Duncan Mazza
Elias Gabriel

Refresher on our project: https://github.com/sd19spring/CropMeOn/blob/master/PROPOSAL.pdf

Overview

We both have identified areas of skill and knowledge, so we are splitting up the tasks equally to most efficiently complete the project. Duncan has more experience with OpenCV and computer vision, so is taking more of a lead on post-processing of the video and image stills. Elias has more experience with frontend-backend architecture and communication, so he is taking more of a lead on integrating the OpenCV code with a web-facing GUI (a web app). We have both agreed, however, that we will actively collaborate on all aspects of the project, as we believe that it is important for both of us to having a complete picture of how everything works and fits together.

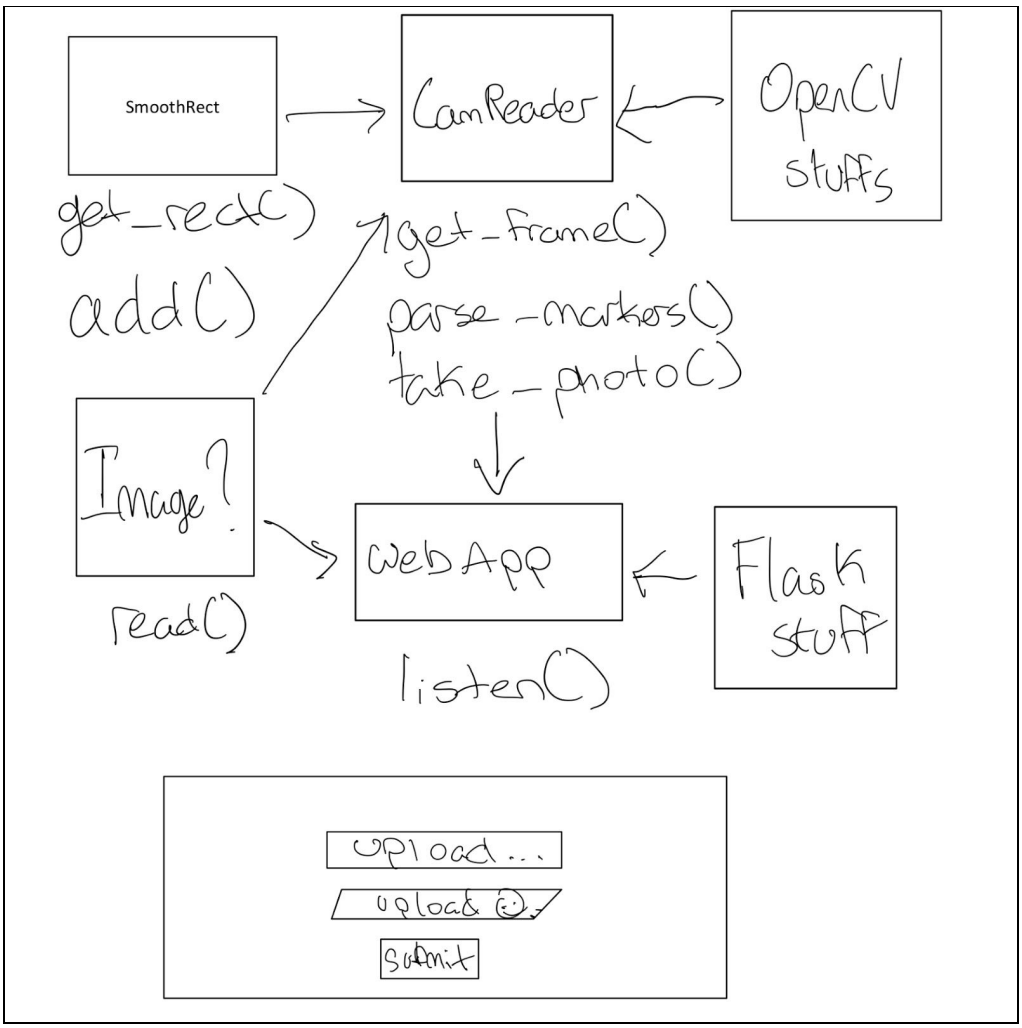
We are off to a good start on our project. Before beginning the project, we completed the OpenCV toolbox and the Flask toolbox to provide us with a solid foundation for how to approach our end product, and how to interact with and use those libraries. Based on our initial UML, we have began implementing classes and tests for hosting a web app and processing live video feeds.

(https://github.com/sd19spring/CropMeOn/blob/master/cam_classes.py)
(https://github.com/sd19spring/CropMeOn/blob/master/web_classes.py)
(https://github.com/sd19spring/CropMeOn/blob/master/app.py)

In terms of our specific goals, we have kept on track with almost all of them. Initially, we wanted to use Pygame to create a UI, but after some reflection we decided it would be both easier and more user-friendly / accessible for it to be a web app. Both of us have looked into web frameworks and OpenCV, and have made progress in learning about their utilities and features. In addition, while Duncan has done the OpenCV toolbox and Elias has done the Flask toolbox, we plan on completing each other's' tool boxes so that we both can gain a more in-depth understanding of the topics we know less about.

Implementation

UML:



Next Steps:

- We have the CamReader class working using OpenCV's blob detection, but we would also want to try tracking ARUCO markers; testing out code for the viability of this will be the next step on the computer vision side. Duncan has spent over an hour trying to get the ARUCO library to work, and is currently communicating with a NINJA to solve this problem. As it stands, the blob detection works, but is non-ideal and will hopefully be replaced by ARUCO markers.
- We have the web app working and running, but it doesn't yet communicate directly with the OpenCV python code. Ideally, we will solidify how the frontend and backend will

communicate, and abstract out common methods so that both parts can develop separately without breaking compatibility.

Further Into The Future:

- We will need to get the web app to talk to the computer vision code - this means initializing it with relevant parameters, displaying the resulting image in the web browser, and controlling the code so that the image capture and saving is a process triggered through the web app.
- The GUI also needs to be fleshed out so that it is visually appealing. It is currently just HTML with very rudimentary styling, but our end product will ideally use CSS to ensure it looks nice and is usable on a range of devices.