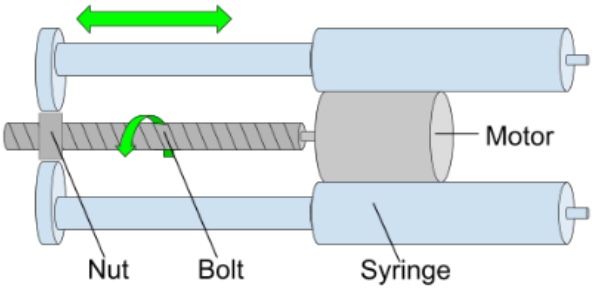


Fig. 5. Diagram of sinking mechanism

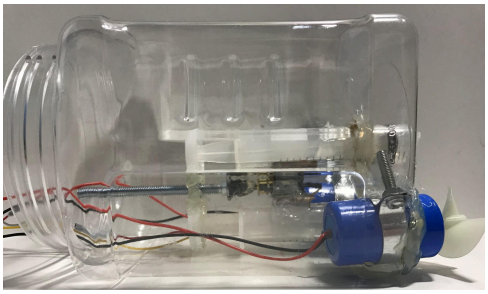


Fig. 6. Image of sinking mechanism in machine

Detection Script

The purpose of the detection script is to identify and locate the target. A camera located at the bottom of the machine periodically takes pictures of the bottom of the body of water. The images are then converted into Hue Saturation Values (HSV) using OpenCV, a python library designed for real-time computer vision using machine learning functions. Each color has a specific range of HSV values. The HSV values for red, the color of the target, are defined as an NumPy array within the program. Given the HSV values of an image, a mask is applied with an OpenCV function to filter out pixels with HSV values not within the red range. The program

then iterates through the pixels of the masked image in increments of fifty, if any red pixels are found the program will then return true and signal the machine to descend. If no red pixels are found in a given image, the machine will continue moving and searching.


Fig. 7. Original image

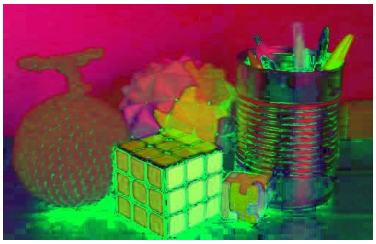


Fig. 8. HSV image


Fig. 9. Masked image

See Appendix C for partial code.

Method of Search

The ideal method of search will cover the entire area in the shortest amount of time, this means minimizing turns and maximizing straight line movement. Starting at the initial position in the corner of the surface of the area, the first best move is a straight line in the shortest direction following the border of the area. Once the length is complete, the next best move is a 90 degree turn and another straight line. This is repeated for the remaining sides until the length is shortened. The machine then moves in concentric rectangles, ending in the center of the area or until the target is found.

1. Straight line along border
2. Single 90 degree turn
3. Repeat for three sides
4. Decrease length for fourth side
5. Repeat in concentric rectangles

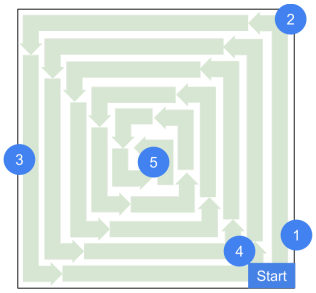


Fig. 10. Search method diagram

See Appendix D for implemented process

Procedure

1. Place target at random location at bottom of 120 by 56 centimeter area body of water
2. Place machine in close-right corner of given area at the surface of the water
3. Turn on machine, run program, and allow machine to move
4. When LED light is on, record whether or not the machine made contact with target
5. Repeat steps 1 through 4 ten times

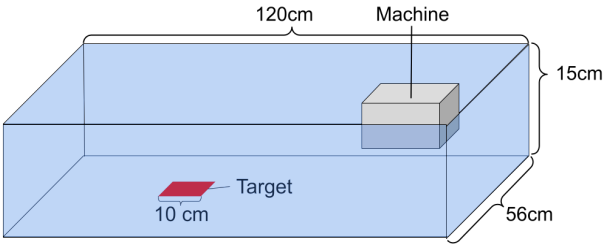


Fig. 11. Diagram of experiment setup

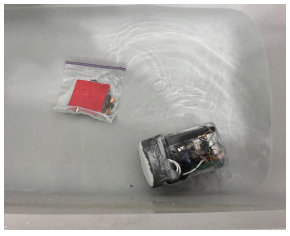


Fig. 12. Image of experiment setup

Fig. 13. Image of machine searching

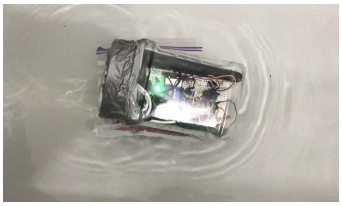


Fig. 14. Image of machine completion

Results

The machine achieved a 70% success rate, failing 3 out of 10 trials.

Trial Number	1	2	3	4	5	6	7	8	9	10
Contact with Target	False	True	True	True	False	False	True	True	True	True

Table 1. Boolean result of each trial.

Analysis and Conclusion

The model was a success with a passing success rate. The failures were caused by water disturbance from the movements of the machine that shifted the machine as it attempted to land. Another limitation of the machine was the time consumption, as each trial took 5 to 10 minutes to complete. An improved turning mechanism with stronger motors would allow the machine to move faster. A camera with wider scope would also decrease the time needed for search. The model would also benefit from a container designed for balance and stability, this would make the machine more resistant to water disturbance. With these improvements in mind, further investigation could be run in a larger area with greater depth to further test three dimensional movement. Other experimentation could also include varying targets, such as multiple or moving targets. The model was designed to achieve underwater mobility. The model used a novel sinking mechanism and machine learning functions for searching to accomplish this goal. After testing in an aquatic environment, the machine was shown to complete the goal.

References

Baylakoğlu, İlknur, et. al. “The Detrimental Effects of Water on Electronic Devices.” *Advances in Electrical Engineering, Electronics and Energy*, vol. 1, 2021, https://doi.org/10.1016/j.prime.2021.100016. Accessed 23 Feb. 2023

Harris, C.R., et al. “Array Programming with NumPy.” *Nature*. https://doi.org/10.1038/s41586-020-2649-2.

“L293D/ Motor Driver IC.” *Components 101*, 8 Oct. 2017, https://components101.com/ics/l293d-pinout-features-datasheet. Accessed 23 Feb. 2023.

NOAA. “How Much of the Ocean Have We Explored?” *National Ocean Service*, 20 Jan. 2023, https://oceanservice.noaa.gov/facts/exploration.html. Accessed 23 Feb. 2023.

OpenCV. *Open Source Computer Vision Library*, 2015.

"Raspberry Pi Documentation." *Raspberry Pi*, www.raspberrypi.com/documentation/computers/os.html. Accessed 16 Nov. 2022.

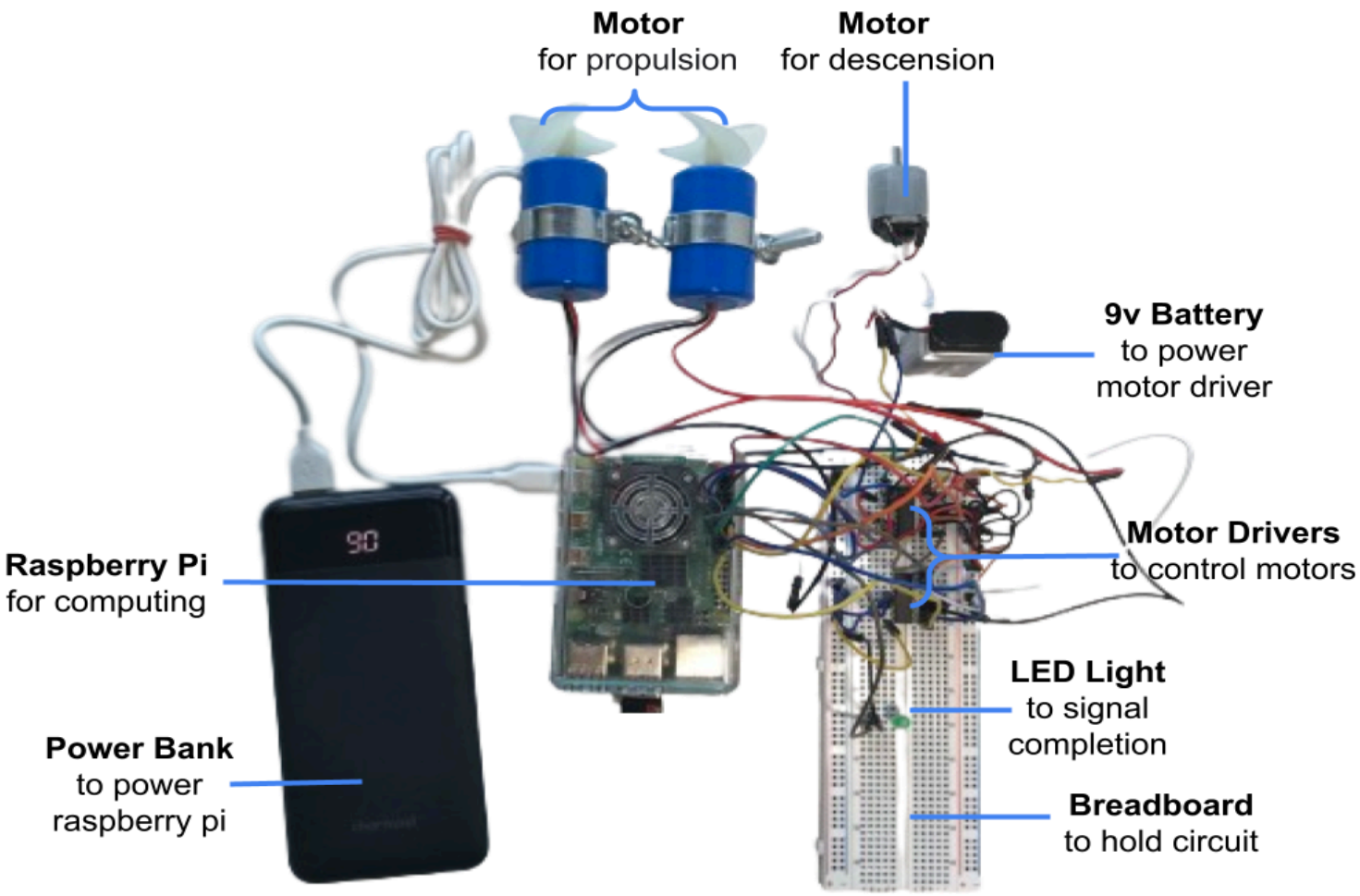
Raspberry Pi Pin Layout. *Raspberry Pi*, www.raspberrypi.com/documentation/computers/os.html. Accessed 16 Nov. 2022.

Saha, Sumit. "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 Way." *Towards Data Science*, 15 Dec. 2018, towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accessed 16 Nov. 2022.

All images created by student researcher

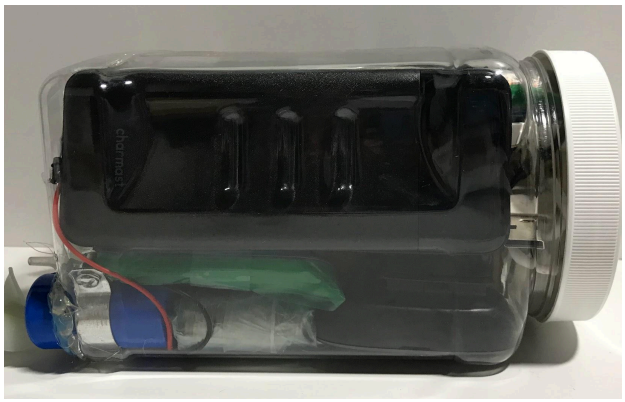
Appendix A

Circuit Design



Appendix B

Images of Machine Views



Side Views

Top View

Bottom View

Appendix C

Detection Script Partial Code

```
# Import statements
import cv2
import numpy as np

# Set camera
cam = cv2.VideoCapture("/dev/video0")
cam.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M','J','P','G'))

# Define HSV values
redMin = np.array([0, 70, 50])
redMax = np.array([10, 255, 255])

# Set condition
found = False
while not found:

      # Read image from camera
      _,img = cam.read()

      # Convert image to HSV
      hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

      # Filter for red in image
      mask = cv2.inRange(hsv, redMin, redMax)

      # Iterate through pixels in image
      for i in range(0, len(mask), 50):
            for n in range(0, len(mask[i]), 50):

                  # Check for red
                  if mask[i][n] == 255:
                        # Break loop
                        found = True

# Close camera
cam.release()
```

Appendix D

Implemented Process

Program Pseudocode

```
Setup output pins
Define variables for speed, lengths, square side, line progress, found status
While no red found
    Check for red
    If no red
        If line progress < length
            Both motors on
            Increase line progress
        Else
            If square side < 4
                One motor on
                Increase square side
            Else
                 One motor on
                 Set square side to 0
                 Decrease line length
            Set line progress to 0
Descension motor on
Light on
```

Program Flowchart