PoolTrackerDDR

Jack Dangremond, student: Electrical and Computer Engineering, Carnegie Mellon University
Adithya Raghuraman, student: Electrical and Computer Engineering, Carnegie Mellon University
Karnkumar Dalmia, student: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of tracking and recording a swimmer's workout in real-time, using aerial video footage of a pool. This system consists of a position detection as well as stroke classification, allowing for distances, strokes, and splits to be recorded. The backend of this system will consist of Python scripts utilizing OpenCV and OpenPose and will be responsible for all the video processing and tracking that that is needed. This backend will send data to a web application, which will be stored to be displayed and analyzed by the user.**

***Index Terms*—Object Detection and Tracking, Supervised Learning, Video Processing**

I. Introduction

THE project that we chose to work on is a tool that would allow swimming coaches to comprehensively track the progress of swimmers over the course of a workout. One of the most difficult things for coaches and/or individual swimmers is to reliably record all the records of a given workout session. Therefore, we believe that such a tool would be immensely helpful for coaches, swimmers, and teams with a limited coaching bandwidth.

At this point in time, there are no commercial technologies available that achieve what our project will. Professional swimmers can afford to have personal coaches to watch and record their workouts, but that isn't available to the greater part of the swimming population. There are also touch pad timing systems that can very reliably record splits, but these do not differentiate between different types of touches and are therefore not useful for recording times in a workout setting.

The goal for this project is two-fold. Firstly, we aim to get swimmer detection and tracking with a reasonable degree of precision. Specifically, we aim to have our tracking algorithm to track lap times and rest times to within 0.5 seconds of the ground truth. Secondly, we aim to have our stroke classification algorithm to differentiate between the four basic strokes (freestyle, backstroke, breaststroke, butterfly) with an accuracy of 80%.

II. Design Requirements

The most crucial aspect of this project is that we are able to accurately detect when a length swum. We feel that this can be done with 100% accuracy since it has a very high margin for error. The larger challenge will be to record accurate times for each length, because this requires finding the exact instance when a swimmer touches the wall. For a practice environment, we feel that 0.5 second precision is sufficient for splits, which is why we have made this a requirement. We validate our system by recording an hour-long swim workout and comparing our results to the ground truth times.

The second goal that was mentioned was to classify the stroke that the swimmer is performing with an accuracy of at least 60%. We believe that this will the hardest part of this project since there are so many variables such as (joint angles and individual swimmer style nuances).

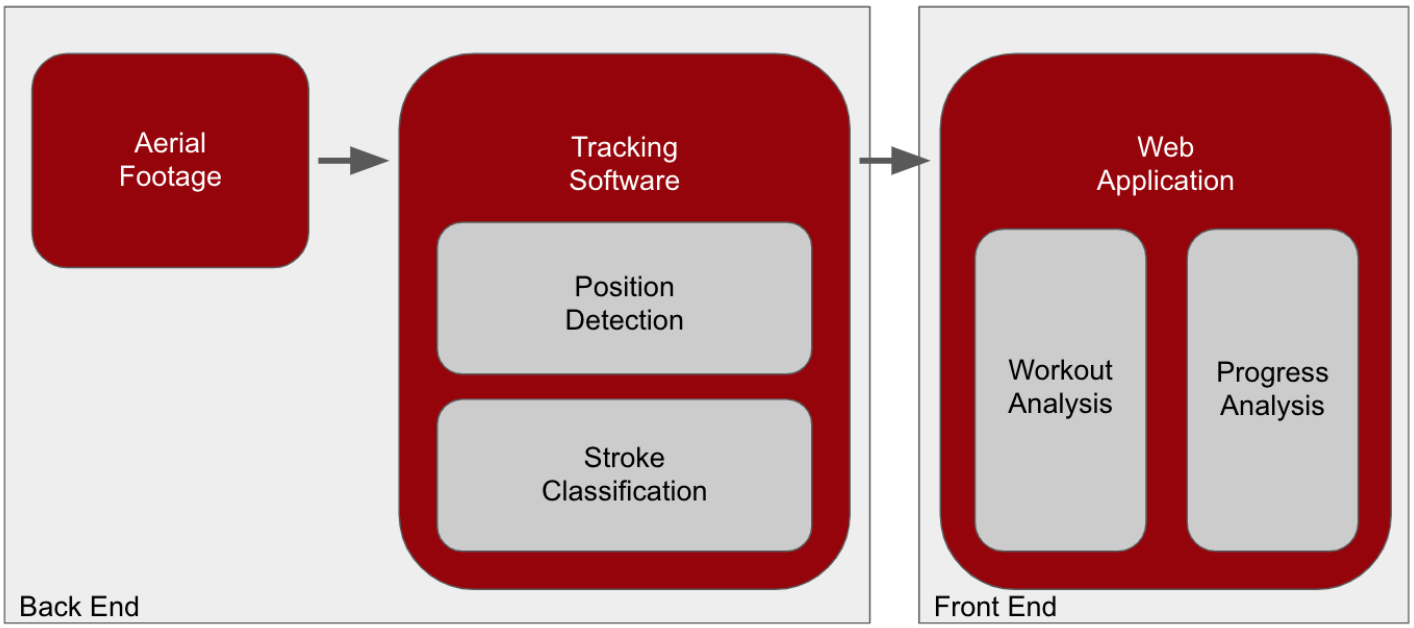


Fig. 1. Overview of PoolTrackerDDR system archicture.

III. Architecture and/or Principle of Operation

Our system consists of two, main components: backend software which performs all of the video processing and tracking, and a frontend web application which receives and displays all of the collected data.

The backend of our system has two major parts, the aerial video footage and the software which processes this. In an actual product, we would want the footage to be live-streamed for real-time processing. Due to the fact that we were unable to find a pool that would allow us to install a camera, we will be using pre-recorded videos that will still be processed in real-time by the tracking software. This will replicate the functionality of the product that could eventually be used by consumers without the expense or logistical complications of installing hardware permanently.

The tracking software has two main functions. The first is to detect and track the position of a swimmer in the pool, and the second is to identify which stroke they are swimming on a length-by-length basis. The input will be the raw video footage, as well as a description of features of the pool and lanes that need to be tracked, which will be provided by the user.

Technical details for how we plan to implement the detection and classification systems will be provided later in this report. The results of these two sub-systems will be packaged up and sent over to the web application upon the completion of each length. These results will be stored in a MySQL database to be reviewed and analyzed within the web application. More specific details of the interface between the tracking software and web application will be provided in Section V.

While not the most technically challenging part of this project, the web application will the part that makes the data we're collecting meaningful to coaches and swimmers. We will be providing tools to visualize data both within a workout and across workouts over a span of time. By storing everything in a database, we can easily extract very specific data that a user might be interested in (i.e. the average time that a swimmer is able to hold on a 100 freestyle in practice) and display it in a graphical way.

IV. System Description

As described earlier, the overall system here consists of two main parts: backend software which performs all of the video processing and tracking, and a frontend web application which receives and displays all of the collected data. In this section, we will discuss each of these and the interfaces between them in much greater detail.

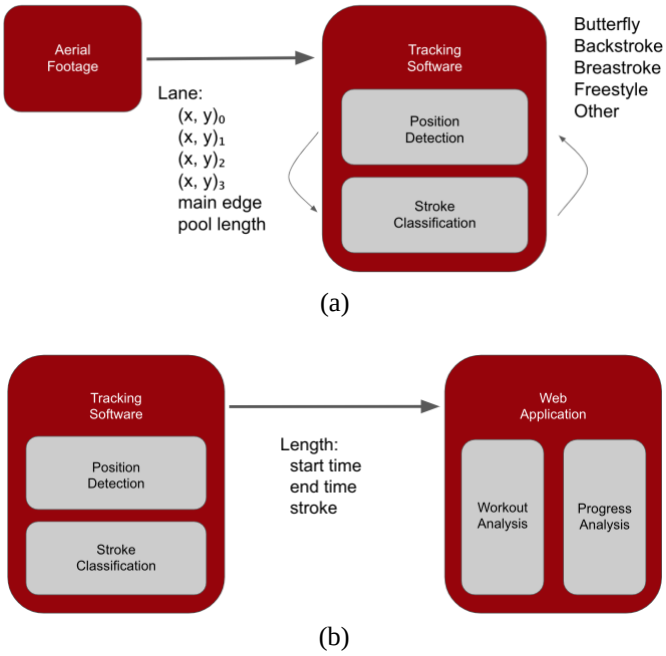


Fig. 1. System overview. (a) backend software. (b) interface between backend and web application.

Let us first look at the workflow of our project. This workflow, as described in Figure 2.a describes the way in which the front end and the back end communicate with each other. The frontend web application is the point of entry for a user. The frontend then establishes a connection with the backend (a camera + processing unit). This backend then captures a still shot of the pool and sends it to the front end, where the user then has the responsibility to demarcate the lane edges. We thought that it would be best to have this as the responsibility of the client since our product is now a lot more flexible as there are no further restrictions on the real-world separation of lanes. Once the lane edges are established, the backend software detects when a swim session is complete and sends data to the front-end web app to be displayed to the client. The user can then choose from a variety of visualization settings to get the most out of the data that has been received.

A. Backend

The backend is the workhorse of our project. This backend is responsible for doing the actual swimmer detection, tracking, and stroke identification. As was mentioned in Section III, the main input to the backend will be the raw aerial footage of the pool. To limit the scope and make our system more robust, we have also decided that the user will manually demarcate lanes to be tracked within the web application. This will involve a "handshake" between the backend and frontend that must occur before any processing begins, as described below:

1. The backend and web application establish a connection via an initial HTTP request from the backend.
2. The backend captures a still shot of the pool and sends it to the web application. We are requiring that the camera remains perfectly stationary, so the pool will be positioned exactly the same in all following frames.
3. The user manually marks up the lanes that need to be tracked, first by outlining the four edges of each lane (as shown in Figure 3), then marking the line that represents the "main" pool edge where workouts will start from.
4. These boundaries, as well as the length of the pool (25y, 25m, or 50m), will be sent to the backend.
5. The tracking software analyzes the aerial footage in real-time and sends results to the web application upon the completion of each length.
6. The web application receives this data and stores it in a MySQL database to be visualized for user analysis.

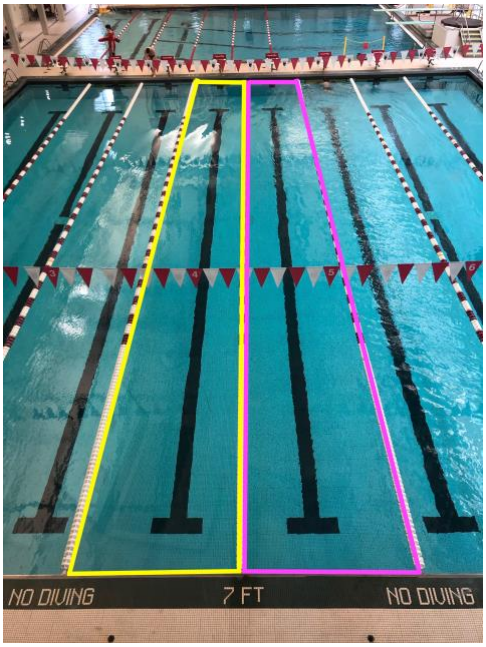
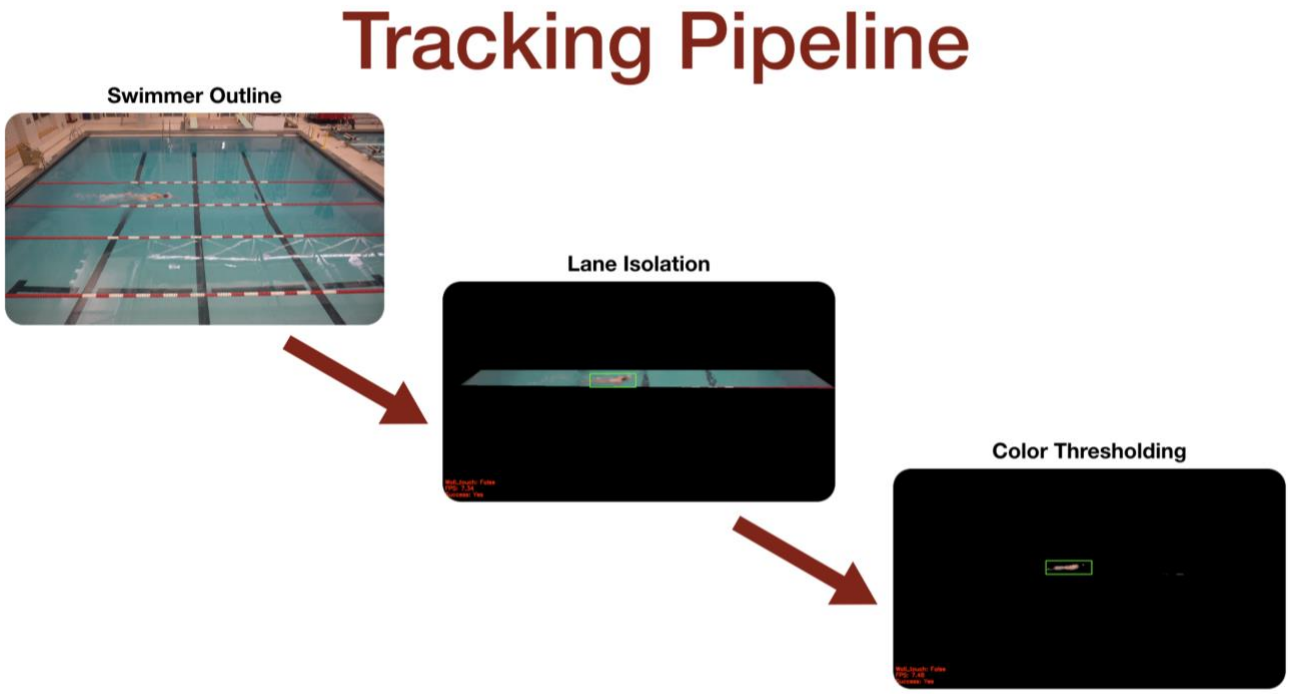


Fig. 2. Screenshot of aerial footage with lanes of interest marked up.

After the video footage and lane information are provided, the backend can get to work tracking the workout. The two sub-components of this tracking software are position detection and stroke classification.

The first of these systems aims to locate and track the position of a swimmer in the pool. We have chosen to limit PoolTrackerDDR to tracking just one swimmer in a lane at a time. We felt that allowing multiple swimmers in a lane would add too much complexity and noise to our input footage. Here is a description of the algorithm that we use to do our tracking and detection. Firstly, we subtract from our footage all of the



background information. We do this by taking an empty (with no swimmers) shot of the pool and use that as a mask. Once we have a bounding box on the swimmer, we perform a color thresholding to get rid of noise from the surrounding pool. We generate a distribution of the pixel intensities within the bounding box and in subsequent frames, move the bounding box to the location where the pixel intensities match this initial distribution.

The second component that we have on our backend is the stroke classification system (shown on next page). The first step of this process is isolating the swimmer from the rest of the pool. We do this by using the same tracker as described above and taking only the part of the frame which is in the ROI. For this segment of the image, we run a background subtractor to obtain the shape of the swimmer's body in the water. This is run through a pre-trained feature extractor and then through our classifier which outputs one of free, fly, back, or breast. Because we weren't able to detect limbs, we weren't able to encode a swimmer's motion over a sequence of frames. This means that our model was trained on individual frames and classifies on a frame-by-frame basis. In order to best determine which stroke is being performed, we take the mode of the outputs over a sequence of
frames.

B. Frontend

The front end of this project serves two main purposes. Firstly, the first end is what establishes a connection with a backend processor and sends user demarcated information regarding lane edges. Subsequently, the front end becomes a tracker that stores the data that it receives from the back end and visualizes it to the user.

The front end here is implemented as a web application written in Django, a python framework. Django has various features that fit out need greatly such as high data processing rates and highly secure communication protocols. This Django web app is hosted on an AWS EC2 instance. All of the data that is received from the backend device will be stored in a MySQL database for easy lookup.

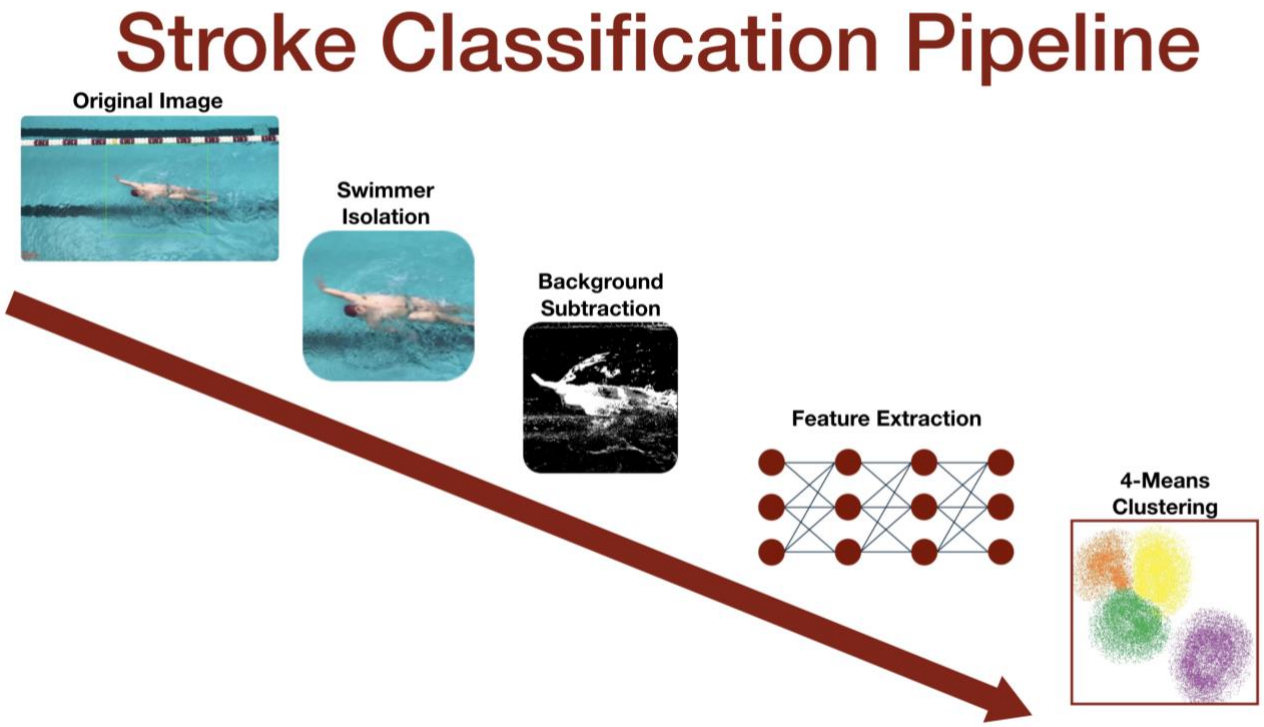
In terms of analyzing and visualizing the data that's received, our web application will provide two main functionalities. The first of which is being able to look at a swimmer's progress in an individual workout, and the second is to look at a swimmer's progress over multiple workouts. Both of these views will provide tools to create in-depth visualizations of many trends, including trends in splits, distances, and strokes. With these tools, a coach or a swimmer will be able to see clearly where they are progressing or regressing and respond accordingly.

C. Validation

Because of the nature of our project, we are limited in the amount of data we can collect and validate against. However, we think we can enough manual testing to have a high degree of confidence in the effectiveness of our project.

To test PoolTrackerDDR, we will record a workout with just one swimmer in a lane. We will record the ground truth of this workout (distances, strokes, and splits), and compare this to the result found using our tool. The following are the metrics against which we will evaluate our final project:

- Get lap time and rest time within 0.5 second error
- Detect laps completed with 95% accuracy
- Classify the correct stroke for at least 60% of laps (this number is fairly low because freestyle and backstroke have extremely similar limbic characteristics and can even be difficult for humans to classify).



V. Project Management

A. Schedule

Our tentative schedule for the implementation of this project is attached in Figure 4. It should be noted that while there are dependencies within each component, there are very few across components. We have purposefully designed our system this way to avoid scheduling backlogs due to one task being delayed. The only time where this is not true is near the end of the project, when we will be integrating all of the components. However, we have already mitigated scheduling risks arising from this by implementing the interfaces between components first.

B. Team Member Responsibilities

Given that there are three main components to our project, (web app frontend, the detection + tracking component of our backend and the stroke classification component of our backend) we deemed it most appropriate for each of the three members of our team to lead the development of one of the components.

As stated earlier, Jack and Adithya are responsible for the actual web-application as they both have extensive knowledge in that domain space. Adithya is also going to be working with developing the detection and tracking algorithm along with some assistance from Jack. Karn will be working on implementing the spatial parallelism of swimmer detection as he has had the most experience with GPU programming, though the GPU component of the project is still tentative. All three team members will work on coding the actual stroke classification and split calculations, as that constitutes the bulk of the technical portion of the assignment.

C. Budget

We have less use of the budget allocated to us since we have already collected high quality videos, and all our necessary software, OpenCV and OpenPose, are completely free. That said, should we go through with the GPU parallelism, we plan to rent a 4-core compute instance from AWS for $0.900/hr in order to spatially decompose the video. Additionally, there may be a little overhead cost in deploying the actual web app. We guess that these costs will be well under the actual $600 allocated to us.

D. Bill of Materials

Material	***Cost***
iPhone 8 Plus	*Already owned*
2017 MacBook Pro	*Already owned*
Python 3	*N/A*
OpenCV	*N/A*
TensorFlow	*N/A*
OpenPose	*N/A*
Django 2	*N/A*
MySQL	*N/A*
AWS EC2 instance	*Free instance*

E. Risk Management

One of the biggest risks of our project is not finishing tasks in a timely manner. This tends to be quite problematic when tasks are not parallelizable (one task must complete prior to commencement of the next), as an entire schedule bottleneck occurs. We plan to mitigate the risks in the following ways.

First, we hope to have backup plans should one particular task not follow through in a timely manner. Moreover, it is also possible to do weekly progress checks, and see if we are on schedule. Should there be a lack of progress on particular week, we can reschedule subsequent tasks well ahead of time, and effectively avoid huge traffic jams with scheduling. The TeamGantt software is a very helpful calendar tool to visualize tasks in an iterative way and can help us better parallelize tasks that are not dependent while working to mitigate bottlenecks for the sequence of tasks that are. One of the largest risks that we have in terms of scheduling is integrating the different components together. We have mitigated this risk by implementing the interfaces first such that the frontend and backend can be developed in parallel. While they will rely on each other in the end, we can build and test each component independently.

VI. Summary

The following are our metrics, and a brief description of our outcome:

- Get lap time and rest time within 0.5 second error
  - We were able to achieve this metric.
- Detect laps completed with 95% accuracy
  - We were able to achieve this metric.
- Classify the correct stroke for at least 60% of laps (this number is fairly low because freestyle and backstroke have extremely similar limbic characteristics and can even be difficult for humans to classify).
  - We were not able to achieve this metric but did get results suggesting that stroke classification could be improved with more data and more advanced methods.

Even though our project didn't meet all of the initial metrics, we still feel that this project was a success in most ways. One of the biggest factors that prevented us from reaching the stroke classification metric was the lack of data that we had. This was due to the fact that we weren't allowed to mount a camera above the CMU pool, and other pools nearby in Pittsburgh never responded to us reaching out. If we had more aerial swim footage, we would have been able to train and classify based on sequences, which would likely boost our classification accuracy.

A. Lessons Learned

One of the biggest lessons that we've learned this semester is that we should have spent more time on our highest risk items first. In our application, stroke classification was the component that was giving us the most difficulty. We were initially planning to encode limbic movement for training and classification but weren't able to make this work due to the noise of the pool water. Eventually we were forced to abandon this idea and had less than two weeks to figure out a new mechanism for stroke classification. To any future groups working on any type of video classification, we would highly recommend implementing a couple methods in parallel to see which works the best.

VII. References

Cao et al. OpenPose: realtime multi-person 2d pose estimation using Part Affinity Fields, 2018

Lukezic, Alan et al. Discriminative Correlation Filter Tracker with Channel and Spatial Reliability, 2016

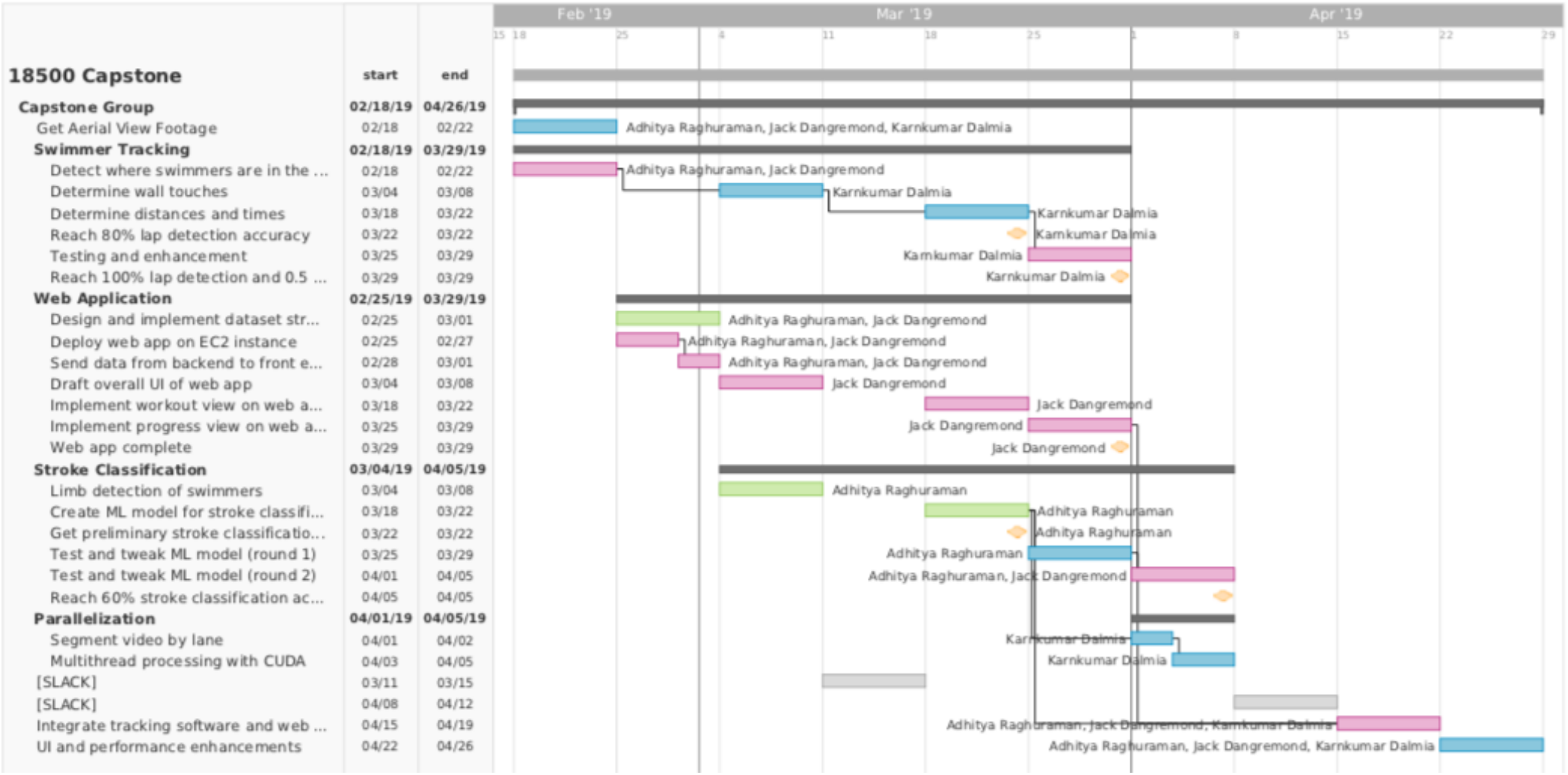


Fig. 2. Schedule of tasks for team C5.