CE356 SOFTWARE ENGINEERING

HAND AND FINGER DETECTION WITH

COMPUTER VISION PROJECT

Instructor

Elif Pınar Hacıbeyoğlu

Students

F. Talha Altınel

Onur Tarakçı

TABLE OF CONTENTS

1 INTRODUCTION ……………………………………………………………………... 3
2 AIM AND IMPORTANCE ………………………………………………………….. 4
3 REQUIREMENTS ………………………………………………………………….... 5
4 DESIGN ………………………………………………………………….................. 6
5 IMPLEMENTATION …………………………………………………….................... 7
6 TESTING ……………………………………………….. 8
7 CONCLUSIONS ……………………………………………………………………… 9
8 REFERENCES ………………………………………………………………………... 10

1 INTRODUCTION

In this project, a python based software program is designed and implemented according to waterfall software methodology. Timeline of this project was from start of March 2020 to end of April 2020. We have utilized only 2 python libraries in this project such as numpy for scientific calculation and openCV for image processing and computer vision. We also have made use of “pipenv” which is a library dependency management tool for all operating systems so everyone can make use of the final program. Everthing in this project used only detection calculation methods without any machine learning recognition methods. This project is designed for a purpose to achieve simplicity for image processing and computer vision newcomer students and be able to make them understand fundamentals of image processing and computer vision. In the end, we produced great results for detecting the hand and the fingers with only a few scientific math calculations without any kind of recognition algorithms which rely on lots of classified image datas.

2 AIM AND IMPORTANCE

This project's essential aim is to create a computer vision software which serves people's adaptive needs without the need of physical or audial interaction with a computer machine in 2 months. The project also aims towards a future where everything is contactless and easy to understand to do with hand gestures such as showing numbers. The case for this might be a subway ticket machine where physical interaction with the buttons may spread a disease to others with the button or the button may get malfunctioned over time by getting pressed a lot. Subway ticket machines also can't make efficient use of audial interaction with the people due to lots of people marching and making noises. For those specified reasons, we see that visual interaction of a computer vision may benefit us a lot in the long term if every aspects of normal ticket machine requirements are satisfied such as specifiying the ticket number count to buy.

3 REQUIREMENTS

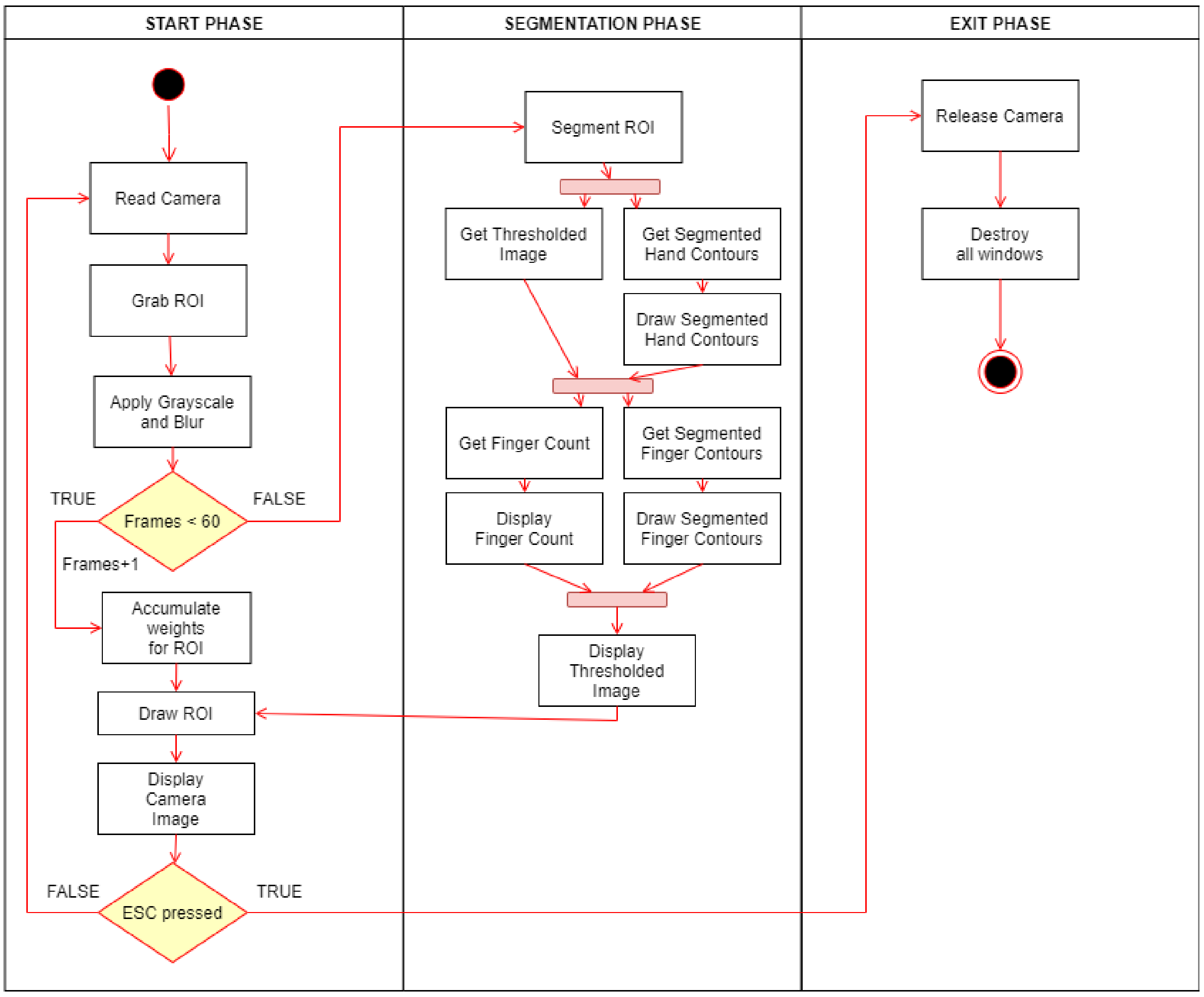
This project's requirements are listed below according to people's needs. For the priority of tasks we made 4 categories such as not essential, essential, important and very important. After that, we wrote down the people's needs as tasks.

Tasks	●	▲	■	◆
Segment hand's fingers and outer convex shape				X
Count the segment of fingers				X
Perform hand's estimated size calculation			X	
Detect the palm of the hand in a specific range of brightness.		X		
Eliminate the background noise properly in extreme light conditions	X			

●	Not Essential
▲	Essential
■	Important
◆	Very Important

4 DESIGN

This project is designed as it is shown below. We utilized activity diagram for better understanding of different phases and processes. Phases are divided into 3 categories such as start phase, segmentation phase and exit phase. Main logic of the the program was encapsulated in segmentation phase.



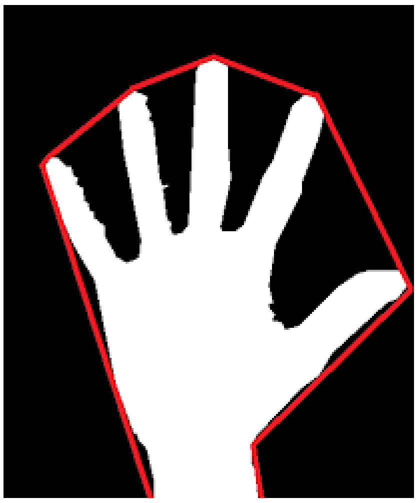
5 IMPLEMENTATION

From implementation side, we used pipenv python library's virtual environments to make sure all of the team members have the similar openCV, numpy and python versions. Because different versions had different APIs and dependencies. We also used source control tool such as git to make sure small incremental changes are done without breaking things.

Initially we had 3 distinct functions that we implemented, first one("accumulate func.") was for accumulation of initial background, second one("segment func.") was for segmentation of the hand and the third one("count_fingers func.") for the segmentation of fingers and counting the count of it.

Due to the fact that our computations take lots of spaces in the third one("count_fingers func."), we decided to add new utility function for calculating eucledian distances and named it as "eucledian_distances".

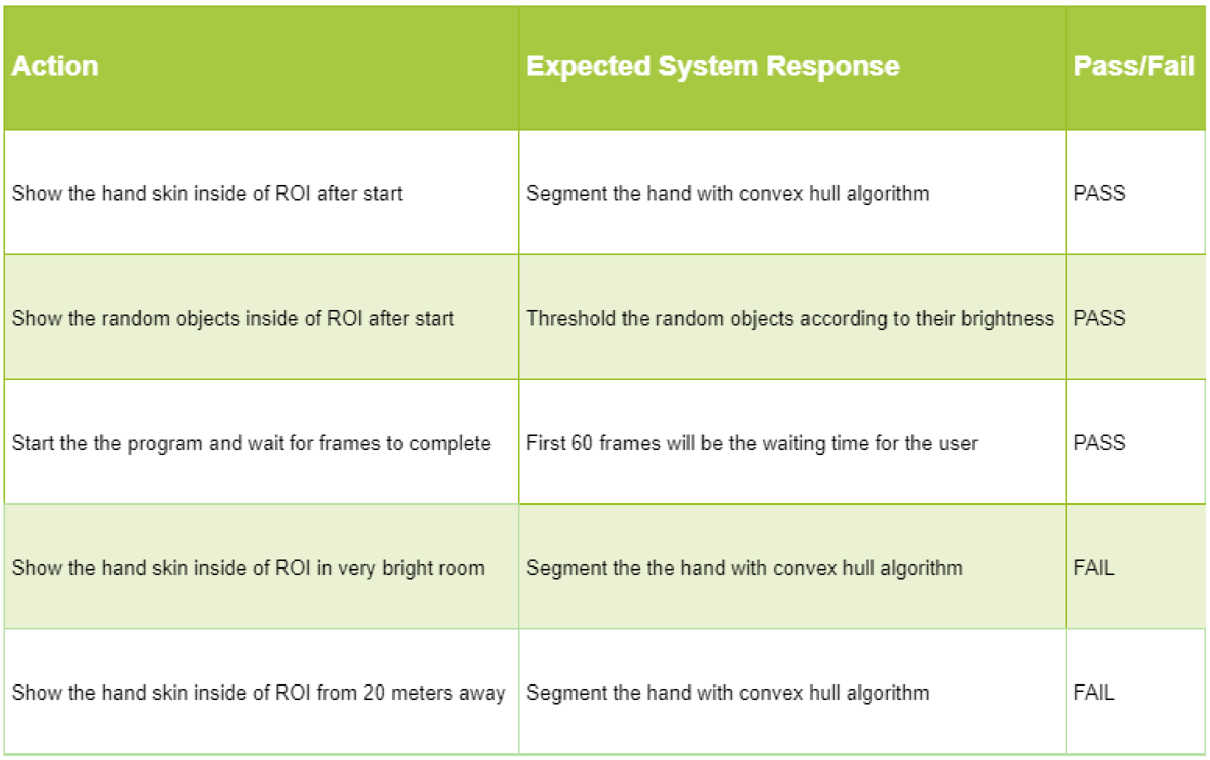
After all of our methods are implemented, we added main program's thread according to design schema. After the design schema is added, we did some small debugging to figure out if the program is working correctly in areas such as thresholding the image, picking up the contours, applying convex hull algorithm to the image. After debugging is finished, we added comments to each method's brief description as well as general steps that are done by the third party libraries.



Convex Hull Algorithm on Hand Shape

6 TESTING

After applying segmentation test, thresholding test, video frame test, brightness test and distance test. We constructed a table which shows expected system response and pass/failure of the tests. We listed the tests according to the importance level in table. Segmentation test, thresholding test were very important for the system to work properly. Video frame test was important as well for to not analyze frames so fast or so slow. Brightness and distance tests were the least important ones because our design from the beginning was designed for typical subyway ticket machine places. We also observed the extreme points from the least important tests.



Action	Expected System Response	Pass/Fail
Show the hand skin inside of ROI after start	Segment the hand with convex hull algorithm	PASS
Show the random objects inside of ROI after start	Threshold the random objects according to their brightness	PASS
Start the the program and wait for frames to complete	First 60 frames will be the waiting time for the user	PASS
Show the hand skin inside of ROI in very bright room	Segment the the hand with convex hull algorithm	FAIL
Show the hand skin inside of ROI from 20 meters away	Segment the hand with convex hull algorithm	FAIL

6 CONCLUSION

In conclusion, we learnt about how to design/implement a computer vision system as well as how hard it is to maintain it. We faced small challenges and fixed the issues with small incremental steps. We tried to document every step of the progress while improving. Test has showed us our system is not that perfect. By design, we are still getting great results with all of the fingers except the first thumb which sometimes needs a bit more flex to make it count. In future, there might be an implementation of hand recognition techniques with machine learning algorithms so that we don't need to calculate finger segments from the hand segments which is a trouble work and accuracy can be better with that way. We are glad with system results except the brightness and distance issues. There is always a room for improvements but whole design might be reconstructed from scratch in case of doing things in different ML ways.

Project presentation link can be found here. Github link also can be found here.

7 REFERENCES

The SciPy Community 2008, *NumPy Reference – Numpy v1.17*, viewed 1 May 2020, < https://docs.scipy.org/doc/numpy/reference/>

Open Source Computer Vision Community 2000, *OpenCV: OpenCV Modules v3.4.8*, viewed 1 May 2020, < https://docs.opencv.org/3.4.8/index.html>

Parker J. R. 2011, *Algorithms for Image Processing and Computer Vision The Second Edition*, Wiley Publishing, Indianapolis