Xilin Gao, Zixiang Zhou, Emily Ferguson
CSE 237D
Professor Kastner
June 10, 2021

FishSense Final Report

1. Abstract

To modernize archaic fish research methods such as Catch-and-Release, the FishSense project introduces underwater imaging and machine learning pipelines. Although fish length is essential for studying fish populations, fisheries have been incurring the cost of ordering large quantities of fish and measuring them by hand, one-by-one. We automate the measurement of fish length by aligning RGB and depth images captured with a RealSense D455 camera. Automation will dramatically reduce the cost of measuring fish, allowing fisheries to drastically scale up their research. Currently, our length detection algorithm calculates the length of a fake fish from images taken in air within 8cm with 86.88% accuracy. Our framework filters noise from underwater images and isolates the edges of fish for measurement.

2. Introduction

2.1 Background

Fish are a vital part of our global ecosystem; in order to protect and preserve fish populations, researchers must be able to effectively study fish. Fish length is the main parameter that ichthyologists use to determine fish reproduction, recruitment, growth and mortality [3]. Historically, a primary research method for fisheries has been to order large quantities of fish and measure them by hand, one-by-one [3]. Not only is this approach extremely invasive, but it also results in lost information about the fish. When fish are studied outside of the ocean, ichthyologists cannot track the interactions between fish or between fish and their environment. Moreover, measuring fish after catching them introduces potential sources of error such as the shrinking of fish due to preservation techniques and inconsistencies in the physical measuring process [2]. Other techniques such as catching a small sample of fish, measuring them, and releasing them (Catch-and-Release) as well as encouraging fish to swim through measurement devices can cause damage to the fish and potentially injure them [4]. Recently, machine vision approaches have attempted to revolutionize fish research [1]. However, the required technology such as laser calipers and stereo camera rigs can be inaccurate and bulky. Thus, researchers are seeking an efficient, non-invasive approach to measuring fish.

2.2 Proposed Solution

We present FishSense as a solution for capturing all characteristics of fish in their natural habitat. FishSense is a diver-operated handheld system which encapsulates two cameras - an RGB camera and a RealSense D455 depth camera [5] - inside of a water resistant casing for underwater imaging.

Figure 2.1: The handheld FishSense device encapsulates a depth camera and an RGB camera in a water resistant casing.

The FishSense team has already developed a machine learning model for fish detection in underwater images. Other goals of the project include developing an ML model for fish species classification and gathering geometric and volumetric information about fish within their natural habitat in real time. The extra information gathered from studying fish efficiently and accurately in their local environment will improve AI models used for fish research, which will allow fisheries' research to scale up dramatically.

2.3 Our Contributions

The current FishSense solution lacks a pipeline for automatically detecting the length of fish from underwater images. To accomplish this task, we first used images captured in air with a fake fish in order to test our length detection algorithm. Then, we utilized various algorithms to process underwater images. Through filtering noise from underwater images, we developed a framework to detect the edges of the heads and tails of fish from depth images for measurement. Specifically, our contributions include the following:

1. Calibrating the camera - this step involves using a checkerboard model to undistort images.
2. Aligning the RGB and depth cameras - aligning the cameras is necessary in order to process information from the rgb images (such as bounding boxes around detected fish) along with the information from depth images in order to determine fish length.
3. Developing a length detection algorithm - our algorithm can detect the length of a fake fish from images captured in air using aligned RGB and depth images and bounding boxes around the detected fish.
4. Processing underwater images - we utilized various underwater image processing algorithms in order to improve the quality of our underwater images.
5. Developing an edge detection framework - our edge detection framework takes in depth images and the coordinates of bounding boxes around fish and returns the coordinates of the edges of the head and the tail of the fish which alongside with depth information can determine the length of the fish.

3. Technical Material

This section of the paper discusses the technical details of our contributions.

3.1 Calibration

Some pinhole cameras have distortion and make the images unreal, e.g. straight lines become bulged out. The two most common distortions are radial distortion and tangential distortion.

radial distortion:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

tangential distortion:

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

combine them and get the distortion coefficients:

$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

To get the distortion model and coefficients, we need some additional information, like the intrinsic matrix and extrinsic matrix. Intrinsic matrix is unique and unchanged to a camera once the principle points (x_0, y_0) and focal length (f_x, f_y) are fixed (in openCV, skew is set to 0), so that it can be reused. Extrinsic matrix relates to the rotation and translation matrices. Once the camera pose changes, the extrinsic matrix will change accordingly.

To find the parameters, a well-defined pattern, e.g., a checkerboard, is required. The corners in the checkerboard are known in both pixel frame and world frame. The way of calibration openCV used [8] is based on Dr. Zhang [9]. Then, we would be able to undistort the image.

3.2 Alignment of RGB image and Depth image

Thanks to the built-in function from the RealSense library [10], we were able to transfer between 2D pixel frame and 3D space, between color frame and depth frame, or directly use an aligned frame set.

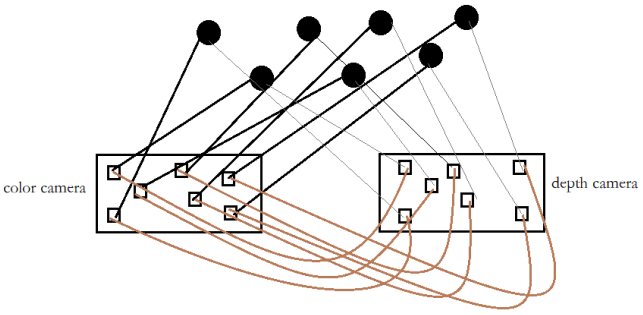

Figure 3.1: Transformation of Color Camera and Depth Camera

3.2.1 2D pixel -> 3D space

Given pixel coordinates and depth in an image with no distortion or inverse distortion coefficients, compute the corresponding point in 3D space relative to the same camera

static void rs2_deproject_pixel_to_point(float point[3], const struct rs2_intrinsics * intrin, const float pixel[2],

float depth).

3.2.2 3D space 1 -> 3D space 2

Transform 3D coordinates relative to one sensor to 3D coordinates relative to another viewpoint
static void rs2_transform_point_to_point(float to_point[3], const struct rs2_extrinsics * extrin, const float from_point[3]).

3.2.3 3D space -> 2D pixel

Given a point in 3D space, compute the corresponding pixel coordinates in an image with no distortion or forward distortion coefficients produced by the same camera.
static void rs2_project_point_to_pixel(float pixel[2], const struct rs2_intrinsics * intrin, const float point[3])

3.2.4 How to do RGBD alignment by utilizing the three functions above?

Generate color stream and depth stream separately, then use a point to point mapping that is to map a depth pixel to a color pixel.

First, we would like to transfer from depth 2D pixel $P_{u,v}^{d}$, $(u,v,z)^{T}$, to depth 3D space P_{dc}, $(x,y,z)^{T}$, where K_d^{-1} is the inverse intrinsic matrix of depth camera.

$P_{dc} = K_d^{-1} P_{u,v}^{d}$

$$\begin{bmatrix} u \\ v \\ z \end{bmatrix} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ z \end{bmatrix}$$

Second, we would like to transfer from depth 3D space P_{dc}, $(x,y,z)^{T}$, to color 3D space P_{cc}, $(x',y',z')^{T}$. Actually there are two steps here, first transfer from depth 3D space P_{dc}, $(x,y,z)^{T}$, to world coordinate P_w, $(X,Y,Z,1)^{T}$, then transfer from world coordinate P_w, $(X,Y,Z,1)^{T}$, to color 3D space P_{cc}, $(x',y',z')^{T}$, where T_{w2d}^{-1} is the inverse extrinsic matrix for depth camera, and T_{w2c} is the extrinsic matrix for color camera.

$P_w = T_{w2d}^{-1} P_{dc}$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$P_{cc} = T_{w2c} P_w$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} R'_{11} & R'_{12} & R'_{13} & t'_1 \\ R'_{21} & R'_{22} & R'_{23} & t'_2 \\ R'_{31} & R'_{32} & R'_{33} & t'_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Combine the two equations above, we get:

$P_{cc} = T_{w2c} T_{w2d}^{-1} P_{dc} = T_{d2c} P_{dc}$

$$T_{d2c} = T_{w2c} T_{w2d}^{-1} = \begin{bmatrix} R_{w2c} R_{w2d}^{-1} & t_{w2c} - R_{w2c} R_{w2d}^{-1} t_{w2d} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} R'_{11} & R'_{12} & R'_{13} & t'_1 \\ R'_{21} & R'_{22} & R'_{23} & t'_2 \\ R'_{31} & R'_{32} & R'_{33} & t'_3 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Third, we would like to transfer from color 3D space P_{cc}, $(x',y',z')^{T}$, to color 2D pixel, $P_{u,v}^{c}$, $(u',v',z')^{T}$, where K_c is the intrinsic matrix of the color camera.

$P_{u,v}^{c} = K_c P_{cc}$

$$\begin{bmatrix} u' \\ v' \\ z' \end{bmatrix} = \begin{bmatrix} f_x' & s' & x_0' \\ 0 & f_y' & y_0' \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

3.2.5 Another way to do alignment

This approach is simpler, that is to create an align filter alignment between the depth image and the RGB image. In this way, we first create an aligned frameset and then get color and depth frames.

3.2.6 After Alignment

Our main goal is to get the length, so after we align RGBD images, we can get the X and Y coordinates in the world coordinate with corresponding Z coordinates. By using such points from head and tail, we can then get the real length of a fish. Here we would do the 2D pixel -> 3D space transfer again for the color camera, then the Euclidean distance between such two points is the fish length.

3.3 Length Detection Algorithm

Our length detection algorithm computes Euclidean distance between two points on the fish in three dimensions (taking into account the depth information). The points on the fish are chosen using the bounding box around the entire detected fish. We choose points near the edges of the bounding box as the head and tail of the fish. We test our algorithm on 75 images of a fake fish taken in air.

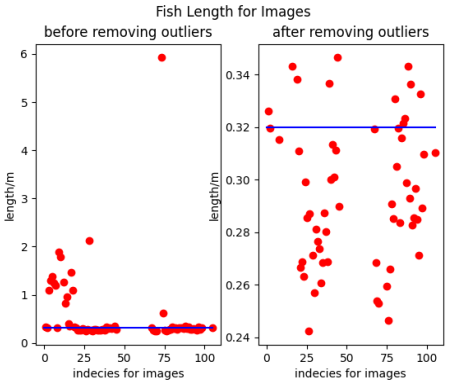

Figure 3.2: Each red dot represents the calculated metric length of the fake fish for a given image. The blue line indicates the true metric length of the fake fish. After removing outliers, we see that we can calculate the metric length of the fake fish within 8cm with 86.88% accuracy.

3.4 Color correction

The acquisition of underwater optical imaging faces more challenges than that in the atmosphere, whereby degradation is usually caused by strong absorption and scattering. Because of the wavelength dependence of light attenuation, the shorter wavelengths (green and blue colors), can reach greater depths under the water than the longer wave-lengths (red color) which vanish rapidly after 4-5 m, leading to images with a typical bluish or greenish tone. [7]

3.4.1 Color transfer

The main goal of color transfer is to adjust the color characteristics of the source image according to the target image. There is a strong correlation between R, G, B channels, so it is really difficult to change the color while in the meantime changing the three channels. Therefore, the idea is to find a way that the channels are uncorrelated, that is, orthogonal color space. The authors recalled the lαβ color space proposed by Ruderman. at el. The three axes are orthogonal, which means that changing either of them will not affect the other two, hence it is better to maintain the natural performance of the original image. [16]

The main idea is to first transfer color in BGR mode to LAB mode, then perform color transfer, and finally transfer from LAB mode back to BGR mode. Luckily, openCV has built-in functions for the color mode transformation. The algorithm for color transfer are as follows:

1) Read in source image and target image (BGR mode), i.e., source[3], target[3].
2) Transfer the images to LAB mode, i.e., sc[3], tc[3].
3) Calculate the mean and standard deviation for each of the LAB channel for the images, i.e., s_mean[3], s_std[3], t_mean[3], t_std[3].
4) For each of the LAB channel (i = 0, 1, 2):
   rc[i] = (sc[i]-s_mean[i])*(t_std[i]/s_std[i])+t_mean[i]
5) Transfer the result image rc to BGR mode, i.e., result.
6) Write back the result image, show the source, target, and result image.

Note: pay attention to the shape of the images. Say the source image in LAB mode is of size r*c*3, the target one in LAB mode is of size m*n*3, mean and standard deviation for each image is of size 3*1. Then for the calculation and to maintain high speed (by vectorization instead of using multiple loops for each pixel and each channel), we would like to broadcast 3*1 to r*c*3, which is to create an r*c*3 matrix full of ones. Then multiply each channel by the corresponding mean or standard deviation.

The whole process is as follows:

1) RGB->XYZ:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.5141 & 0.3239 & 0.1604 \\ 0.2651 & 0.6702 & 0.0641 \\ 0.0241 & 0.1228 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

XYZ->LMS:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

combine these two steps to: RGB->LMS:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2) get the logarithm:

$$\boldsymbol{L} = \log L \quad \boldsymbol{M} = \log M \quad \boldsymbol{S} = \log S$$

3) LMS->lαβ:

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix}$$

4) subtract the mean from the data points (< >means the average mean):

$$l^* = l - \langle l \rangle \quad \alpha^* = \alpha - \langle \alpha \rangle \quad \beta^* = \beta - \langle \beta \rangle$$

5) scale by standard deviation (s:source, t: target, σ: standard deviation):

$$l' = \frac{\sigma_t^l}{\sigma_s^l} l^* \quad \alpha' = \frac{\sigma_t^\alpha}{\sigma_s^\alpha} \alpha^* \quad \beta' = \frac{\sigma_t^\beta}{\sigma_s^\beta} \beta^*$$

6) lαβ->LMS:

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix}$$

7) LMS->RGB:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4679 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

3.4.2 Dehazing

The problem proposed in the paper [11] is dehazing, which can restore the color and visibility of the image, and also utilize fog density to estimate the distance of the object. These can be important applications for computer vision, e.g., 3D reconstruction & identification. Even though the paper was proposed a little bit earlier, it is still simple and efficient compared to nowadays when researchers often use the way of deep learning or machine learning.

The authors paid attention to the characteristics of the haze-free images. In haze-free images, there will be shadows, or pure/black objects in every local area. Therefore, for every local area, at least one color channel will have rather low values.This statistical low is called Dark Channel Prior. Intuitively, Dark Channel Prior thinks there are always some dark objects in every local area.

Since the haze is always gray and white, once the image is affected by the haze, those objects that should be bark will become gray and white. What is more, by the scale of gray and white, the authors were able to know the density of the haze and further estimating the distance of the object.

The definition of dark channel is: $J^{dark}(\mathbf{x}) = \min_{c \in \{r,g,b\}} (\min_{\mathbf{y} \in \Omega(\mathbf{x})} (J^c(\mathbf{y})))$ where J^c is a color channel of **J** and Ω(x) is a local patch centered at x, and J^{dark} is the dark channel of **J**. The patch's transmission is $\tilde{t}(\mathbf{x}) = 1 - \omega \min_c (\min_{\mathbf{y} \in \Omega(\mathbf{x})} (\frac{I^c(\mathbf{y})}{A^c}))$, where A is the global atmospheric light, ω here represents the proportion of how much haze would be removed. The reason for not setting ω to 100% is due to making the scene real and practical with a little proportion of haze and dust from human's eyes. In the paper the authors set it to 0.95, but in our real program, we set it to 0.8, 0.6, or 0.4 according to the proportion of pixel value that was under 50 in the histogram. Then the final formula of dehazing would be: $\mathbf{J}(\mathbf{x}) = \frac{\mathbf{I}(\mathbf{x}) - \mathbf{A}}{\max(t(\mathbf{x}), t_0)} + \mathbf{A}$. Here picking the maximum of t(x) and t_0 is to set a lower bound of the transmission, where t_0 is set to 0.1. Another difference between our program and the paper is that, in the paper, the authors used a local region of 15-pixel wide, but this could cause the problem of obvious edges between blocks, so we changed to use the whole image. Although the problem discussed in this paper is in air, the idea of Dark Channel Prior is still applicable for underwater scenarios.

3.5 Noise Filtering

3.5.1 HE (Histogram Equalization)

Histogram Equalization is a computer image processing technique used to improve contrast in images, which is achieved by effectively spreading out the most frequent intensity values. This method will enhance the contrast of different color pixel dramatically. [17]

3.5.2 CLAHE (Contrast limited adaptive histogram equalization)

Ordinary AHE will have problems of overamplifying the contrast in close-related regions of the image, while in these regions, the histogram is highly concentrated. Hence, AHE may raise the noise in the near regions. To solve this, Contrast Limited AHE (CLAHE) comes, which is a variant of adaptive histogram equalization and the contrast amplification is limited, so as to reduce this problem of noise amplification. [12]

3.5.3 GC (Gamma Correction)

Gamma Correction mainly controls the brightness of the image. It is formed by a non-linear function used to encode and decode luminance or tristimulus values in video or still image systems. It has the following power-law expression:

$$V_{\text{out}} = AV_{\text{in}}^{\gamma}$$

Images taken underwater will always face problems like too bright or too dark. We will use $\gamma < 1$ for gamma compression process and $\gamma > 1$ for gamma expansion. [18]

3.5.4 RGHS (Relative Global Histogram Stretching)

This idea was first illustrated in paper Shallow-Water Image Enhancement Using Relative Global Histogram Stretching Based on Adaptive Parameter Acquisition [13].

Histogram stretching is adopted to the underwater shallow images that can provide a better pixel distribution of the image channels to the whole dynamic range and thus improve the image contrast. A linear contrast stretching function following is used to achieve the idea:

$$p_o = (p_i - a)\left(\frac{c-d}{b-a}\right) + d$$

where p_i and p_o are the input and output pixels intensity values, respectively, a, b and c, d represent the minimum and maximum intensity of the input image and the targeted output images, respectively. In a global stretching, c and d are constant and often set to 255 and 0 respectively; a and b are selected at 0.2% and 99.8% in the whole histogram of the original image.

In most of shallow-water images, the histogram of red light is focused in such values [50, 150], while G channel and B channel have the most numerical concentration in the range [70, 210]. This indicates that histogram stretching should be sensitive to channels. Thus we can rewrite the stretching function to be:

$$p_{out} = (p_{in} - I_{min})\left(\frac{O_{max}-O_{min}}{I_{max}-I_{min}}\right) + O_{min}$$

where p_{in} and p_{out} are the input and output pixels, respectively, and I_{min}, I_{max}, O_{min}, O_{max} are the adaptive parameters for the before and after stretching images, respectively.

3.6 Edge detection

We have developed a framework to detect the edges of the heads and tails of fish from the depth image of the fish.

3.6.1 Inputs

The framework takes in a depth .csv file as well as the coordinates of bounding boxes around detected fish.

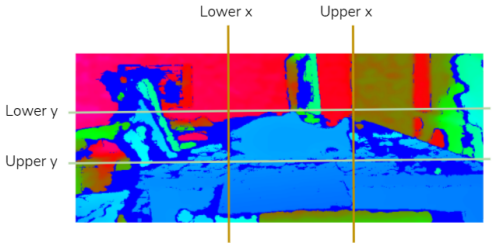

Figure 3.3: The detected fish is indicated by the required coordinates of a bounding box around the fish. The coordinates required include the minimum x and y coordinates (Lower x and Lower y) as well as the maximum x and y coordinates (Upper x and Upper y).

3.6.2 Depth Image Processing

The depth .csv file is first converted to a .png image file for processing. Then, the framework uses the coordinates of bounding boxes to isolate the depth information of fish. Only the depth information of fish is kept in the depth image; all other pixels of the image are set to zero to produce a significant contrast between fish and background in the depth image.

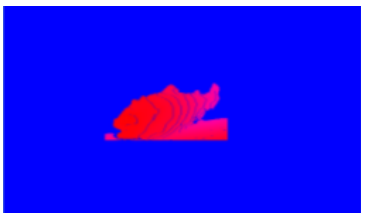

Figure 3.4: The depth information of the fish from the depth image is kept, while the depth information of the background in the image is set to 0 to produce a strong contrast.

3.6.2 Edge Detection

The processed depth image is passed through a Gaussian Blur and Canny Edge Detection algorithm [6] in order to detect the edges of the fish in the image. The bounding box coordinates are then used to find the coordinates of the edges of the head and the tail of the fish. Along with the preserved depth information of the fish, these coordinates can be used to detect the length of the fish.

Figure 3.5: The coordinates of the edges of the head and the tail of the fish are returned by the edge detection algorithm. The selected edges of the head and tail are highlighted in the produced edge detection image.

4. Milestones

4.1 Original Milestone Schedule

SCHEDULE		
Week	MILESTONE DUE	PARTY RESPONSIBLE
4	Searching for length detection algorithm and transfer the bounding box	Xilin Gao, Zixiang Zhou, Emily Ferguson
5	Implement the algorithm to measure the length of a fake fish in air with low noise from RGB images and Depth images	Xilin Gao
5	Perform the selected algorithm with some sample data	Zixiang Zhou
5	Testing the length detection algorithm on images taken in air with minimal noise	Emily Ferguson
6	Research on some papers about underwater noise filtering and calibration	Xilin Gao
6	Complete implementation of length detection algorithm. Perform testing and generate results.	Zixiang Zhou
6	Search for noise filtering algorithm to use on images with moderate noise	Emily Ferguson
6	**Deliverable**: Length detection algorithm succeeds on air images with minimal noise	Xilin Gao, Zixiang Zhou, Emily Ferguson
7	Testing the algorithms for length detection and underwater noise filtering	Xilin Gao
7	Search for different noise filtering algorithms and perform tests.	Zixiang Zhou
7	Test the noise filtration on noisier images	Emily Ferguson
8	Implement the length detection algorithm with underwater scenarios (more noise)	Xilin Gao
8	Combine the noise filtering with the length detection algorithm. Perform more training and testing.	Zixiang Zhou
8	Test the length detection algorithm on noisier images given the noise filtration	Emily Ferguson

9	Combine noise filtering with length detection to get better results	Xilin Gao
9	Generate the final result of length detection after noise filtering on moderate images in air.	Zixiang Zhou
9	Clean up the final algorithm and try the noise filtration and length detection on images with a lot of noise and underwater images if time allows	Emily Ferguson
9	**Deliverable**: Length detection algorithm succeeds on air images with moderate noise	Xilin Gao, Zixiang Zhou, Emily Ferguson
10	Maintain the data structure and code cleaness. Prepare the final report and final presentation	Xilin Gao, Zixiang Zhou, Emily Ferguson

Figure 4.1: The initial schedule by which we planned to complete our milestones.

4.2 Changes to Milestones

Our projected milestones were pushed back approximately one week. In the middle of the quarter, we adjusted our schedule so that each milestone was to be completed one week later. We also adjusted our milestones and deliverables to use underwater images. While we were able to complete image processing with underwater images, we had to revert to using noisy images captured in air for length detection.

4.2.1 Difficult Setup

We ran into various issues in the beginning of the quarter with performing the length detection, including calibration concerns with the RealSense camera. We also had to overcome issues regarding aligning the depth camera and the RGB camera. We anticipated in our initial risk assessment that we might run into issues with the hardware such as calibration and with aligning the RGB and depth images; the difficulty of the setup tasks pushed our planned schedule back.

4.2.2 Fish Length Detection Technique

Instead of initially using bounding boxes around entire fish, we explored a different approach to simulate the natural process of measuring fish from head to tail; we labelled hundreds of images by head and tail of fish in order to create separate bounding boxes around heads and tails rather than one bounding box around the entire fish. However, the head and tail detection did not work with the FishSense's team's fish detection pipeline, as the model would erroneously detect heads as tails and vice versa. Thus, we lost time trying this approach and reverted to our original method of using bounding boxes around the entire fish.

4.2.3 In-Air Versus Underwater Images

In our initial milestones in our project specification, we anticipated that we would only be able to perform length detection on moderately noisy images in air due to an underwater calibration issue that the E4E FishSense team has been facing. However, in the middle of the quarter, we thought that we might get to work with some underwater images due to the E4E team's progress. Until late in the quarter, we did not have any underwater images to work with. When we did obtain some underwater images, we were able to use them to test our noise filtration. However, due to a calibration issue with the cameras underwater, we were not able to align the RGB and depth images for length detection. Thus, we have had to revert to our original plan of detecting length with only images captured in air of a fake fish. Yet, we have been able to test the underwater image processing techniques with images captured underwater. We have built a framework described in this report that has been tested on noisy images captured in air, and the FishSense team should be able to use it on underwater images once they have solved their calibration issue and they have collected more underwater images of fish.

4.3 Milestones Completed

4.3.1 Length detection algorithm

We completed this milestone by doing research on various length detection techniques and sharing thoughts between each other. We tried two algorithms - a pinhole camera model algorithm [14] and an algorithm to find Euclidean distance between heads and tails - and decided to move forward with the Euclidean distance algorithm as it was more accurate.

4.3.2 Calibration

We used two different approaches to calibrate the camera, including calibration by hand using openCV and calibration using the RealSense SDK tools [15]. The details of these algorithms are discussed in Section 3.1.

4.3.3 Bounding box labeling

We labelled the heads and tails of hundreds of images of our test fish in order to pass these images into the fish detection machine learning model to detect head and tails of fish rather than entire fish.

4.3.4 RGB and depth image alignment

We aligned the RGB and depth images as described in Section 3.2.

4.3.5 Length detection implementation
We implemented our Euclidean distance algorithm with the completed RGBD alignment in order to detect fish length and finish our MVP.

4.3.6 Underwater image processing research
We found multiple different algorithms and tested them with online underwater images.

4.3.7 Underwater image processing implementation
We tested our image processing algorithms with our own images captured underwater.

4.3.8 Noise filtration combined with pixel length detection
We combined our Canny Edge Detection algorithm with a depth image processing algorithm in order to calculate the pixel length of fish.

4.4 Deliverables Completed

Week 7	**Deliverable**: Length detection algorithm succeeds on air images with minimal noise	Xilin Gao, Zixiang Zhou, Emily Ferguson
Week 9	**Deliverable:** Noise filtration algorithms succeed to denoise the images underwater	Xilin Gao, Zixiang Zhou, Emily Ferguson
Week 10	**Deliverable:** Fish pixel length detection combined with Edge detecting for underwater Images	Xilin Gao, Zixiang Zhou, Emily Ferguson

Above are the major deliverables we have for this quarter. Basically we completed our plan at the beginning of the quarter. The only difference is that we didn't achieve the fish length detection for all underwater images but just for the sample ones. The main reason and challenges we faced will be discussed detailly in the following section.

4.6 Problems and Discussion
4.6.1 Setup
It is the first step that is troublesome. Our very first step is the hardware and software setup. It took us a long time to prepare all the libraries and tools (realsense SDK and openCV), the hardware support (Intel realsense camera D455 must use a 3.0 USB), and some compile errors due to wrong usage of makefiles, and get familiar with the realsense camera. After the setup, we then encountered issues with RGBD alignment. The align filter way was not possible as the depth frame and the color frame were not in the same size, so it would be very difficult to recognize the corresponding pixels. The one-to-one mapping could thankfully work. We mapped a depth pixel to a color pixel. We could not do it vice versa, as the depth value was corresponding to each depth pixel. Later, with this "transformed" depth image and the original color image, we would finally be able to calculate the length of a fish.

4.6.2 Lack of images
Due to the virtual setting as well as issues that the E4E team has encountered with their camera calibration, we have faced a lack of diverse datasets to test our code with. However, we have collected some of our own images and built our pipeline with the images that were available.

4.6.3 Underwater Length Detection
Unfortunately, we are currently unable to calculate the true metric length of fish from underwater images due to an underwater calibration issue which prevents us from aligning RGB and depth images taken underwater. However, our framework handles underwater images by filtering the depth information of the images before finding pixel length, so it should be able to use underwater images to calculate fish length once the calibration issue is solved.

5. Conclusion
In conclusion, our main goal was the automatic calculation of fish length. To realize this, we first set up the camera in air, did calibration and the alignment of RGB images & Depth images, then calculated the length. Then, we did some experiments of underwater image processing: color transfer & dehazing, different noise filtration algorithms, and Guassian blur & canny edge detection. Our future goal is to automate the length detection underwater when the underwater calibration issue is solved. Additionally, we hope to gather additional data to test our framework with and fine-tune our algorithm to achieve better accuracy with more diverse data.

Other future goals include: combining the whole system, making it real-time, and hopefully integrating the system with a UI/UX interface.

6. References

[1]: Hao, M., Yu, H. and Li, D., 2015, September. The measurement of fish size by machine vision-a review. In International Conference on Computer and Computing Technologies in Agriculture (pp. 15-32). Springer, Cham.
[2]: Mous, P.J., Goudswaard, P.C., Katunzi, E.F.B., Budeba, Y.L., Witte, F. and Ligtvoet, W., 1995. Sampling and measuring. In Fish Stocks and Fisheries of Lake Victoria. A handbook for field observations (pp. 55-82). Samara Publishing Ltd.
[3]: Rahim, M., Abdullah, N., Amin, I., Zakaria, M., Man, M. and Othman, N., 2010. A new approach in measuring fish length using FiLeDI framework. IAJIT First Online Publications.
[4]: Tillett, R., McFarlane, N. and Lines, J., 2000. Estimating dimensions of free-swimming fish using 3D point distribution models. Computer Vision and Image Understanding, 79(1), pp.123-141.
[5]: Anon, 2021. Introducing the Intel® RealSense™ Depth Camera D455. Intel® RealSense™ Depth and Tracking Cameras. Available at: https://www.intelrealsense.com/depth-camera-d455/ [Accessed June 9, 2021].
[6]: FienSoP, FienSoP/canny_edge_detector. GitHub. Available at: https://github.com/FienSoP/canny_edge_detector [Accessed June 9, 2021].
[7]: Min Han, Zhiyu Lyu, Tie Qiu, and Meiling Xu, A Review on Intelligence Dehazing and Color Restoration for Underwater Images, 1820-1832, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, VOL. 50, NO. 5, MAY 2020
[8]: OpenCV Camera Calibration: https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html
[9]: Zhengyou Zhang, A Flexible New Technique for Camera Calibration, Microsoft Research
[10]: Intel Realsense Github, librealsense2/rsutil.h: https://github.com/IntelRealSense/librealsense/blob/master/include/librealsense2/rsutil.h
[11]: Kaiming He, Jian Sun, Xiaoou Tang, Single Image Haze Removal Using Dark Channel Prior, 1956-1963, 978-1-4244-3991-1/09/$25.00 ©2009 IEEE
[12]: G. Yadav, S. Maheshwari and A. Agarwal, "Contrast limited adaptive histogram equalization based enhancement for real time video system," 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, pp. 2392-2397, doi: 10.1109/ICACCI.2014.6968381.
[13]: Dongmei Huang, Yan Wang, Wei Song, Jean Sequeira, Sébastien Mavromatis. Shallow-water Image Enhancement Using Relative Global Histogram Stretching Based on Adaptive Parameter Acquisition. 24th International Conference on Multimedia Modeling - MMM2018, Feb 2018, Bangkok, Thailand. ffhal-01632263f
[14]: Principles of Pinhole camera model: https://threeconstants.wordpress.com/tag/pinhole-camera-model/
[15]: Dynamic Calibration User Guide for Intel® RealSense™ D400 Series: https://www.intel.com/content/www/us/en/support/articles/000026723/emerging-technologies/intel-realsense-technology.html
[16]: Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley, Color Transfer between Images, 34-41, 0272-1716/01/$10.00 © 2001 IEEE
[17]: W. Zhihong and X. Xiaohong, "Study on Histogram Equalization," 2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing, 2011, pp. 177-179, doi: 10.1109/IPTC.2011.52.
[18]:Rahman, S., Rahman, M.M., Abdullah-Al-Wadud, M. et al. An adaptive gamma correction for image enhancement. J Image Video Proc. 2016, 35 (2016). https://doi.org/10.1186/s13640-016-0138-1