Facial Landmark Visualizer

Lada, Legolas Tyrael B.

in fulfillment of the requirements of

CMSC 191 – COMPUTER VISION IN PYTHON

Submitted to

Assoc. Prof. Armacheska R. Mesa-Satina, Ph.D.

May 2025

Abstract

Arguably one of the harder parts of image processing and computer vision is visualizing the preprocessing steps required to get to workable data, that is, data that is easy to analyze and perform logic on. Moreover, visualizing these preprocessing steps can be cumbersome, especially for steps that are more complex than applying simple filters and transformations over the image. This project acts as a tool to be able to visualize steps in isolation or in combination with the other preprocessing steps required in obtaining facial landmarks via *Haar cascades*, a simplified approach to facial detection.

Objectives

This project aims to provide an easier way to visualize the preprocessing steps involved in obtaining facial recognition data from a Haar cascade produced by `dlib`. Specifically,

1. Allows an intuitive interface for activating showing specific preprocessing steps;
2. allows for overlap of displayed preprocessing steps;
3. apply the preprocessing to a live webcam feed;

Project Features

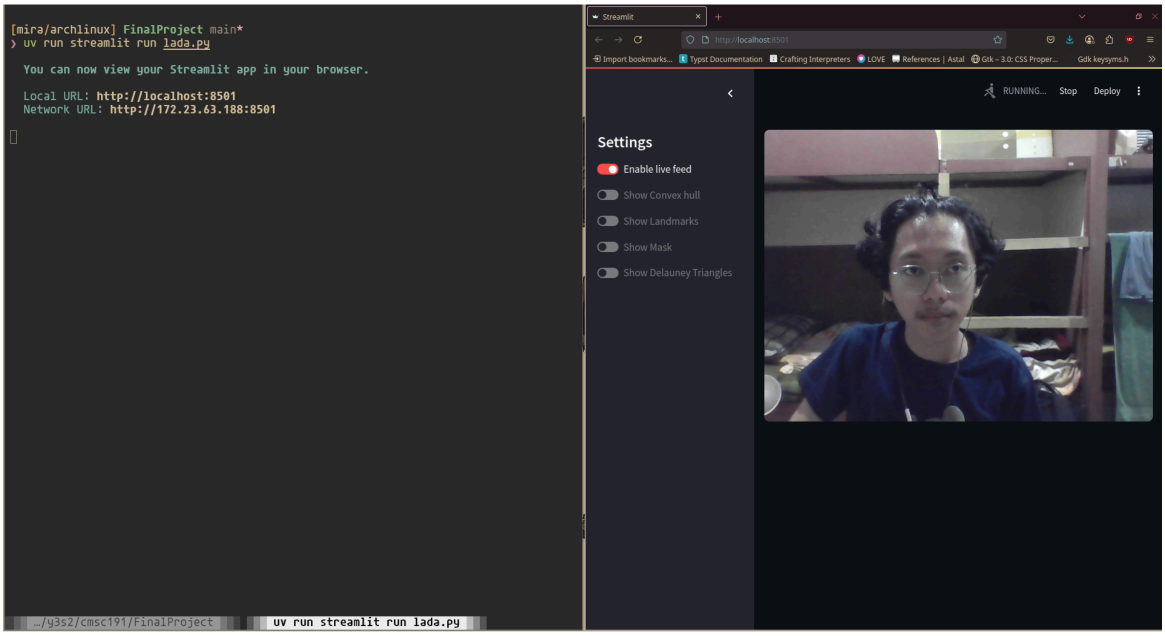
The project allows for the toggling of four preprocessing steps in particular. These are: locating and predicting *68* (obtained using `shape_predictor_68_face_landmarks.dat` and `dlib`) facial landmark points, displaying the convex hull formed by the landmark points, displaying the mask formed by the convex hull, and the *Delauney triangles* formed by the landmark points.

These are all laid out using *Streamlit.* The aforementioned settings are then applied to the live webcam feed.

Techniques Used

This project utilizes a pre-trained model with `dlib`, as done in one of the laboratory works. It also uses various *OpenCV* functions to perform preprocessing and filters on the image, besides loading the image from the webcam. The image was turned grayscale, blurred, and then morphed using the *open* morphological transform, to improve detection of the face.

Screenshots and Workflow



```
[mira/archlinux] FinalProject main*
❯ uv run streamlit run lada.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.23.63.188:8501

```

```
…/y3s2/cmsc191/FinalProject    uv run streamlit run lada.py
```

Settings

● Enable live feed

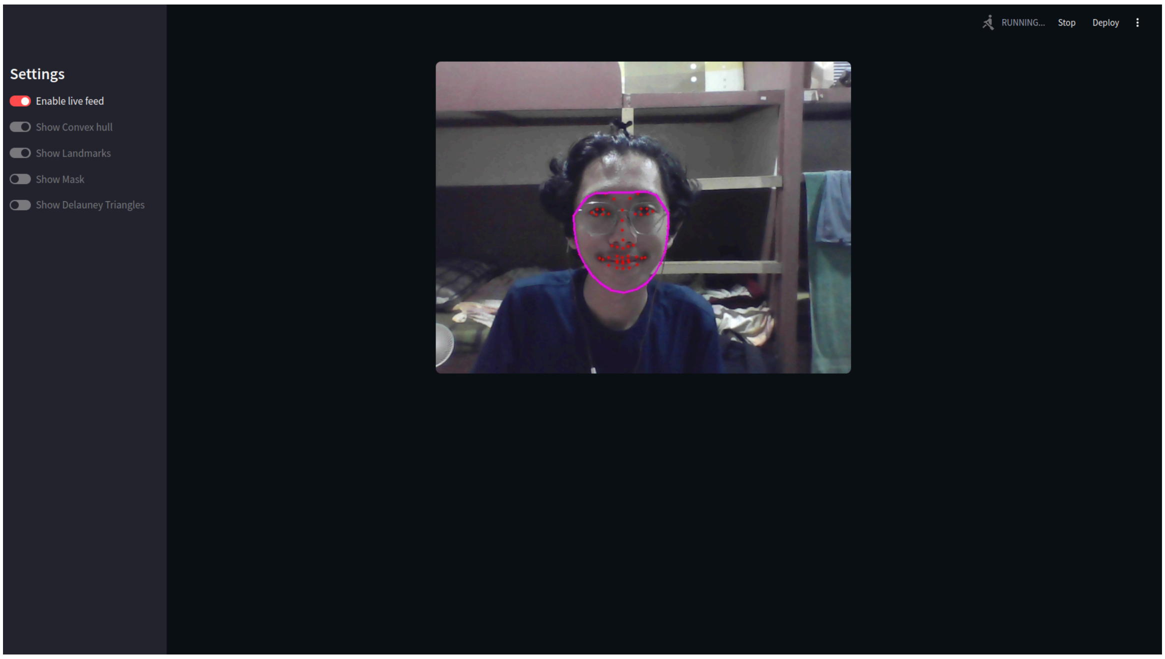
○ Show Convex hull

○ Show Landmarks

○ Show Mask

○ Show Delauney Triangles

Figure 1: A live feed without any filters visualized.



Settings

Enable live feed

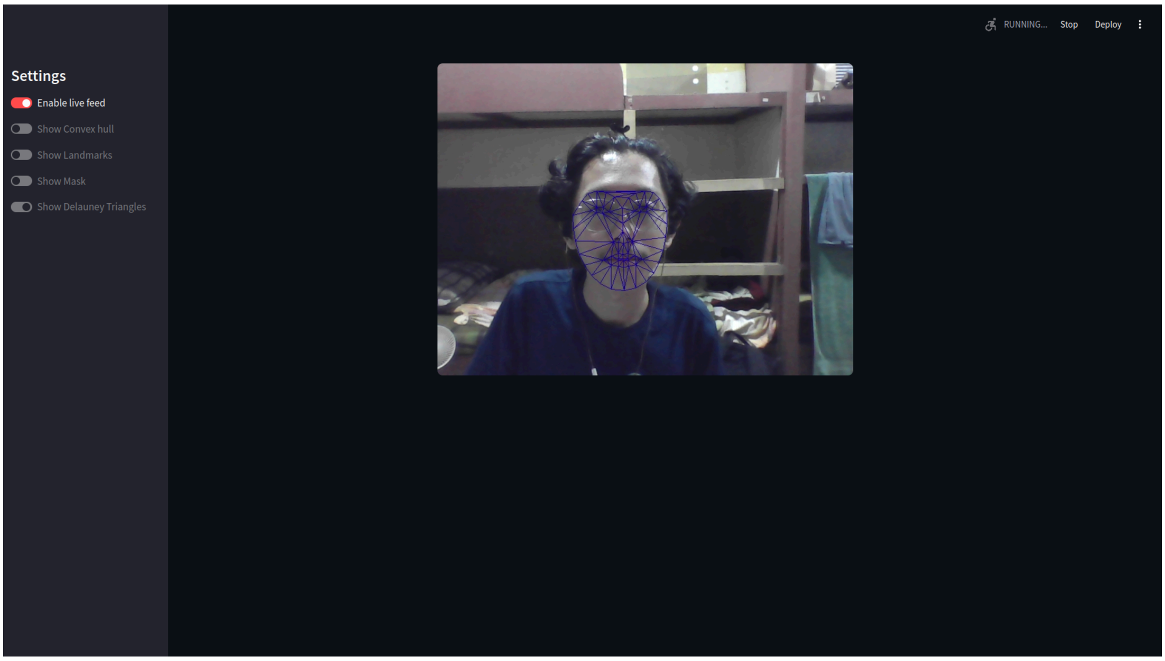
Show Convex hull

Show Landmarks

Show Mask

Show Delauney Triangles

Figure 2: A live feed visualizing the landmark points and convex hull.



Settings

Enable live feed

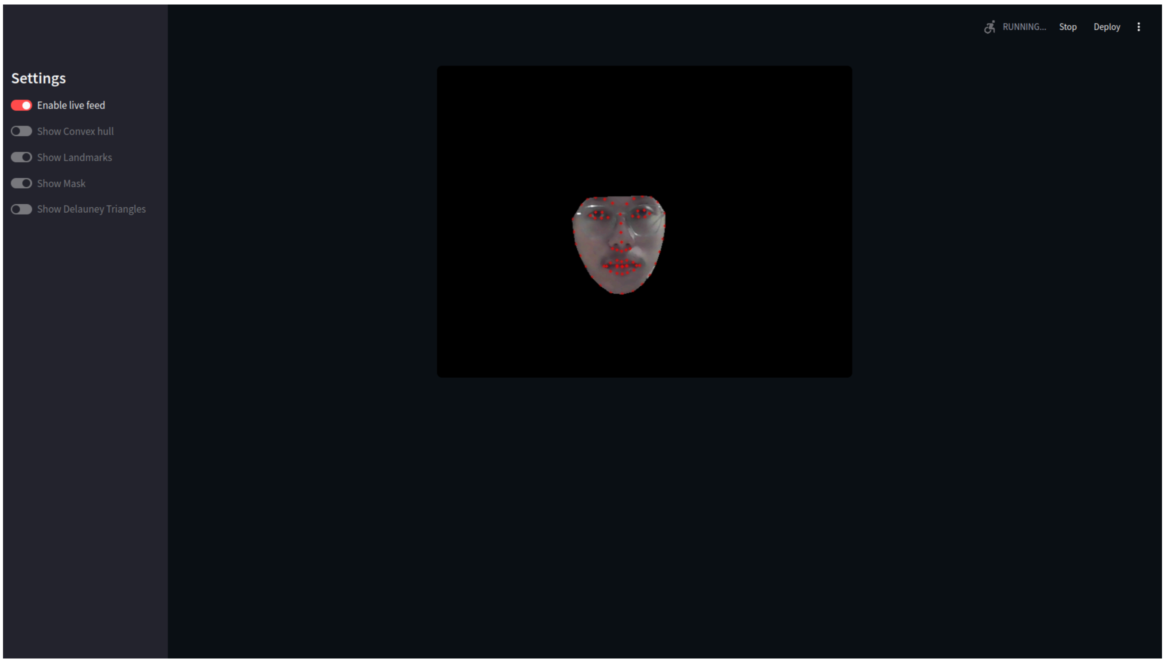
Show Convex hull

Show Landmarks

Show Mask

Show Delauney Triangles

Figure 3: A live feed visualizing the Delauney triangles formed by the landmark points.



Settings

- ● Enable live feed
- ○ Show Convex hull
- ○ Show Landmarks
- ○ Show Mask
- ○ Show Delauney Triangles

Figure 4: A live feed visualizing the image mask formed by the convex hull along with its landmark points.

Challenges and Limitations

The main challenge that plagued me for the development process of this project was the ideation process. I must have spent about 3 days trying to think of a project idea. Additionally, it is also my first time using Streamlit, so I encountered issues regarding its state handling, as it refreshes the page on input.

I also had to relearn a bit of the preprocessing done in using a Haar cascade model, particularly all of the steps being visualized in the project.

Lastly, I tried to let the settings be toggleable without closing the live feed, but I couldn’t figure it out, because of the inexperience with Streamlit.

Future Enhancements

I believe the user experience can be improved by being able to toggle the filters while the live feed is active. And being able to customize the appearance of the filters would also be good.

And of course, adding more steps to be visualized is also a great way of improving upon this project.

Reflection

This project was very helpful in helping cement my learnings in the course, as it had me culminate them into a project that I chose. There was definitely a struggle in the ideation but only because of the fact that computer vision lends itself to an overwhelming amount of possible projects, which was very fun to think about, in hindsight. I wish I could have been able to do a whole lot more, but there was a part of me that just felt like over-investing into my first proper computer vision project would give me very diminishing returns. Although I am not fully satisfied with it, I still am very much looking forward to applying more of what I have learned in the future, and I think this project has been a very good springboard, in that regard.

References

The code was written entirely by me, besides the dataset, which was obtained from Laboratory Work #7 of this class.