Detailed Design and Analysis Plan

Target Acquired

Justin Campbell, Nicholas Aufiero, Preston Hart, Ryan Ylagan, Rohan Wariyar

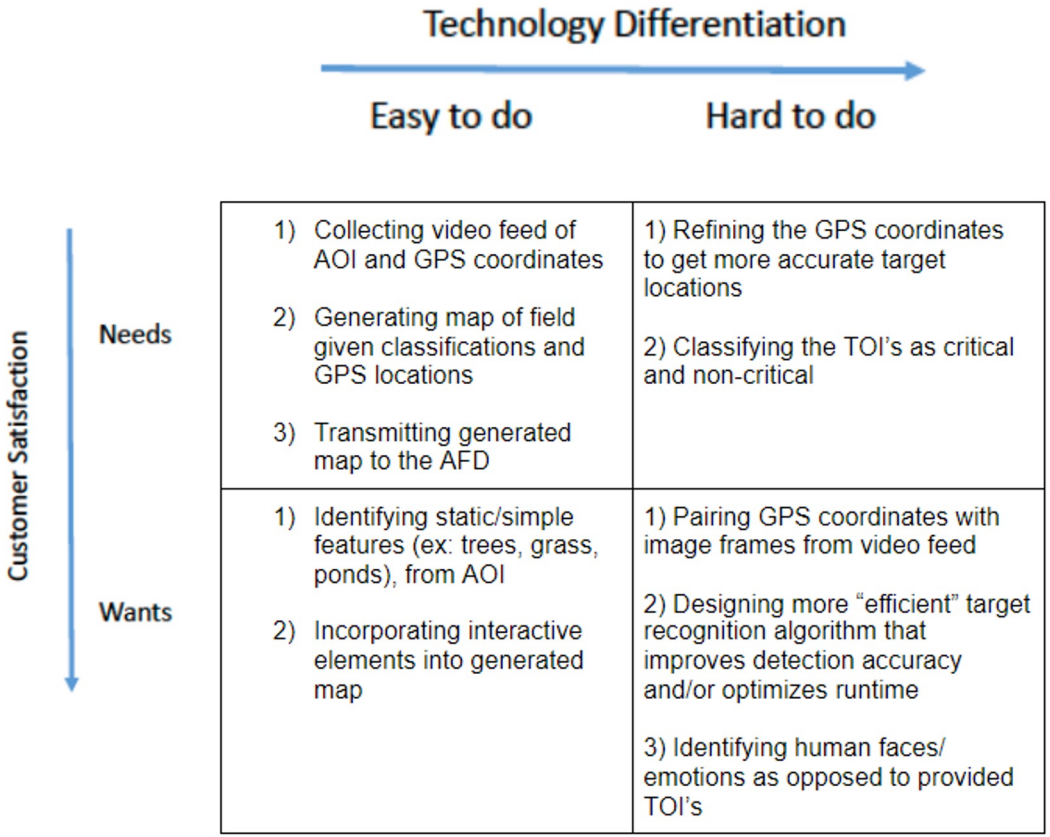
Recap of Requirements and Deliverables

Need statement: The ASE Design Team and Austin Fire Department need a software able to recognize targets of interest, discern between critical/non-critical targets, and deliver accurate GPS coordinates within a generated map.

- Guaranteed Deliverables
  - Image recognition software able to recognize and classify TOI's
  - Targets of Interests' GPS location & classification (CAT/CRT)
  - Generated map of area of interest
- Stretch Goals
  - An interactive map with that is paired with the respective GPS coordinates
  - Image recognition software that can identify human emotion rather than just our preset TOI's
  - A more efficient algorithm that improves on accuracy and required search time
  - An algorithm that is capable of identifying targets on more complex terrain

(Nicholas)

Recap of Decision Matrix

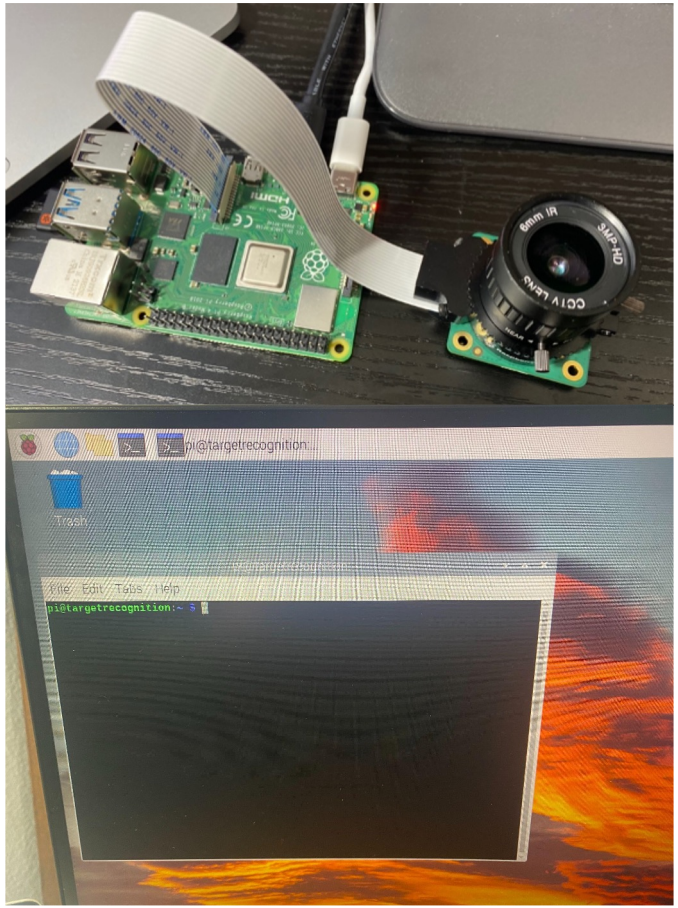


	Easy to do	Hard to do
Needs	1) Collecting video feed of AOI and GPS coordinates 2) Generating map of field given classifications and GPS locations 3) Transmitting generated map to the AFD	1) Refining the GPS coordinates to get more accurate target locations 2) Classifying the TOI's as critical and non-critical
Wants	1) Identifying static/simple features (ex: trees, grass, ponds), from AOI 2) Incorporating interactive elements into generated map	1) Pairing GPS coordinates with image frames from video feed 2) Designing more "efficient" target recognition algorithm that improves detection accuracy and/or optimizes runtime 3) Identifying human faces/ emotions as opposed to provided TOI's

(Justin)

Detailed Design

Assembly and Set-up of Imaging System

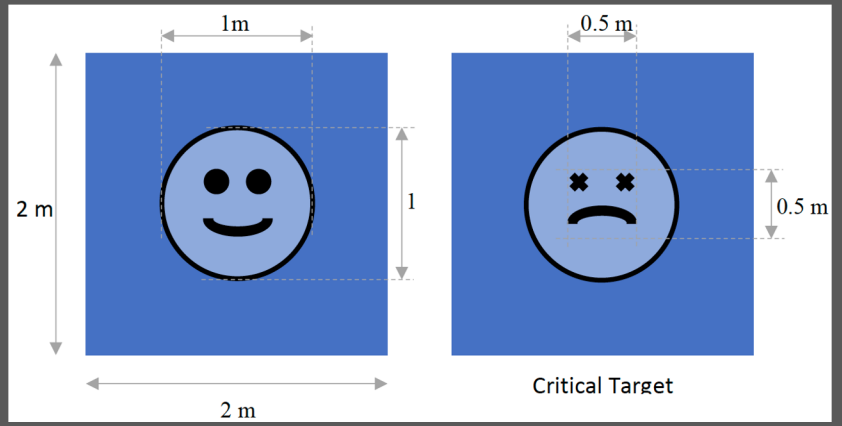


- Debian Linux OS was flashed from user guide to SD card and inserted into SD card port
- Power supplied to Raspberry Pi using USB-C interface
- Micro HDMI to HDMI cable connected for display of Raspberry Pi computer to connected device
- Peripheral devices (mouse and keyboard) connected through USB port
- CSI interface used to connect the Raspberry Pi High Quality Camera (HQC)

(Justin)

Building Mock-up Targets for Training Images

- Mock-up targets were developed using equipment shown to the left
- Effectively, the targets were created from cardboard cutouts and the brown background color is used to approximate the color of plywood of actual targets in the AOI
- The cutouts were then placed on silver and blue tarps to resemble mission conditions
- The dimensions of the cutouts closely resemble the dimensions of the exact targets



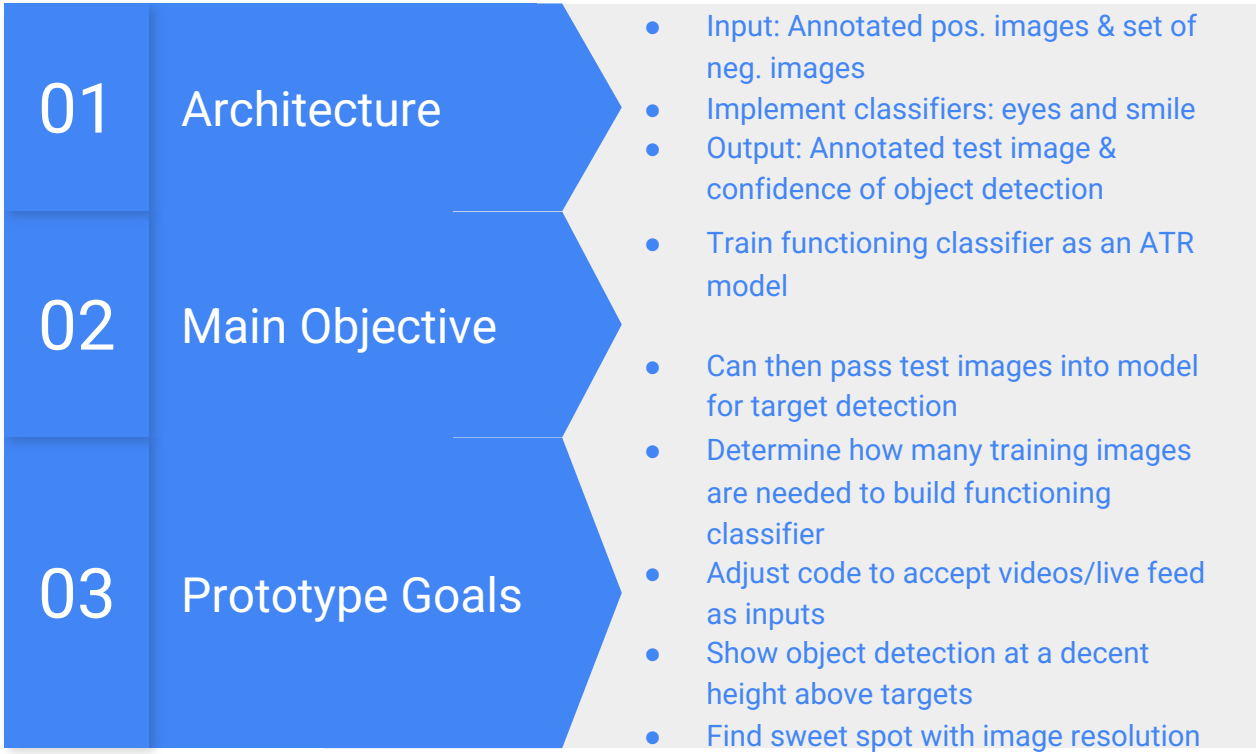
(Justin)

Collection of Training Images: Preliminary Iterations

- Command Line Interface on Raspberry Pi Terminal used to capture images of targets with a high resolution of ~12.3 MPx
- In first procedure, a small volume of images of both critical and candidate targets (positive images) were collected in house garage with moderate lighting and severe height restrictions giving way to inadequate mission representation (lighting and pixel number of targets)
- In second procedure, a much larger volume of positive images (100) were collected in addition to a reasonable number of background training (negative) images
- Here, images were collected from fourth story building floor in environment with natural lighting
- Ultimately, both of these batches of training images are being used to train OpenCV Haar Cascade Algorithm Prototypes

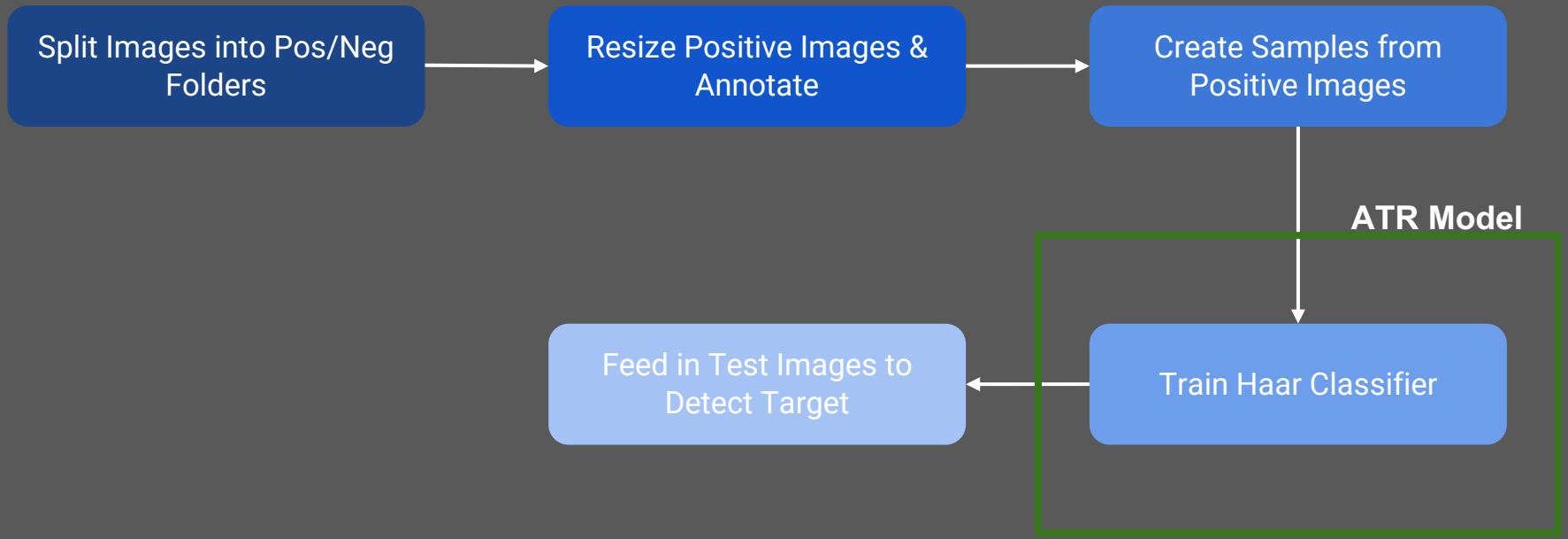
(Justin)

OpenCV Haar Cascade Prototype 1



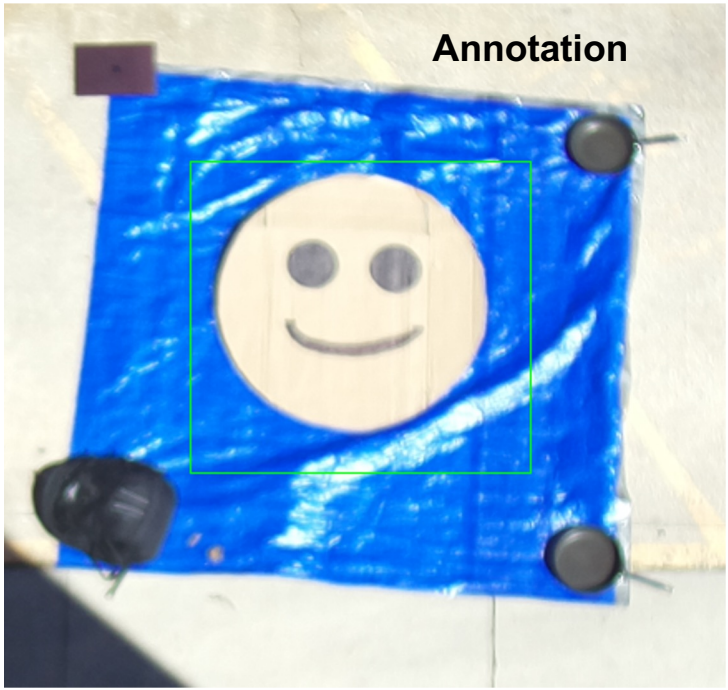
(Preston)

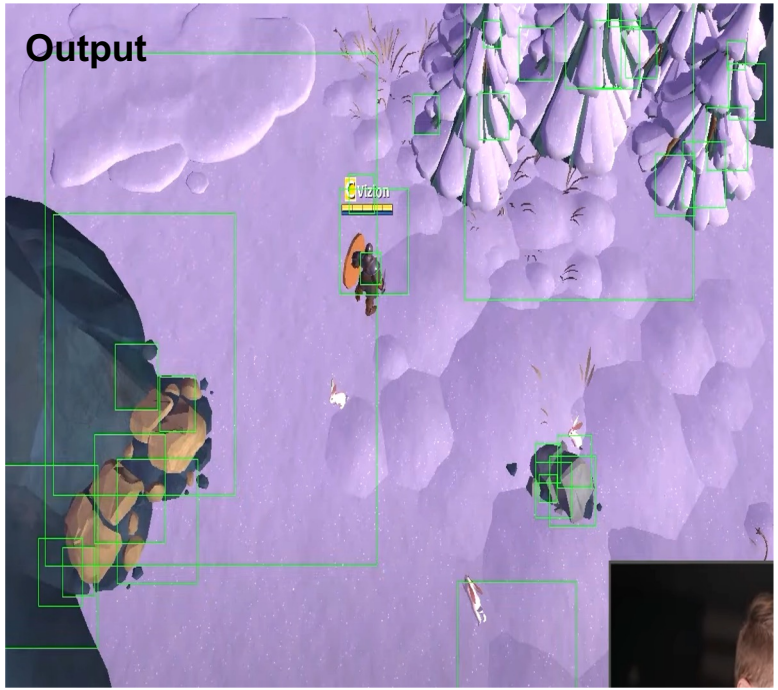
Flowchart of Haar Cascade Classifier



(Preston)

```
C:\Users\prest\Desktop\UT Stuff\Spring 2022\Senior Design\opencv\build\x64\vc15\bin>opencv_annotation.exe --maxWindowHeight=100 --annotations=pos.txt --image
s=pos_smiley
' mark rectangles with the left mouse button,
' press 'c' to accept a selection,
' press 'd' to delete the latest selection,
' press 'n' to proceed with next image,
' press 'esc' to stop.
^C
C:\Users\prest\Desktop\UT Stuff\Spring 2022\Senior Design\opencv\build\x64\vc15\bin>opencv_annotation.exe maxWindowHeight= 100 --annotations=pos.txt --images
=pos_smiley
```





(Preston)

Retrieving data from flight controller: Proof of concept

01 **Architecture**

- Input: Build virtual flight controller using a Docker container
- Output: Grab flight information from the virtual flight controller using Pymavlink

02 **Main Objective**

- Proof of concept for sending flight information from the flight controller to the Raspberry Pi

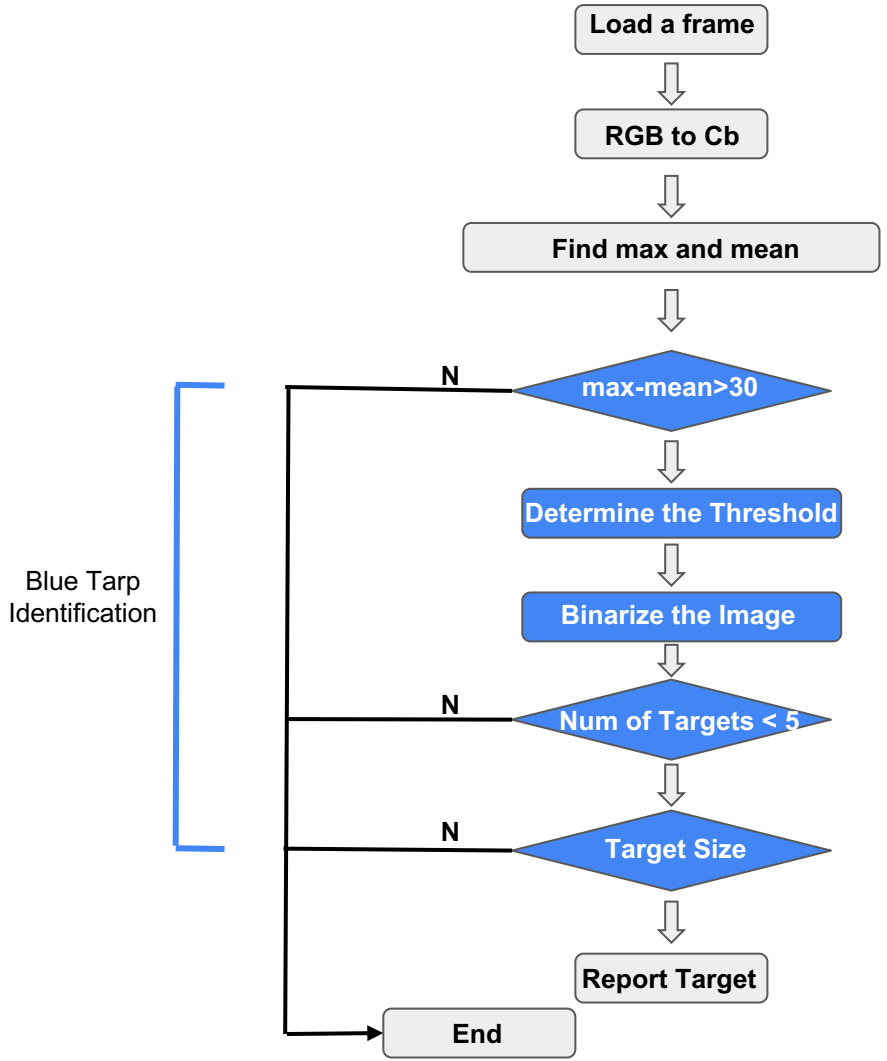
03 **Prototype Goals**

- Have a functioning script when the raspberry Pi is implemented into the UAV

(Nicholas)

RBG-Based Method Design

(Ryan)



Map Generation Algorithm

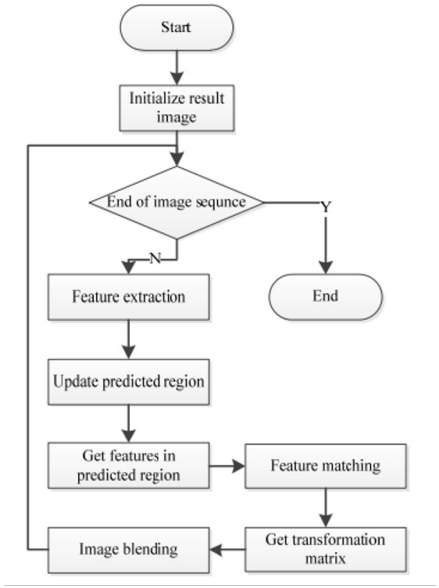


Figure: Flow Chart/Block Diagram for our Aerial Image Stitching algorithm and example output

What algorithms/methods were considered for our Map Generation?

- Utilizing Google Maps API
  - Ruled out due to cost
- Aerial Image Stitching
  - There is an open source code for a similar algorithm that utilizes OpenCV modules and libraries
  - Written in C++, so we would need to port it to python to operate on our Raspberry Pi

(Ryan)

Analysis Plan

Analysis Plan

01 **Evaluating Classifier**

- Upwards of 85% confidence in detecting targets in image
- Upwards of 90% detection accuracy in test images set

02 **Runtime**

- Objective detection, recognition, and identification runtime shall not exceed 48 seconds per positive image
- Threshold detection, recognition, and identification runtime shall not exceed 72 seconds per positive image
- Can then pass test images into model for target detection

03 **Evaluating Combination of Methods**

- Looking to see increased detection accuracy & decreased detection runtime

(Rohan)

Tasks/Requirements Update

Completed/In-progress Tasks:

- Collect training images for prototype image recognition algorithm (Completed)
- Create a working prototype for our algorithm using images (In Progress)
- Create an algorithm for our Map Generation (In Progress)
  Collect training images for our final model, taken
  of similar features (In Progress)

Requirements Challenges:

- Problems with the image recognition algorithm not correctly identifying training images
- Due to or algorithm not behaving correctly, we do not have the data to quantify whether our model meets certain requirements

(Ryan)

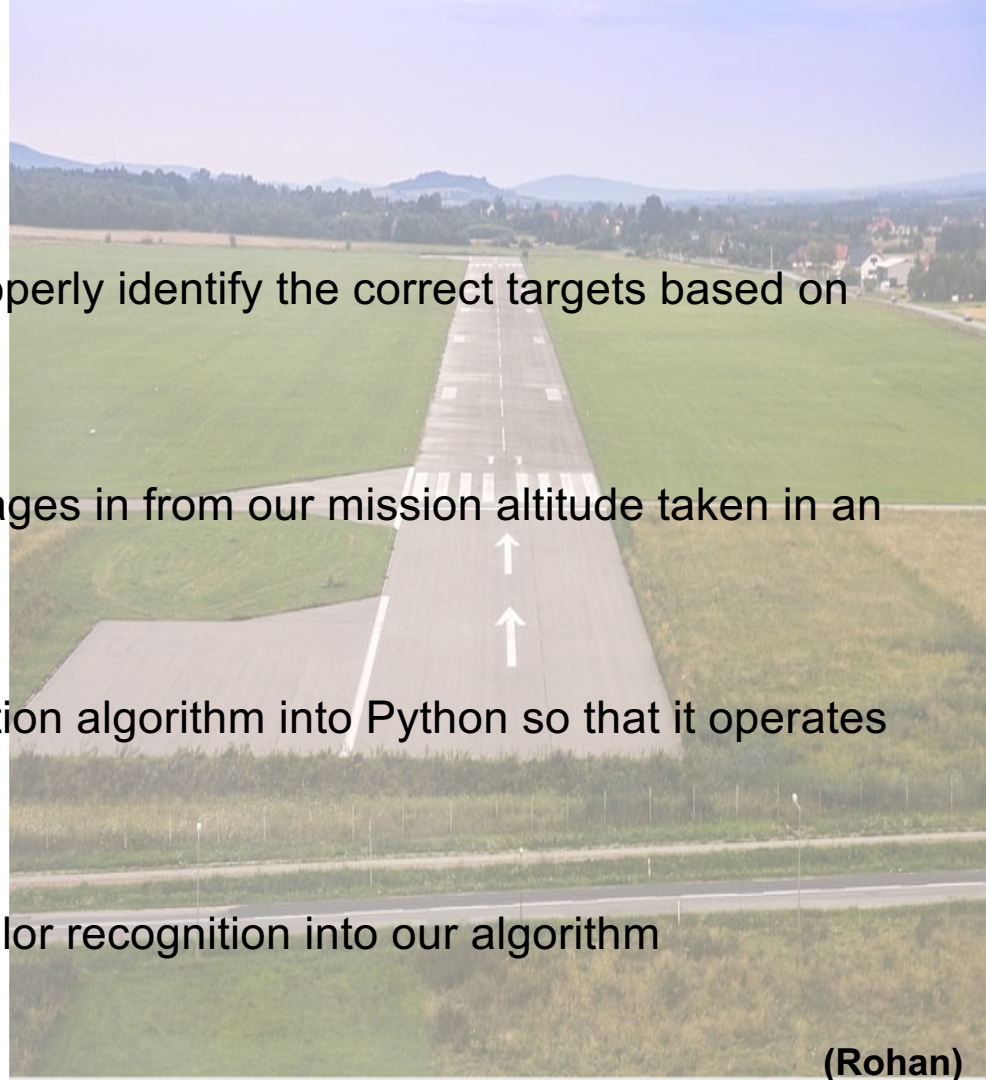
Next Steps

- Troubleshoot our CV algorithm to properly identify the correct targets based on the given training images

- Collect proper training images i.e images in from our mission altitude taken in an open field

- Begin to implement our Map Generation algorithm into Python so that it operates on our co-processor

- Figure out how to implement RGB color recognition into our algorithm



(Rohan)

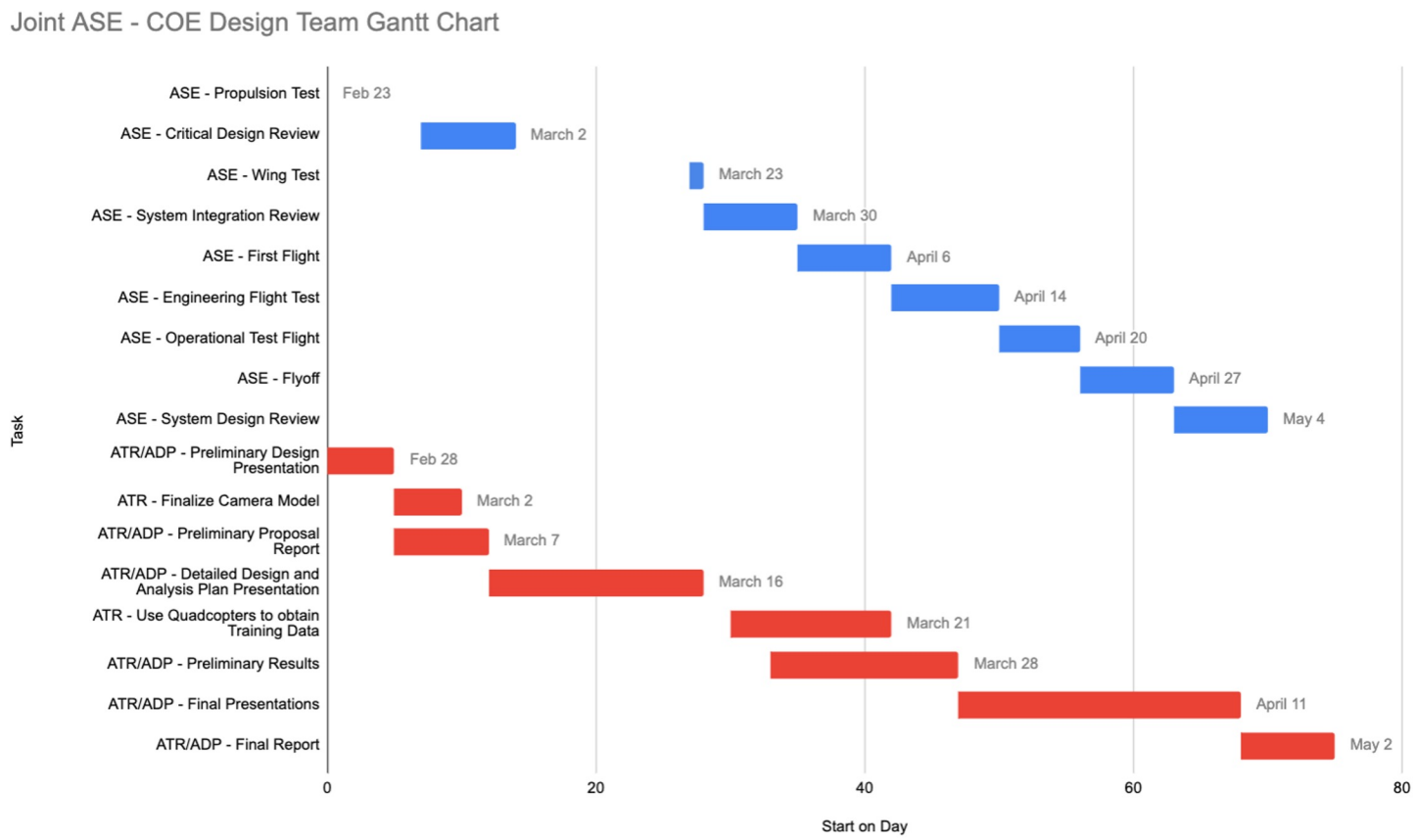
Next Steps: Gathering Images to Test Algorithm

- Creating an accurate OpenCV model requires training data on real imagery

- To acquire those images, we will mount the raspberry pi camera onto a drone and fly it over the training field

(Nicholas)

Updates to Project Scheduling



(Justin)


Q & A