Undulatory Swimming: A Topological Model and Computational Model

Engineering and Technology

Dobromir Iliev, Anish Goyal, Ricardo Chacon

Gwinnett School of Math, Science, and Technology

970 McElvaney Lane NW, Lawrenceville, GA 30044

Acknowledgement of Major Assistance

This project was a collaborative effort between three students: Dobromir Iliev, Anish Goyal, and Ricardo Chacon. Dobromir worked on the mathematical model, established the circuits, administered the experiments, and analyzed the results. Ricardo normalized the synaptic data and helped create a model to detect peaks in the data. Anish set up an SSH server using a Raspberry Pi to facilitate remote access and monitoring of the experimental setup. We utilized synaptic data collected by the Tuan Bui lab to conduct our analysis and multiple online repositories to program the Arduino, serial motor, and INA219 chip. We had no outside assistance other than parental supervision to ensure proper safety precautions.

Table of Contents

Introduction

The initial part of this study investigates the effectiveness of a propulsion system inspired by an organism's synaptic behavior, which governs movement in biological systems. Excessive synaptic activity, especially in Mitophagy and related signaling pathways, can indicate conditions like apoptosis or stroke (Qin et al., 2022). This synaptic activity also triggers muscle fiber movement, resulting in a twitch, as neural circuits controlling movement are influenced by neurotransmitters. Synaptic activity specifically impacts certain locomotor circuits; for instance, NMDA receptors in Zebrafish contribute to motor function in these fish (Rousel et al., 2022). Zebrafish aim for homeostasis, making a propulsion system based on their behavior potentially more energy-efficient. The speed of the fin or motor is directly influenced by the rate of synaptic activity. Despite identifying specific locomotion in Zebrafish, there is limited research on applying this biomimicry to robotics. An energy-efficient submersible, derived from this biomimetic approach, holds potential for enhancing ocean mapping, exploration, and ecosystem monitoring (Fillingham et al., 2023). The design also offers applications in industries like pipeline inspection and offshore infrastructure maintenance, advancing sustainable and effective robotics in many disciplines.

For this purpose, we created two variations of an allocatively efficient model: one based on computational fluid dynamics and the other on topology. The study delves into the efficiency examination of these models. A computational fluid dynamic model, constructed using Navier-Stokes equations, represents the fluid environment of a model (Evgrafov, 2004). These equations incorporate parameters such as water viscosity, flow rate, and water density. While these techniques have traditionally optimized solid structures, their application to undulatory swimming is innovative. Leveraging the success of these optimization programs, our aim is to

use them in modeling a design prototype optimized for a different environment. The use of Navier-Stokes equations for this purpose highlights the novel application of these equations to undulatory swimming. Furthermore, demonstrating the scalability of Navier-Stokes by creating differential equations for a wide range of aquatic environments showcases the versatility of the model. Considering the ongoing changes in ocean compositions due to climate change, a refined model capable of accounting for these parameters holds significant potential.

Materials

- 1 Gallon of distilled water which the model and control will travel through
- 1 meter long water container used in testing
- 1 L985 Motor Driver connected to a stepper motor with inbuilt software
- 1 2019 Raspberry Pi (headless, meaning that there is no hdmi port) used for ROS, reading synaptic inputs, and for controlling
- 1 Arduino Nano used as a microcontroller
- Breadboard Jumper Cables to connect the drivers and arduino to the raspberry pi
- 2 Mini Breadboards (1 for the arduino nano, 1 for the INA 219 sensor module)
- 2 Stepper motors (1 for the control and 1 for the new design)
- 1 USB 3.0 Hub to connect the arduino nano and raspberry pi ensuring consistent current
- 1 Ovonic LiPo battery (5200 mAh, 11.1 V) used to power the system
- 1INA219 Sensor Module to identify the amount of power used in the system
- Material with an equivalent mass of the material used as a control for the design
- 3D printer to print the model
- Duct tape used to connect the motors on the control and the design
- 1 stop watch used to tell the duration of a particular experiment

- Electrical Cables to connect the motor, L985 motor driver, and INA 219 sensor module

Methods

Topological Model:

Fluid Dynamic Simulation:

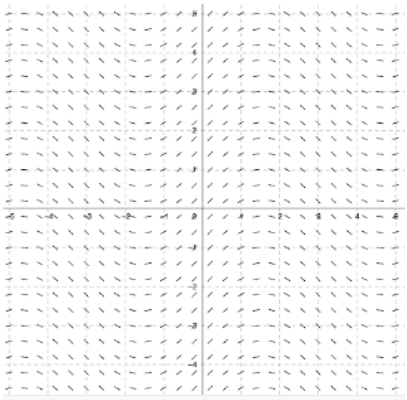
In C++, the Eigen library, featuring the Tensor class, allows for a 3D simulation for fluid dynamics. The process involves parameters such as time step, water density, and viscosity as outlined in the Navier-Stokes equations. A 3D grid representing velocity fields in x,y, and z dimensions is established using the tensor class. To initialize specific shapes, a function ‘intializeShape’ is created implementing the initial velocity values. The ‘dimension()’ function ensures proper tensor dimensions and the main function initializes the simulation grid invoking the ‘intializeShape’ function for initial condition setup. By adjusting parameters and shape logic the code can be adapted as a flexible framework for fluid dynamic models.

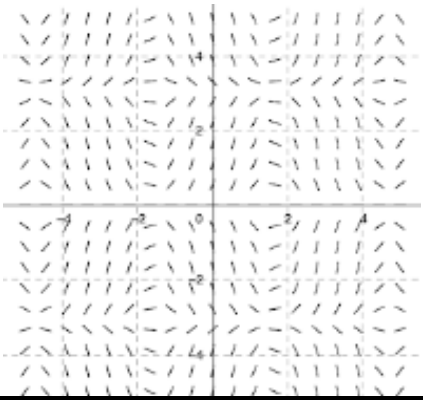
```cpp
#include <iostream>
#include <Eigen/Dense>

using namespace Eigen;

// Constants
const double dt = 0.01; // Time step
const double densityWater = 1000.0;
const double viscosity = 0.1;

// Function to initialize a shape in the velocity field
void initializeShape(Tensor<double, 3>& velocityFieldX, Tensor<double, 3>& velocityFieldY, Tensor<double, 3>& velocityFieldZ) {
    int nx = velocityFieldX.dimension(0);
    int ny = velocityFieldX.dimension(1);
    int nz = velocityFieldX.dimension(2);

    // Specify shape parameters or initialization logic here

    // Initialize shape in the velocity field
    for (int i = 0; i < nx; ++i) {
        for (int j = 0; j < ny; ++j) {
            for (int k = 0; k < nz; ++k) {

                velocityFieldX(i, j, k) = 1.0;
            }
        }
    }
}

int main() {
    //  the size of the simulation grid
    int nx = 5;
    int ny = 5;
    int nz = 5;

    // Initialize velocity fields as Tensor objects
    Tensor<double, 3> velocityFieldX(nx, ny, nz);
    Tensor<double, 3> velocityFieldY(nx, ny, nz);
    Tensor<double, 3> velocityFieldZ(nx, ny, nz);

    initializeShape(velocityFieldX, velocityFieldY, velocityFieldZ);

    return 0;
}
```

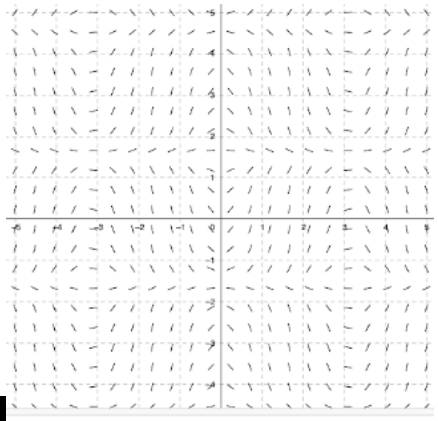
Graphing the velocity fields in MATLab:

The velocity fields can be represented as differential equations and be graphed using MATlab. First create a meshgrid in the x,y,z dimensions to represent the 3D space based upon the simulation grid. Then load the velocity field data from the velocity fields model. Next plot the streamlines, the library used to graph the data, into the x,y,z dimensions. The matlab code is presented below and the results of the differential equations for each dimension are presented.

The velocity field differential equations in the X-axis| Y-axis | Z-Axis



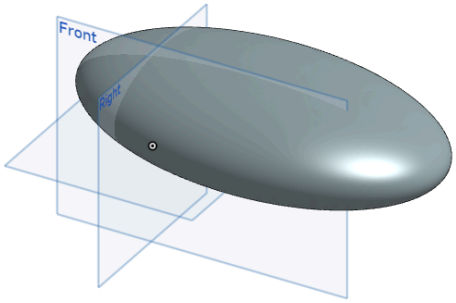




```
% Specify the size of the simulation grid
nx = 5;
ny = 5;
nz = 5;

% Create a meshgrid for the 3D space
[x, y, z] = meshgrid(1:nx, 1:ny, 1:nz);

% Load the velocity field data
```

3D Printing the Model:

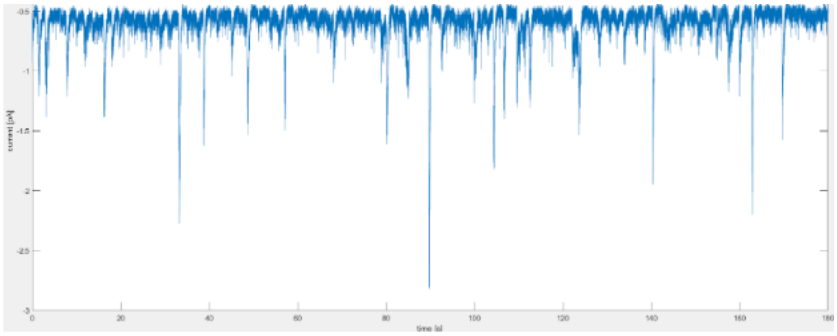
The velocity fields were imported into Onshape. The file was then saved as x.t. file and then 3D printed using SLA techniques. The Onshape model and 3D printed model is shown below.



Neurological Model:

The neurological model takes the synaptic data provided by the BUI lab from the study linked in Roussel et. al, 2021. In matlab, based upon the synaptic, a model which identifies abnormal occurrences in synaptic activity represents locomotion. First the data was normalized by calling normalizedFiltTraces which normalizes the data. Next a structured array ‘minis’ is preallocated and the loop goes through the entire synaptic activity throughout the duration. Within the loop, each trace is processed and peaks are identified with the ‘findpeaks’ function. Further analysis identifies peaks to ensure a rapid decrease before and after. Peaks meeting the criteria are added to the ‘minis’ structure array. A ‘synapticValue’ takes the number of occurrences of ‘minis’ and divides it by the time. The synaptic value is used in the ROS driver to determine the motor speed. The neurological model is provided below as well as the normalized synaptic data.

```
% preallocate minis structure array
minis(numel(normalizedFiltTraces)).minis = [];

parfor i = 1:numel(normalizedFiltTraces)
    % get trace data
    trace = normalizedFiltTraces(i);

    % convert trace to double
    traceDouble = [];
    for j = 1:length(trace.data)
        traceDouble = [traceDouble, double(trace.data(j))];
    end

    % find peaks that meet criteria
    minis(i).minis = [];
    [peaks, locs] = findpeaks(traceDouble, 'MinPeakHeight', 3, 'MinPeakDistance', 5000);

    % check for rapid decrease before and rapid increase after
    for j = 1:numel(peaks)
        beforePeak = traceDouble(1:locs(j));
        afterPeak = traceDouble(locs(j):end);


        % check for rapid decrease
        if any(diff(beforePeak) < -0.05)
            continue
        end


        % check for rapid increase
        if any(diff(afterPeak) > 0.05)
            continue
        end


        % peak meets criteria, add to minis struct
        minis(i).minis = [minis(i).minis, peaks(j)];
        double synapticValue =  (normalizedFiltTraces.size)/time
    end
end

```



Propulsion Model:

The propulsion method is a ROS, robot operating system, node designed for motor control based upon the synaptic data modeled. It reads the Synaptic Value from the neurological model to determine the motor speed. The motor speed is published to a ROS topic which is initialized. The ROS node is shown below.

```cpp
#include "ros/ros.h"
#include "your_package_name/SynapticData.h"
#include "std_msgs/Float64.h"
#include <fstream>

// Function to read synaptic data from a file
float readSynapticDataFromFile(const std::string& filename) {
    std::ifstream file(filename);
    float synapticValue;

    if (file.is_open()) {
        file >> synapticValue;
        file.close();
    } else {
        ROS_ERROR("Unable to open file: %s", filename.c_str());
    }

    return synapticValue;
}

void synapticCallback(const your_package_name::SynapticData::ConstPtr& msg) {
    // Use synaptic data to determine motor speed
    float synapticValue = readSynapticDataFromFile(C:\Users\hungr\Downloads\synaptic_data);

    float motorSpeed = synapticValue;

    // Publish motor speed
    ros::NodeHandle n;
    ros::Publisher motorPub = n.advertise<std_msgs::Float64>("/motor_speed", 1);
    std_msgs::Float64 motorSpeedMsg;
    motorSpeedMsg.data = motorSpeed;
    motorPub.publish(motorSpeedMsg);
}

int main(int argc, char** argv) {
    ros::init(argc, argv, "motor_control_node");
    ros::NodeHandle nh;

    // Subscribe to synaptic data
    ros::Subscriber synapticSub = nh.subscribe("synaptic_data", 1, synapticCallback);

    // ROS spin
    ros::spin();

    return 0;
```

Circuitry:

Motor Driver Connecting the 28BYJ Stepper Motor with ULN2003 Arduino:

To connect the stepper motor to an Arduino follow the following steps:

1) Connect the VCC (power) pin of the motor to the 5V output on the Arduino. 2) Connect the GND (ground) pin of the motor to the GND on the Arduino. 3) Connect the IN1, IN2, IN3, and IN4 pins of the motor driver to digital pins 9, 10, 11, and 12 on the Arduino, respectively. 4) Connect the motor to the driver using the provided connector.

The code provided by the repository by Barnik, 2019, is used to adjust the motor for open loop control. Adjust the number of steps or add conditions as needed for your specific application.

```
int step_number = 0;
void setup() {
pinMode(STEPPER_PIN_1, OUTPUT);
pinMode(STEPPER_PIN_2, OUTPUT);
pinMode(STEPPER_PIN_3, OUTPUT);
pinMode(STEPPER_PIN_4, OUTPUT);

}

void loop() {
 for(int a = 0; a < 1000; a++){
  OneStep(false);
  delay(2);
 }

}


void OneStep(bool dir){
    if(dir){
switch(step_number){
  case 0:
  digitalWrite(STEPPER_PIN_1, HIGH);
  digitalWrite(STEPPER_PIN_2, LOW);
  digitalWrite(STEPPER_PIN_3, LOW);
  digitalWrite(STEPPER_PIN_4, LOW);
```

INA219 Power Measurement:

To connect the INA219 to the raspberry pi connect the following 1) VN+ to the positive terminal of the power source 2) VN- to one lead of the indicator lamp (load) 3) Other lead of the indicator lamp (load) to the negative terminal of the power source 4) VCC to the 3.3V pin on the Raspberry Pi 5) GND to the ground pin on the Raspberry Pi 6) SCL to GPIO3 (SCL pin on the Pi) 7) SDA to GPIO2 (SDA pin on the Pi) Next Install the code from the repository in Borill, 2023. Run the commands sudo apt-get update and sudo apt-get upgrade. Install the necessary Python libraries using commands pip install adafruit-circuitpython-ina219. This Python code continuously reads voltage, current, and power from the INA219 sensor.

Open Loop Control:Open-loop control involves sending a command to a system without receiving feedback. Perform the following steps 1) l:Connect the stepper motor to the Arduino using the ULN2003 driver. Power the motor by connecting the VCC pin to the 5V output on

Arduino. 1) Connect the GND pin of the motor to the GND on Arduino. 2) Connect IN1, IN2, IN3, and IN4 pins of the motor driver to digital pins 9, 10, 11, and 12 on Arduino, respectively. 3) Use the provided code by Bartnik (2019) to control the motor. Adjust steps and conditions based on your application

Closed-loop Control:

Closed-loop control involves feedback to regulate the system involving ROS. 1) Set up the ROS node for motor control 2) Publish motor speed to a ROS topic 3) Ensure that the control loop reads the synaptic value.

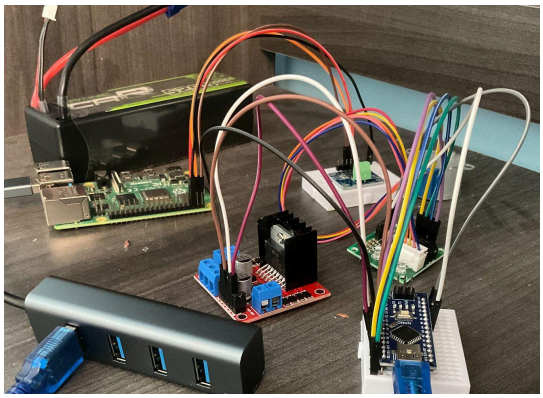
Comms Layer:

The comms layer involves interfacing between different components. 1) Connect the INA219 sensor using GPIO pins. Install the necessary python libraries and use the code provided by Borill (2023).

Robot Controller:

The robot controller is implemented using ROS. Ensure that the ROS node for motor control is set up to receive synaptic data.

Here is a picture of the model before the power cables were connected



Supply Voltage:

Ensure that the power connections for the motor driver, raspberry pi, and other components are established. Turn on the LiPO battery to power the system.

Download Repositories/Connecting to SSH:

First connect to SSH, Secure Shell, you can connect to SSH through PuTTY or through the Linux terminal depending on the operating system of the device. For the repositories linked in the paper use the git clone command to download.

Procedures

A control is needed to determine whether the results of the experiment are statistically significant. For the first test involving the design of the model, the control was a rectangular block which represents a non-biomimetic shape. In addition the wooden block needs to be measured to be equivalent to the mass of the model to make sure there were no confounding variables coming from the mass of the object. The same motor and the same system is used for both the control and the design as well. For the second experiment involving the propulsion system based upon the synaptic data, the control is the non-optimised propulsion model for an open-loop system and the experiment was an optimized propulsion model for a closed-loop system.

The Model Design Test involves the following procedures:

1) Setup: create the topological model, 3D print the model, first set up the circuits with the breadboard pins making sure to set up the INA219 sensor, stepper motor, open-loop, and closed loop designs. Attach the power cables at the end to minimize the risk of shock.

2) Make sure that the control and design models are both of equivalent masses, upload the code for the ROS drivers making sure the correct repositories are downloaded.

3) Start the experiment by initiating the swimming model and control simultaneously. Make sure that the models don't hit the edge of the testing environment as impulses distort data.

4) Ensure that both models are subjected to the same experimental conditions, if they are, then record the data from the time based on the stopwatch. Record the data to the appropriate number of significant figures.

5) Perform the experiment 5 times for the control and experiment taking note of the data.

6) Perform statistical tests, a Z-test, to determine if the data is statistically significant or if there was outside variance affecting the result.

7) Dispose of the distilled water after the experiment, make sure to power off the raspberry pi with the 'systemctl poweroff' command, turn off the LiPo battery and electronics.

The Propulsion Test involves the following procedures:

1) .Setup: create the topological model, neurological model, and propulsion model, then 3D print the model, set up the circuits with the breadboards making sure that the INA219 sensor, stepper motor, open-loop, and closed loop designs are properly connected. Attach the power cables at the end to minimize the risk of shock.

2) Make sure that the control and design models are both of equivalent masses, upload the code for the ROS drivers making sure the correct repositories are downloaded.

3) Start the experiment by initiating the swimming model and control simultaneously. Make sure that the models don't hit the edge of the testing environment as impulses distort data.

4) Ensure that both models are subjected to the same experimental conditions, if they are, then record the data. Take the power before and after the trial using the 'ina.power' command from the repository downloaded from Borill, 2023 after each experiment, it will output in kWh.

5) Perform the experiment 5 more times for the control and experiment noting the data.

6) Perform statistical tests, a Z-test, to determine if the data is statistically significant or if there was outside variance affecting the result.

7) Dispose of the distilled water after the experiment, make sure to power off the raspberry pi with the 'systemctl poweroff' command, turn off the LiPo battery and electronics.

Results

The first test involving the design and the control without the design was recorded in seconds to two significant figures. Since the parameter in the topologic model involving the time-step is 0.1, the model is only accurate to two significant figures. A Z-test was performed as the standard mean and standard deviation was known. The test looked at the statistical significance of the design in comparison to the control. The z-score was -1.912 which corresponds to a p-value of 0.279 which indicates statistical significance at the 0.05 cutoff.

Test #1: Different Designs	Control (s)	Modified Design (s)
Trial 1	3.4	3.2
Trial 2	3.3	3.1
Trial 3	3.4	3.1
Trial 4	3.3	3.2
Trial 5	3.2	3.2
Average	3.32	3.16
Standard Deviation	0.083666003	0.054772256
Z-Score		-1.912365775

The second test involved the propulsion method vs a simple open loop control method. The units were in kWh which is the amount of power consumption over a 1 hour duration. A Z-test was performed as the standard mean and standard deviation was known.The z-score was -1.855 which corresponds to a p-value of 0.0315 which indicates statistical significance at the 0.05 cutoff.

Test #2: Different Closed-loop ROS	Control (kWh)	Loop with Synaptic Data (kWh)
Trial 1	0.0576	0.055
Trial 2	0.0578	0.0546
Trial 3	0.0589	0.0559
Trial 4	0.0626	0.0563
Trial 5	0.0569	0.051
Average	0.05876	0.05456
Standard Deviation	0.002263405	0.002103093
Z-Score		-1.855611744

Discussion

The scalability of the project is demonstrated through the differential equation models, which can be adapted to different environments. For example, in fluid dynamics, the equations can be adjusted to accommodate varying water density due to impurities. Additionally the scalability extends to the size of the models which makes the models applicable to a wide range of scales.

The project's modular circuitry further contributes to the scalability by allowing integration with different components. In a robotic project, for instance, the circuitry accommodates the interchangeability of motors, such as stepper motors to dynamixel motors without extensive redesign.

The project demonstrates great compatibility with LIDAR technology, which currently contributes to optimizing motion planning and enhancing energy efficiency. As demonstrated in Querelta et. al. in 2019, LIDAR is able to generate detailed sensory outputs based on environmental data. The data could be integrated into the project through compatibility with ROS drivers, allowing for communication within the system. The closed loop feedback control system utilizes synaptic data to determine motor speeds, however with the addition of LIDAR in combination with a sensor and JeVois camera, the model would be able to adjust based on the real-time environment. The motor could dynamically adjust speeds leading to optimized motion planning which improves the efficiency of the model and conserves energy consumption.

Conclusion

Based upon the statistical significance for both the propulsion method and the model based upon the navier-stokes equations the null hypothesis -that the propulsion method and model would not be statistically significant -, fails to hold. The results showcased that the

propulsion method and design are likely not caused by random variance as the p-values were less than 0.05.

The results are surprising given that the average power consumption of the trials for the closed-loop system was 0.0545 kWh which was remarkably similar to the .05 kWh expected with the raspberry pi and stepper motor (.1 kWh from the stepper motor and .4 kWh from the raspberry pi). Typically there would be a larger difference in the realm of 15-20% because of the change in current as well as other variance from the environment or loss of heat by the motors. However further investigation taking place over a longer time frame would need to take place to see if the power consumption remains constant for longer duration. A different power sensor and a larger testing environment would be needed to perform that trial.

Furthermore, some variance was expected from the design of the model since it was printed using SLA printing ridges along the surface. However, calibration ensured that the equivalence in mass between the control and design, which maintained a consistent system of 500 revs/m, would reduce the variance. Furthermore the usb-c likely resulted in a constant current being delivered to the arduino nano stabilizing the power consumption. To reduce the variance even more, different motors, a different motor driver, and a model printed using SLS printing technology could be used.

References

Bartnik, N. (2019, May). *Nikodembartnik/ArduinoTutorials: Basic arduino tutorials for Beginners*. GitHub. https://github.com/NikodemBartnik/ArduinoTutorials

Borrill, C. (2023, April). *Chrisb2/pi_ina219: This Python Library supports the INA219 voltage, current and power monitor from Texas Instruments with a raspberry pi using the I2C bus. the intent of the library is to make it easy to use the quite complex functionality of this sensor.* GitHub. https://github.com/chrisb2/pi_ina219

Evgrafov, A. (2004, December). Topology optimization of Navier-Stokes equations. https://www.semanticscholar.org/paper/Topology-Optimization-of-Navier-Stokes-Equations-Evgrafov/c07ff8c44d5a24a9e6ff7dfec6ae8a42d93b054d

Fillingham, J., Harrington, A., Deehan, M., Renta, I., Gugliotti, E., Matlock, G., Bayler, E., Davis, S., Ghirardelli, J., Grasso, M., Harmon, M., Liddel, M., Marshak, A., Arzayus, F., Newcomb, L., Wielgus, J., Schepel, K., Schattel, J., & Yencho, M .(2022). NOAA science report. https://doi.org/10.25923vsa1-gg59

Roussel, Y., Gaudreau, S. F., Kacer, E. R., Sengupta, M., & Bui, T. V. (2021). Modeling spinal locomotor circuits for movements in developing zebrafish. eLife, 10, e67453. https://doi.org/10.7554/eLife.67453

Qin, C., Yang, S., Chu, Y.-H., Zhang, H., Pang, X.-W., Chen, L., Zhou, L.-Q., Chen, M., Tian, D.-S., & Wang, W. (2022, July 6). *Signaling pathways involved in ischemic stroke: Molecular mechanisms and therapeutic interventions*. Nature News. https://www.nature.com/articles/s41392-022-01064-1

Queralta, J. P., Yuhong, F., Salomaa, L., Qingqing, L., Gia, T. N., Zou, Z., Tenunen, H., & Westerlund, T. (2019, October). FPGA-based architecture for a low-cost 3D lidar design and ... https://ieeexplore.ieee.org/abstract/document/8956928/