Alden Kane
CSE 40535
Dr. Adam Czajka
22 November 2019

CSE 40535 Semester Project: Deliverable 2

Project Objective: DDS (Drowning Detection System) - Detection of swimmers from an underwater camera in a swimming pool.
Project Application: Drowning detection utility for autonomous lifeguarding, monitoring of pools with no lifeguards (e.g. apartments, hotels, fitness centers), and redundancy of human lifeguards.

1. Report

A. Methods for Feature Extraction

Features were chosen such that Drowning Detection System (DDS) would detect all swimmers below the surface of the water. Since the objective of DDS is to detect swimmers at risk of drowning, swimmers at the surface of the water are largely ignored. These swimmers are assumed to be non-risks of drowning.

For a robust system, DDS needs to be capable of detecting both (1) moving swimmers underwater and (2) still swimmers underwater. Different features in DDS hone in on classes (1) and (2).

DDS incorporates two features for underwater swimmer detection:

1. Color-Based Swimmer Tracking:

   For color-based swimmer detection, the HSV profile for a number of swimmers in the training set was found using the hsvSelection.py code by Dr. Adam Czajka and Andrey Kuehlkamp, from CSE 40535 practical 1. HSV upper and lower bounds for each swimmer were input into an HSV_tuning.xlsx, then the mean and standard deviation of the upper and lower HSV bounds were calculated. Using these bounds, I ran detect_UW.py on a number of different train videos, adjusting the upper and lower HSV values until an optimum configuration was reached.

   I selected this feature for my algorithm because of its simplicity, yet robustness in detecting swimmers when tuned properly. I was already familiar with this method from Practical 1, so decided to implement it into my project.

   Swimmer recognition based on color tracking is found in detect_UW.py. This script was integrated with motion-based tracking in uw_swimmer_detection.py.

Color-based feature extraction is found in Section 2 of uw_swimmer_detection.py:

```
# Declare hsv upper and lower bounds for color image detection
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
lower = np.array([84, 0, 0])
upper = np.array([111, 190, 150])
binary_image = cv2.inRange(hsv, lower, upper)
```

If DDS was to be implemented as a permanent system in a known swimming pool, this color detection feature could be tuned to account for the specific illumination, water clarity, and depth of the pool. All of these factors affect the HSV profile of a swimmer.

While this feature appears to be robust on for the training and validation data, it is important to note that this data set includes two primary subjects: (1) a 21-year old male of Middle-Eastern and European Descent and (2) a 21-year old female of European descent. The training data for color tracking consisted of subjects (1) and (2), as well as several subjects from YouTube videos of underwater hockey and underwater rugby games — most of these subjects were European (i.e. teams were from Spain, Norway, Great Britain). Future testing data should include subjects of a darker skin pigment (e.g. Africans, African-Americans, Indians). Color bounds ought to be tuned for these subjects — but for this project, their exclusion from training and validation data is a matter of data scarcity.

Relevant frames are shown below:

Figure 1. Color Detection Binary Image

Figure 2. Color Detection Binary Image after Morphological Operations

2. Motion-Based Swimmer Tracking:

   For motion-based swimmer detection, the color image is first converted to grayscale (motion detection is independent of color), and a Gaussian blur is applied to the image to eliminate high frequency noise. The motion-based object detection feature works by taking the absolute difference between the first frame of the current video, and the current frame of the video. The video is then globally thresholded to create a binary image with minimal high frequency noise from moving water and air bubbles.

   This feature is good for my algorithm because of its ability to generalize well; it is mostly independent of illumination, water clarity, etc. and picks up most sizeable moving objects underwater.

   Morphological operations are then applied to this binary image. The binary image after morphological operations is usually noisy and has rather large box-like forms where swimmers are. This is such that when AND'ed with the color binary image, the whole swimmer is preserved.

```
# Initialize first frame. This will only set the first frame once. Now, can compute
difference between current frame and this.
if firstFrame is None:
    firstFrame = gray_Img
    continue

# Now comes absolute difference detection. This is "motion detection"
delta = cv2.absdiff(firstFrame, gray_Img)
thresh = cv2.threshold(delta, 10, 255, cv2.THRESH_BINARY)[1]
```

This motion-based swimmer detection feature is dependent on the quality of the first frame of the video. In my implementation, I reset this with every video to account for potential drift in the camera (my GoPro was on an unsecured tripod and prone to minor movements from water currents). Ideally, this would be set one time in a commercial implementation of DDS, such that no swimmers are in frame (they will eventually move from their initial position, noising up the binary image). I did not have this luxury, as I was collecting my data at Rolf's Aquatic center during open swim. Swimmers at the surface of the water, who are not the aim of detection by DDS, would sometimes appear in the first frame of the recording.

Relevant frames are shown below:

Figure 3. Baseline First Frame for Motion Detection

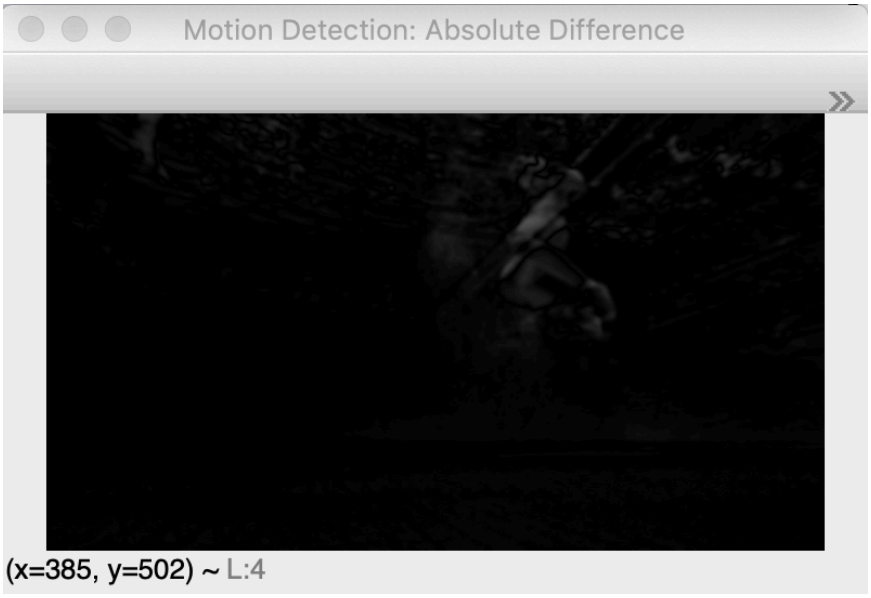

Figure 4. Absolute Difference Between Current Frame and First Frame

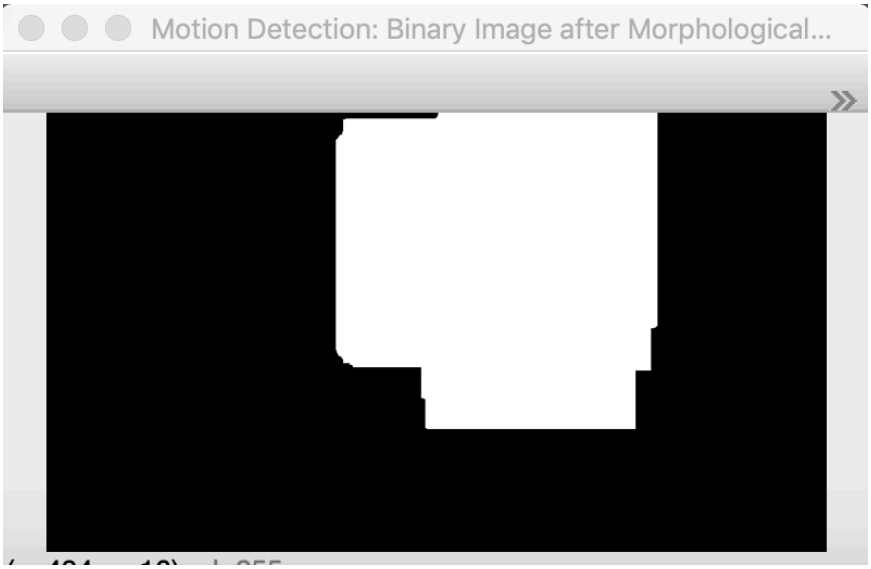

Figure 5. Motion Detection Binary Image with Morphological Operations

B. Choice of Classifier:

DDS uses a binary classifier (i.e. Swimmer(s), not a swimmer) that classifies swimmers based upon the contours and applicable bounding boxes from the bitwise AND'ing of the binary images rendered from features (1) and (2), e.g. figures (2) and (5).

The binary images of both the color-based and motion-based features are designed to be noisy, yet include all true positives. For example, the color-based feature should detect all swimmers in the pool, but will also detect objects of similar HSV profiles (metal grates, certain tiles). While morphological operations filter most of these out, some persist due to their size. Moreover, the motion-based feature will detect objects that are present in the current frame that were not present in the first frame of the video — this works especially well if the frame is calibrated to the pool, with no one initially present. However, the class of moving objects is not only swimmers. Air exhaled from swimmers, the surface of the water, and other pool items (torpedoes, swim aids) all move. While morphological objects filter some of these out, some persist. Each feature should detect all true positives, with some false positives.

The classifier achieves high accuracy with the bitwise AND of these two binary images – – it is unlikely that the false positives from the color-based binary image will overlap with the noise from the motion-based binary image, yet highly likely that the true positive (underwater swimmer) will overlap. For this reason, this binary classifier is sensible and effective for the project.

To account for any noise that might persist after this operation, the classifier only boxes objects > minObjectSize. Therefore, if a swimmer moves far away from the camera, they are unlikely to be detected. In a commercial implementation of DDS, this would be combated by using multiple cameras for DDS, with each focused on certain segments of the pool (with considerable overlap.)

This snippet of code, as well as relevant frames, are shown below:

```
#########################################################
# Section 7: Perform Logical AND'ing of Binary Image, Implement Size Based Object
Detection
#########################################################

# Perform bitwise AND of motion AND color contours
binary_intersection = cv2.bitwise_and(thresh, binary_image)

# Find contours
contours, hierarchy = cv2.findContours(binary_intersection,
                                       cv2.RETR_TREE,
                                       cv2.CHAIN_APPROX_SIMPLE)

# Imshow
cv2.imshow("Logical AND'ing of Motion and Color Contours", binary_intersection)

# Ignore bounding boxes smaller than "minObjectSize". Tuned for optimal swimmer
detection
minObjectSize = 100
```

Relevant frames are shown below:

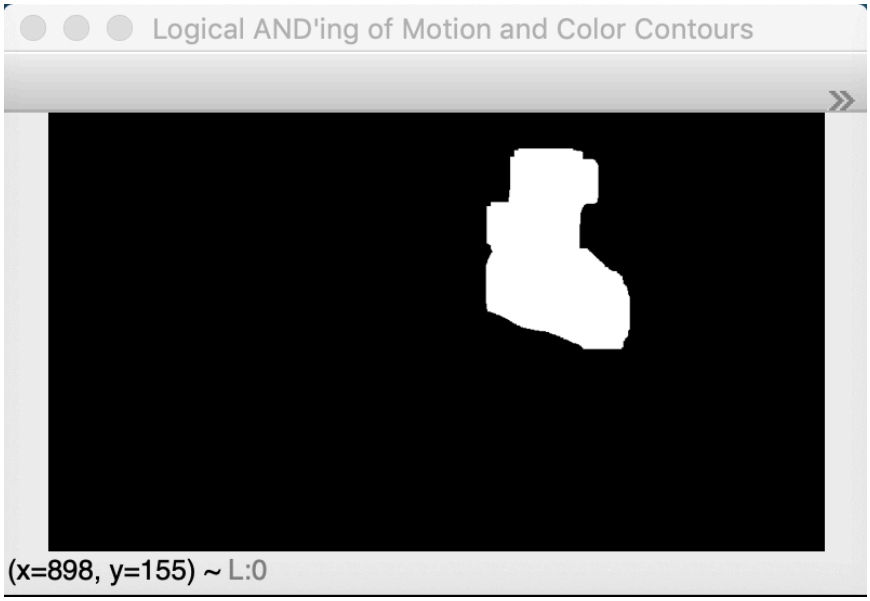

Figure 6. Logical AND of Figures (2) and (5)

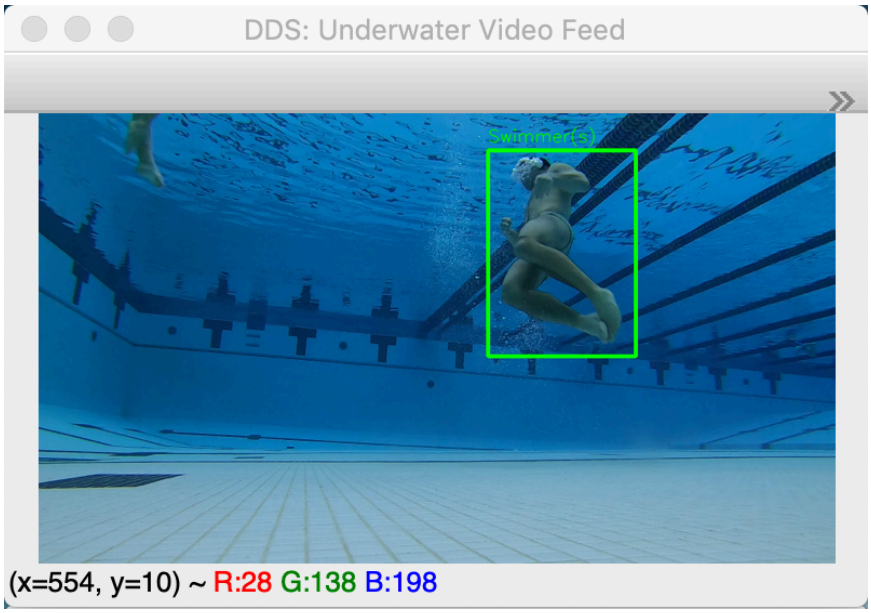

Figure 7. Boxed Swimmer with Contours, Bounding Boxes Drawn from Figure (6)

C. Accuracy of DDS

Accuracy for DDS was defined by the calculation of Intersection over Union (IoU) for random samples from the training and validation sets.

For three training videos and three validation videos, 30 frames were random selected and written to a file. I then drew ground truth bounding boxes in Preview, such that I could calculate the intersection over union of each video. I defined the ground truth bounding box as the smallest bounding box that encompasses the whole swimmer. As all videos were of length 30s and filmed at 30 FPS, my script was choosing 30 frames at random from 900 frames per video. The IoU was only calculated for selected frames that had swimmers in them.

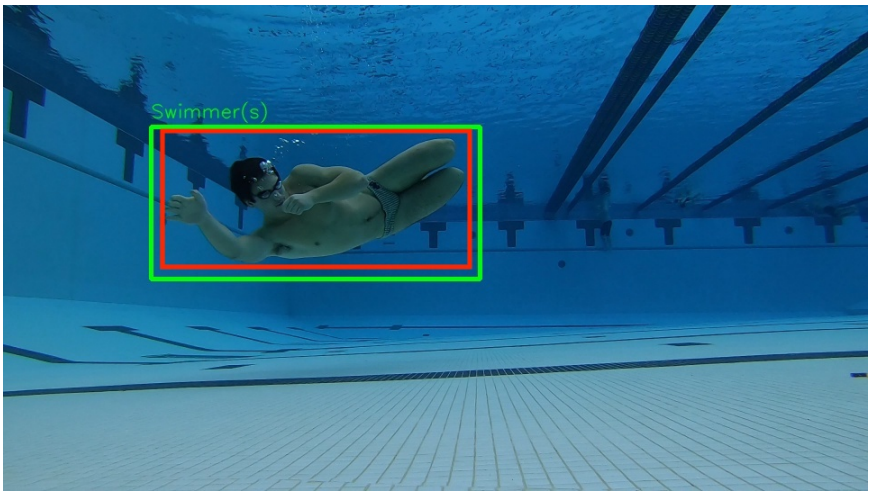

Figure 7. DDS Boxed Swimmer (Green) with Ground Truth Bounding Box (Red)

For the training set, I observed a mean IoU of 0.825 with a standard deviation of 0.111 (0.825 ± 0.111). The three training samples that I calculated IoU for had means and standard deviations of 0.813 ± 0.128, 0.840 ± 0.110, and 0.812 ± 0.089.

For the validation set, I observed an IoU of 0.797 ± 0.134. The three training samples that I calculated IoUs for had IoUs of 0.846 ± 0.118, 0.781 ± 0.108, and 0.782 ± 0.151.

A tabulation of the IoU for the training and validation sets is shown below, in Table 1:

Table 1. IoU for Training, Validation Sets

Data Set	**Video 1 IoU**	**Video 2 IoU**	**Video 3 IoU**	**Overall IoU**
Training	0.813 ± 0.128	0.840 ± 0.110	0.812 ± 0.089	0.825 ± 0.111
Validation	0.846 ± 0.118	0.781 ± 0.108	0.782 ± 0.151	0.797 ± 0.134

Below are samples that have IoUs roughly one standard deviation above the mean, close to the mean, and one standard deviation below the mean:

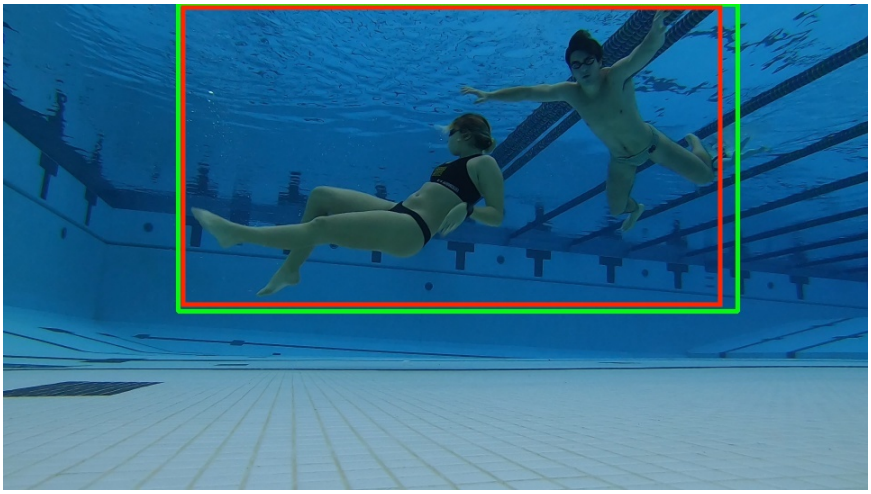

Figure 9. IoU about 1 STDEV above Mean, IoU = 0.934

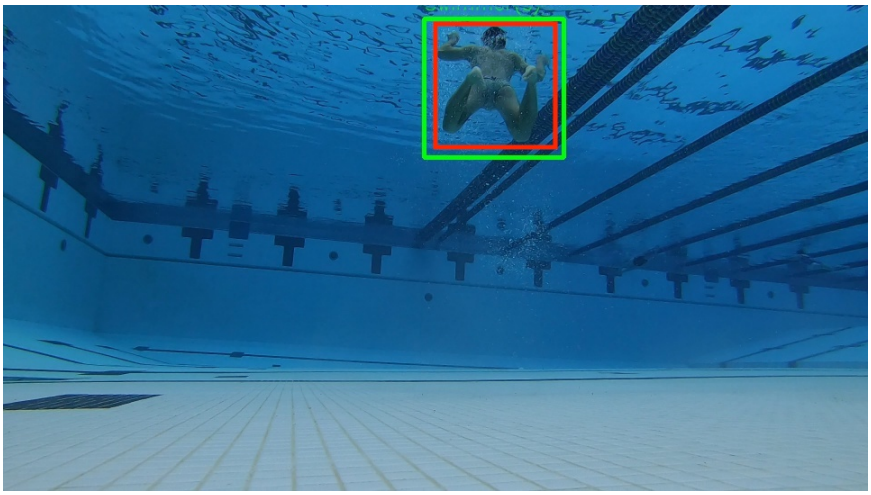

Figure 10. IoU Close to Mean, IoU = 0.796

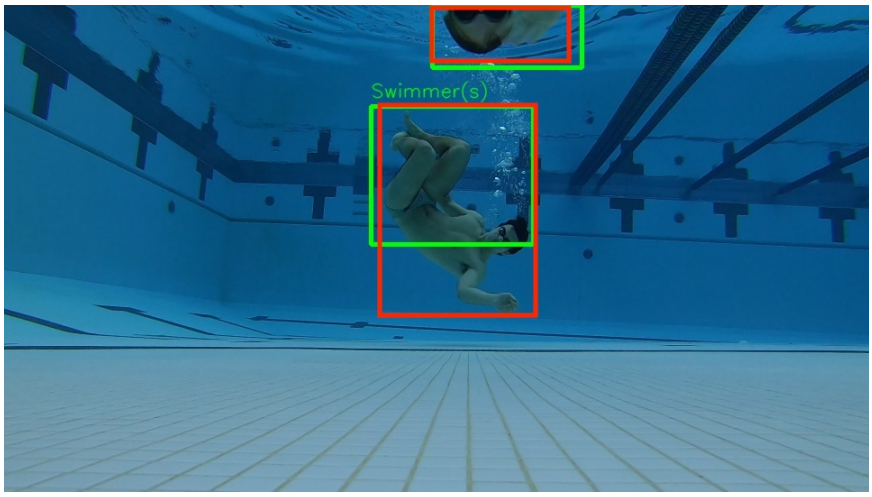

Figure 11. IoU about 1 STDEV below Mean, IoU = 0.717

D. Commentary on Accuracy

DDS, as seen in Table 1, has a good IoU with a fairly well behaved standard deviation. The IoU and standard deviation only varied slightly between the training and validation videos; this is likely because both data sets were collected on the same camera, in the same pool, in similar areas of the pool.

Since the color and motion detection algorithms were tuned with the training data, it is possible that the IoU for the training set is slightly higher than the IoU for the validation set due to this. However, the difference is minimal, and due to the similar conditions of the recording sessions, it can be said that the DDS works very well at Rolf's Aquatic Center at Notre Dame.

Possible sources of error in the accuracy calculations include manual error in drawing ground truth bounding boxes, as well as in measuring the size of the DDS bounding box, ground truth box, intersection, and union. I believe this to be trivial, as these errors would not significantly move the mean or standard deviation.

The use of a neural network as a third feature for DDS could greatly improve the accuracy of the system, as well as its ability to generalize. This would include the extensive annotation of data. With the accuracy achieved by color and motion detection, I believe that use of a neural network in out of scope for this project.

I believe the system will generalize well when tested against other data sets, due to having seen the DDS run on other recordings. Most of these had moving cameras, so are not relevant for motion detection and cannot be used for accuracy metrics.

As part of my test data set and accuracy calculations, I would like to include recordings from the pool at Rockne Memorial, as well as recordings of swimmers with different skin pigments. This will test the robustness and ability to generalize of DDS.

2. Instructions for Running Code

1. Download the `Project` directory from WebDAV
2. Navigate to the `sourceCode` sub-directory
3. Run `python3 uw_swimmer_detection.py`. This is the only script that needs to be run to see all of DDS's features
4. My script is supposed to automatically organize `cv2.imshow()` windows. OpenCV's functionality here is buggy. All of the windows will as nine tabs in one window, and these tabs must be unpacked so that all nine are shown in their correct positions. They do not always unpack in the correct space; the program should be killed and reran if videos are overlapping or occluded. The "unpacking" is shown below:

Figure 12. All Windows in One

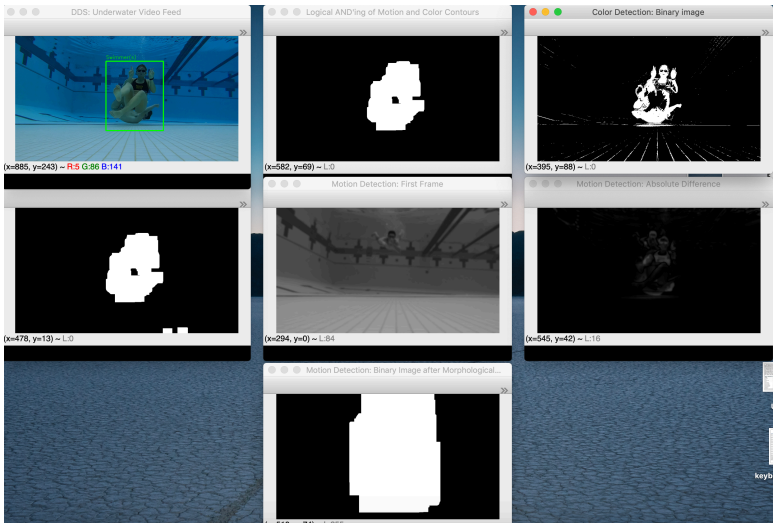

Figure 13. Properly Unpacked Windows

Information on Requirements, Included Files, and Project Structure

- The following Python packages are required to run `uw_swimmer_detection.py`:
  - OpenCV (opencv-contrib-python==4.1.1.26)
  - numpy (numpy==1.17.2)
- I used `Python 3.7.4` to build and run `uw_swimmer_detection.py`
- The .mp4 to be run is housed at `Project/dataSet/swim3/swim3.2-5-of-29.mp4`
- The `accuracy` sub-directory houses random frames for IoU. It will be populated only if `Section 11` of `uw_swimmer_detection.py` is uncommented. I did not include any of my frames for IoU calculations, but am happy to provide my accuracy data sets upon request.
- The `reference` sub-directory hosts the following files:
  - `HSV_tuning.xlsx`: Holds HSV bounds used to tune the color detection feature
  - `int_Over_Union_accuracy.xlsx`: Holds all IoU and accuracy calculations detailed in this report
  - `goPro_Mount.stl`: The GoPro mount that I had 3D printed at Hesburgh Library to record data with
  - `blue_Pool_Tile.png`: Sample HSV profile of pool tile, from running `hsvSelection.py`
  - `blue_Suit_Swimmer.png`: Sample HSV profile of female swimmer with a blue swim suit, from running `hsvSelection.py`

3. References and Resources

The following resources and references were used in completing this project:

- Dr. Adam Czajka was especially helpful in his direction and expertise for this project, through consultation at office hours for CSE 40545.
- Dr. Adam Czajka and Andrey Kuelkamp's `colorTracking.py` script for the University of Notre Dame's Fall 2019 CSE 40535/60535 course was used to help design my color detection feature.
- Adrian Rosebrock's "Basic motion detect and tracking with Python and OpenCV" tutorial at https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/ was referenced in designing my motion detection feature.
- Adrian Rosebrock's "Intersection over Union (IoU) for object detection" tutorial at https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ was referenced for my IoU calculations.
- All course materials for University of Notre Dame's Fall 2019 CSE 40535/60535 course were used in designing this system.