Face Detection and Tracking

Alex Wen, Raj Kishore

Introduction

In this project, we will be creating an app that will be able to detect facial profiles and track it through location and motion detection. This is intended for the problem of recognizing facial profiles through software implementation and tracking it through certain frames. This is such that it attempts to begin facial detection in a broad environment and the software will recognize as such; if time-permitting, saving and recording the feed as well as recognition could be factored into the equation as well.

Background

From Eigenfaces for Recognition, we are trying to detect whether or not certain pictures are faces. At a high level, we do this by seeing how close the picture's features are to those of an average face.

In our assigned lab, we tried to determine if a given picture was a face, as well as try to recognize who the face was. With our program, we were able to detect whether or not an image was a face perfectly, however we were not able to recognize which face it was very well.

Now, instead of taking a picture of a face and determining if it is a face or not, we will be trying to detect all faces as we are recording. We will be doing this by searching for an image of a face in the frame of the video.

[Citations]

Matthew Turk and Alex Pentland, **Eigenfaces for Recognition.** Journal of Cognitive Neuroscience 1991 3:1, 71-86

Deliverables

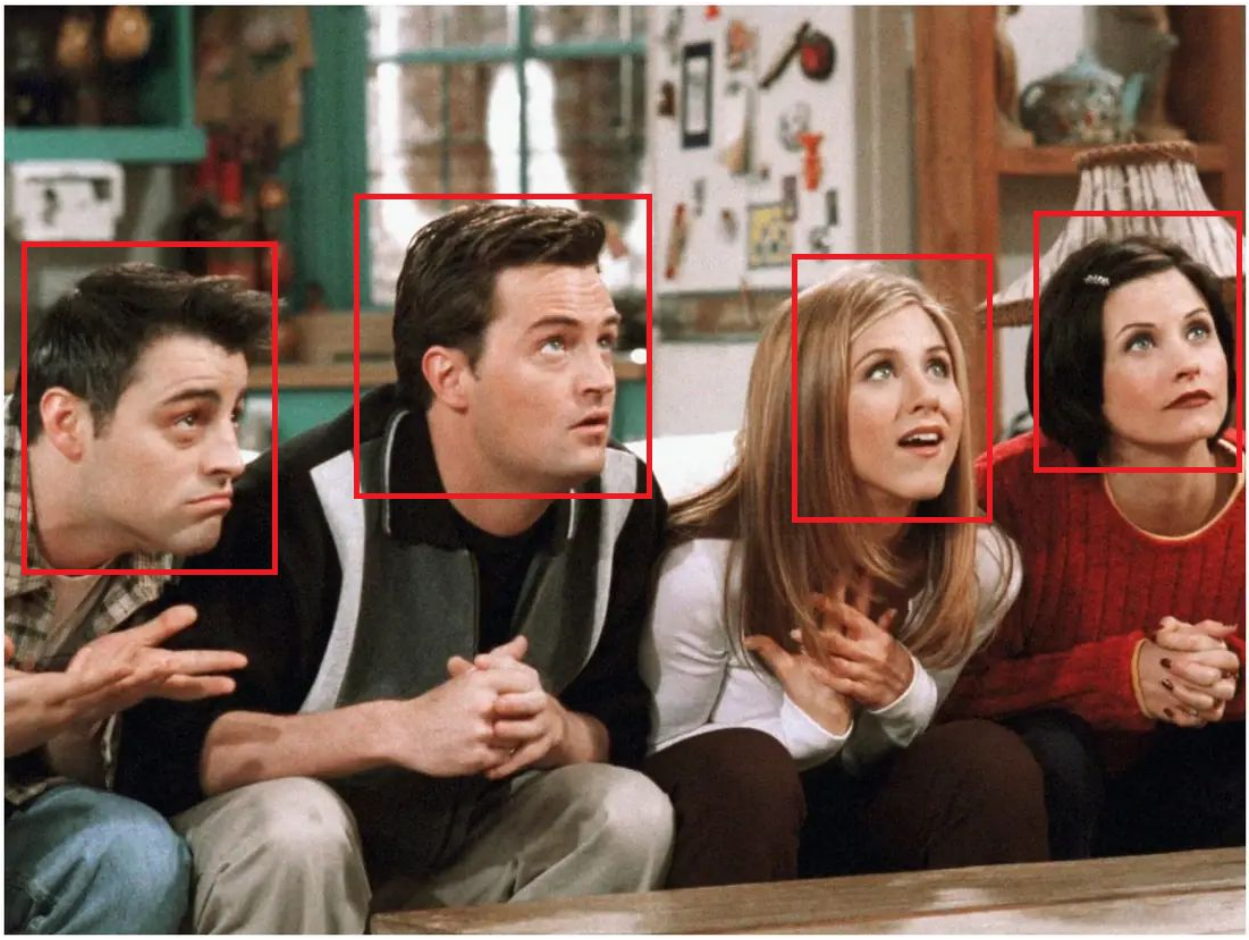
The final product would essentially be an app where the input is given at an initial frame with faces that are 'located' by the algorithm. The output will then be such that for every frame that has faces presented there will be a bounding box encompassing said face, at the minimum.

Some time-permitting features that can be included in the design:

Priority 1: We could also consider saving the feed - that is, recording a video segment or saving an output image that will be saved to local files.

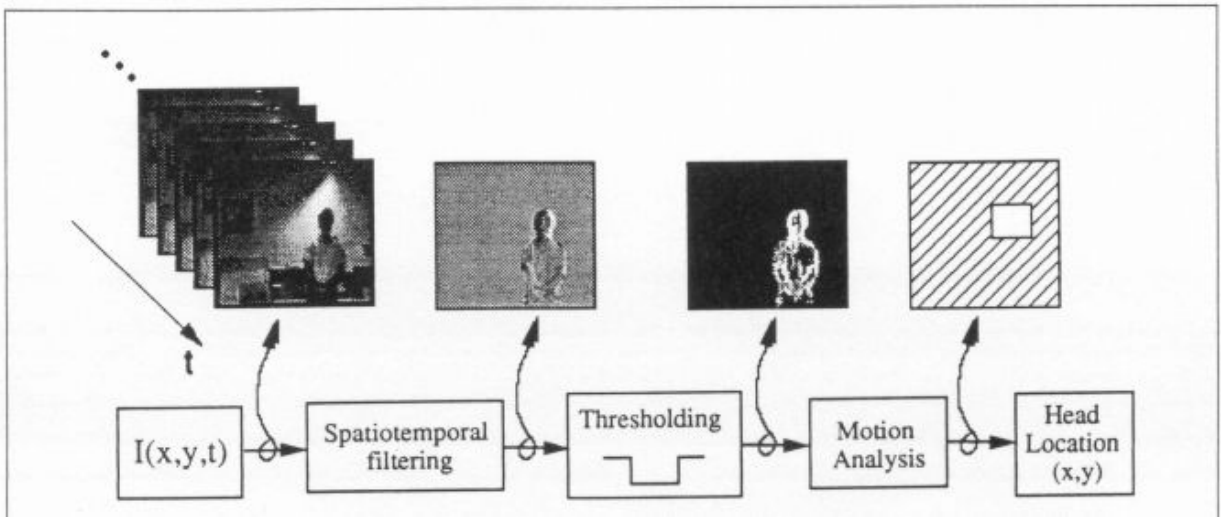
Priority 2: At first we were planning to detect, recognize and track in our final project, but realized it would be too much, but if we have time, we'll try to implement recognition as well.

For the app, the base would merely consist of the feed provided by the camera as well as a 'Start Detection' button. If time-permitting features, a 'Save Video' and/or 'Save Image' button can be included for recording video or saving images. Finally, if doing recognition, a 'Toggle Recognition' button will be included as well



For example, if this image was a frame of the video, this would be the desired base output at that instance.

Algorithm Summary



(from Eigenfaces for Recognition - this will feed from image to video detection processing)

We will be searching through the entire image to see if any chunks of it correspond to faces. We will need to check many parts of the image to find a face, and if we do we will draw a bounding box around it.
We will have some preprocessed data from python with weights of eigenfaces. Using these weights, we can determine if a certain part of the image is a face. We will be doing this every few frames (every ~30 frames or every second), so we can see if any new faces have appeared. If a face is detected, we will track it - using either KCF or a different tracking algorithm.

Milestones

Milestone 1: We want to try to set up the skeleton format of our app in android. We also want to see if we can detect face(s) given an image containing a face in python (as opposed to determining if an image is a face). This is such that for any given single frame, the algorithm will be able to correctly identify the faces present.

Milestone 2: We want to detect face(s) given an image containing a face in our android app. This is a transition from the python part, since instead of being provided an image that can detect a face, we can try and utilize the Android app camera for image taking and detect the faces as such.

Final Demo: We want to finalize the app such that it can detect faces and track them as we move the camera around. This will concatenate all frames of single frames together, where the bounding boxes will follow the faces as given.

Additional remarks: For time-permitting features, we can try and allocate more pictures in the training data for actual recognition. As such, if we do this, inputs would include multiple profiles of target faces for ‘training’ and to transmit that to the output such that all ‘recognized’ faces will be ‘highlighted’ in the output tracking. We can also include a saved file that would include the feeds allocated through image or video recording indicated by the buttons. This would ideally be done between Milestone 2 and Final Demo.

Testing and Validation Plan

We will test this by recording ourselves (maybe family and friends as well), showing that we can detect faces. We will also record with objects in the background to make sure we don’t falsely identify other things as faces. At the base, inputs would be a single-frame image of a face or faces captured by the camera, while the output would be a video with bounding boxes capturing the faces in the feed provided by the tablet’s camera.
If we consider saving/recording feed, a video and/or image file will be added to the outputs consisting of a single-frame feed (image) or multi-frame feed (video) of the feed captured by the camera (for the video, between the start and stop signal).
If we consider recognition, target faces would be indicated on the feed with their respective bounding boxes.