Milestone Report
FishSense
Xilin Gao, Zixiang Zhou, Emily Ferguson
5/18/21

Milestone Progress Update/Reflection:
Our projected milestones have been pushed back approximately one week. We have run into various issues with performing the length detection, including calibration concerns with the RealSense camera. We have also had to overcome issues regarding aligning the depth camera and the RGB camera. Additionally, we have been waiting on underwater images from the E4E team who was encountering a problem with calibrating their cameras for underwater depth imaging concurrently with our project. Thus, we have been testing with the images that we do have that were captured in air until we can get underwater images (which should be this week).

In fact, we anticipated in our risk assessment that we might run into issues with the hardware such as calibration and there would also be issues regarding getting underwater data. Therefore, we have adjusted our milestones slightly in order to account for these issues.

Changes to Milestones:

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. 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.

Fish Length Detection Technique:
We have also slightly adjusted our approach to fish measurement to match the more natural approach that is often used in fisheries. Instead of measuring fish using one bounding box around the entire fish, we have relabelled images of our test fish in order to generate bounding boxes around the head and tail of the fish. We are now taking the average of a few points chosen from the fish’s head and the average of a few points chosen from the fish’s tail and taking the Euclidean distance.

Milestones Completed:

Original plan:

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

Current plan:

As we mentioned in the reflection, our timeline was pushed back a week from our original plan.

~~Week 4~~ -> Week 5:
Length detection algorithm (Xilin Gao, Zixiang Zhou, Emily Ferguson):
We completed this milestone by doing research and sharing thoughts between each other. Then two algorithms were made.

The first algorithm was simply just using a pinhole camera model, but it was rather inaccurate. At that time, when we first found that the RGB image and the Depth image were at different size (from the original streams, RGB image was 720*1280, and Depth image was 480*848), we just simply resized the RGB image to the same scale as the Depth image. We also did not notice the alignment at that time, so the main problem was that at a given RGB pixel (r, c), the depth value of Depth pixel (r, c) would definitely not be the corresponding depth. Also this simple approach did not take calibration and distortion coefficients into account. It was obvious the result was a big mass.

The second algorithm is more straightforward. We directly find the heads and tails of the fish and use their 3D coordinates to calculate its length. With this method, we will then need

the calibration of the camera and a way to find the head and tail accurately.
After discussion, with the bounding boxes provided by the E4E team, we decided to randomly sample n pairs of points in the head and tail bounding boxes and then calculate the average distance of it, which will give us a relatively more reliable result.

~~Week 5~~ -> Week 6:
The following milestones were adjusted due to hardware complications as well as an approach change as mentioned in our reflection.

Calibration: (Xilin Gao, Zixiang Zhou)

- Calibration by hand using openCV
  After fixing the size of the frame (different frame size would cause different camera matrices), we used the camera to capture the checkerboard in different camera poses. As the poses were different, the Rotation and Translation would be different, thus causing the extrinsic matrices to be different, but the intrinsic matrices would be the same.

  In the calibration algorithm, it would like to detect corners and find an r by c pattern in the pixel frame and the world frame, and then use Dr. Zhang's algorithm (A Flexible New Technique for Camera Calibration, Zhengyou Zhang, https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf) to generate the camera matrices, and then it would undistort the image.

- Realsense camera calibration
  Intel Realsense camera has its own tool for calibration called "dynamic calibration". It would capture 15 different images for scale and color, respectively, and then propagate the updates back into the camera.

Bounding box labeling: (Emily Ferguson)
The completion of this milestone involved labelling 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.

~~Week 6~~ -> Week 7:
RGB and depth image alignment & Length Detection implementation: (Xilin Gao)
There were two ways to do RGBD alignment.

One was to generate color stream and depth stream separately, then used a point-to-point mapping that was to map a Depth pixel to an RGB pixel. There were mainly three steps: first transformed from depth 2D pixel to depth 3D space (rs2_deproject_pixel_to_point), then transformed from depth 3D space to color 3D space (rs2_transform_point_to_point),

and finally transformed from color 3D space to color 2D pixel (rs2_project_point_to_pixel). We would need some parameters, such as depth scale, depth intrinsic matrix, color intrinsic matrix, extrinsic matrix from Depth to Color. Why did we start with the Depth pixel? This was simply due to the need of depth value that corresponded to a depth pixel. Thankfully this approach worked.

The second approach was simpler, which was to create an align filter Alignment between the depth image and the RGB image. In this way, we first created an aligned frameset and then got color and depth frames. However, this approach did not work due to two reasons. First, the size of the RGB image was different from the size of the Depth image, so it would be very difficult to mark the pixels as “mapped”. Second, there was an issue using openGL when we tried to change the image size at the very beginning - when creating the Depth stream and the Color stream.

Then, after we aligned RGBD images, we could get the X and Y coordinates in the world coordinate with corresponding Z coordinates. By using such points from head and tail, we could then get the real length of a fish. Remember the coordinates transformation mentioned above? Here from color pixel frame to color camera frame, we used the transformation again. You might be wondering why we were calculating in the color camera frame instead of the world frame. This was because here we put the origin of the world frame on where the origin of the color camera frame was. There was no translation or rotation, so the color camera frame and the world frame coincided. Therefore, the points and distance in the color camera frame were what we wanted.

Noise filtration algorithm research: (Zixiang Zhou, Emily Ferguson)
The completion of this milestone involved reading about noise filtration algorithms and searching the Internet for noise filtration techniques. We have found a technology online that seems promising with our RGB images (https://github.com/danaberman/underwater-hl), but we will need to iterate on the algorithm in order to fit our needs. We also need to continue researching how to denoise our depth images.

Deliverables completed:

Week 7	**Deliverable**: Length detection algorithm succeeds on air images with minimal noise	Xilin Gao, Zixiang Zhou, Emily Ferguson

Milestones Remaining:

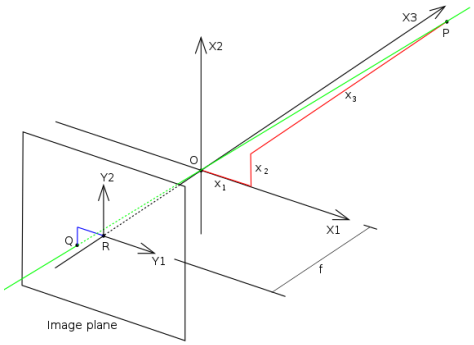
**Note: We generally have the same milestones as in our initial project specification. However, we have pushed back the timeline by a week as mentioned above. Additionally, due to the E4E team solving their underwater imaging calibration issue, we have adjusted our milestones to performing noise filtration and length detection on underwater images rather than noisy images taken in air. Please see the updated milestones below.

SCHEDULE		
Week	**MILESTONE DUE**	**PARTY RESPONSIBLE**
8	Testing the algorithms for length detection and underwater noise filtering	Xilin Gao
8	Search for different noise filtering algorithms and perform tests.	Zixiang Zhou
8	Test the noise filtration on noisier images	Emily Ferguson
9	Implement the length detection algorithm with underwater scenarios (more noise)	Xilin Gao
9	Combine the noise filtering with the length detection algorithm. Perform more training and testing.	Zixiang Zhou
9	Test the length detection algorithm on underwater images given the noise filtration	Emily Ferguson
10	Combine noise filtering with length detection to get better results	Xilin Gao
10	Generate the final result of length detection after noise filtering on underwater images	Zixiang Zhou
10	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
10	**Deliverable**: Length detection algorithm succeeds on underwater images	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

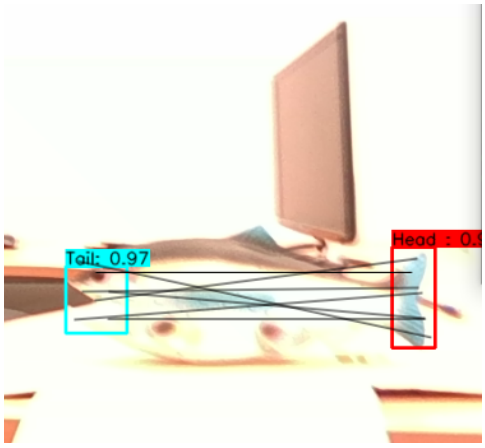
Picture/Video demo:

Week 5:

Length detection algorithm (considering the coordinate transformation):



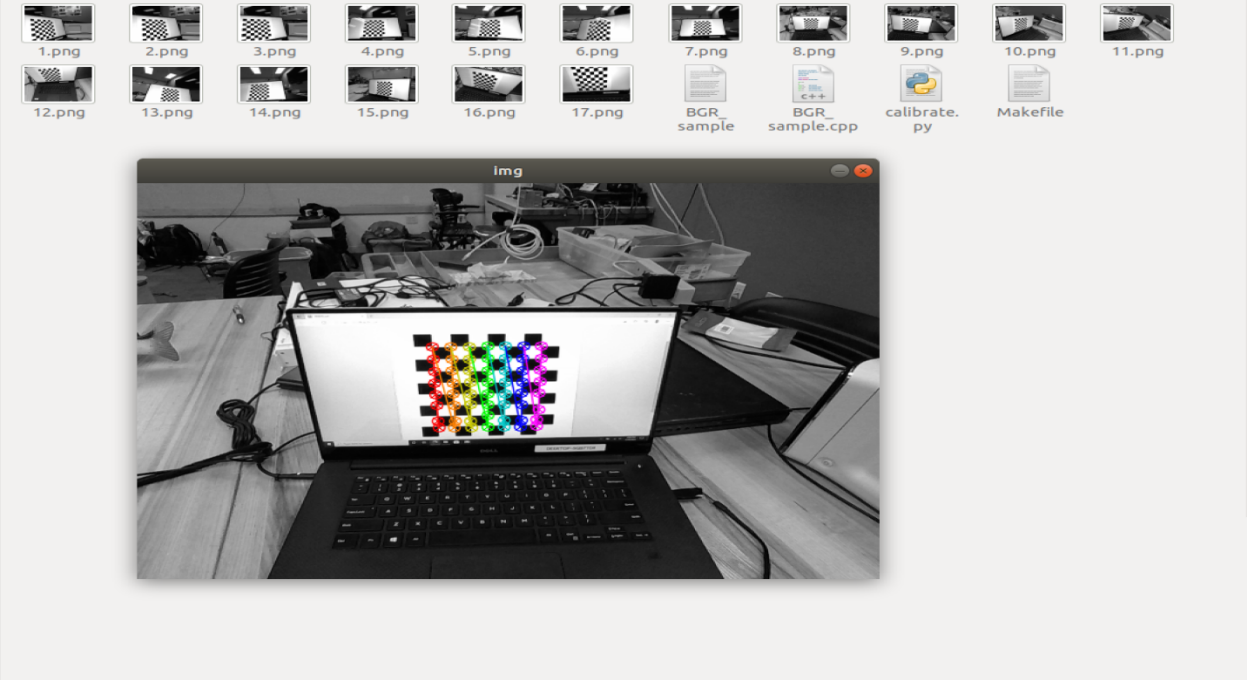
(sited from https://en.wikipedia.org/wiki/Pinhole_camera_model)



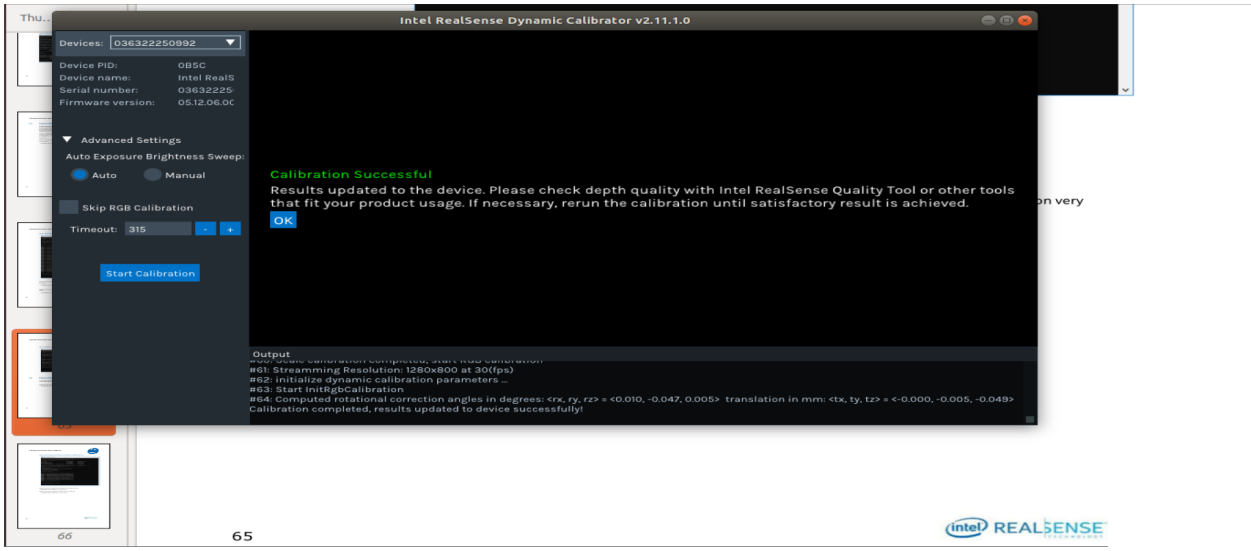
Week 6:

Calibration:

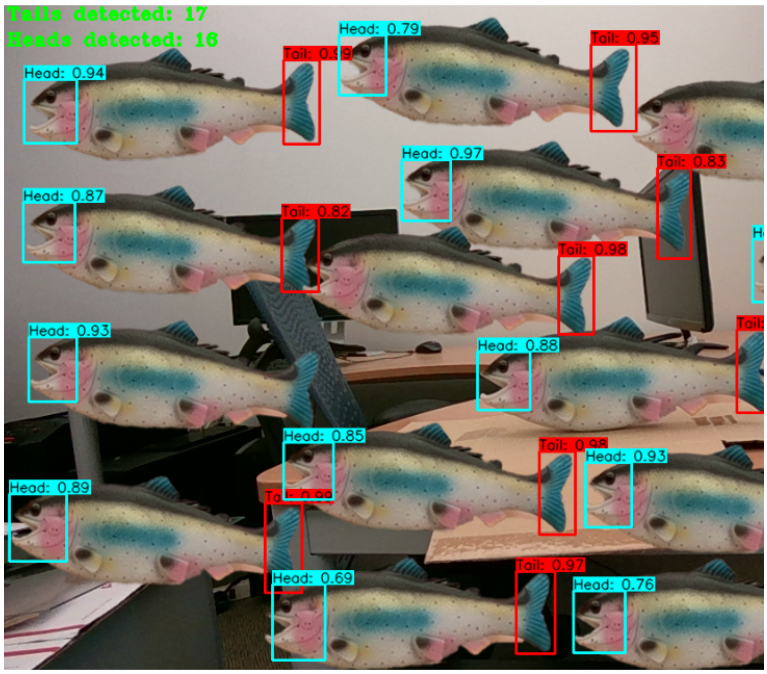
- Calibration by hand using openCV



- Realsense camera calibration



Bounding box labeling:

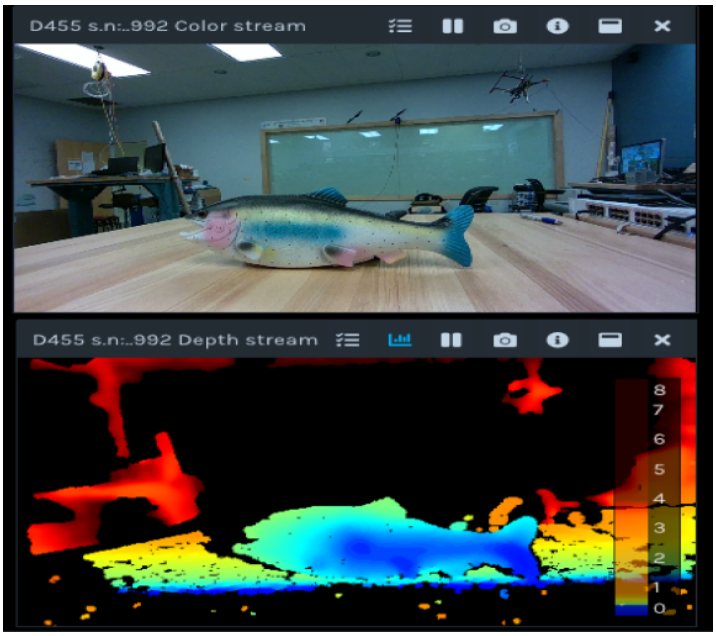


Week 7:

RGB and depth image alignment:

(1) Capture frames: this was a short video of our capturing process https://drive.google.com/file/d/1JX-MMzS4PtMgrXzoo-kmnhkItyThjV4D/view?usp=sharing

(2) RGB image and Depth image in display



Noise filtration algorithm research:

Deliverable: Length detection algorithm succeeds on air images with minimal noise