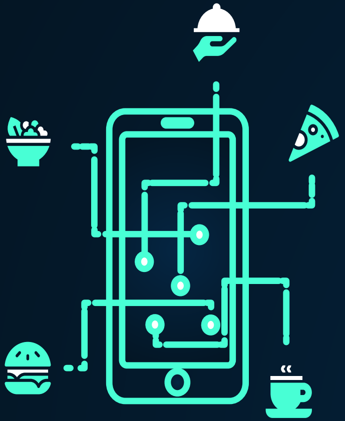
Kombuder Bishon

Team Members:
1. Sudhansh Yelishetty
2. Giri Prasath
3. Ashwin Gopinath
4. S.Shreevignesh

Project topic

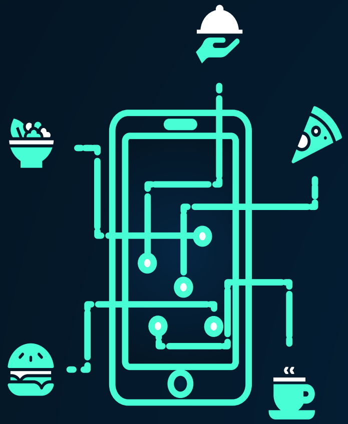
Facial recognition/ verification: Automatic Naming of Characters in TV Video

Reference paper: "Hello! My name is... Buffy"



Objective

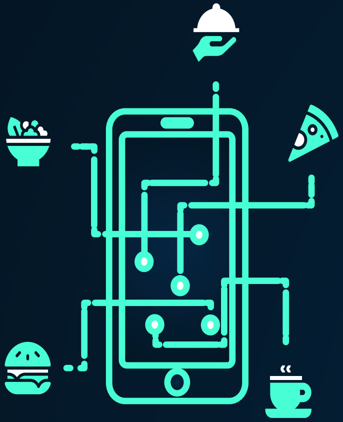
Automatic naming of characters in TV Video, and increasing its accuracy by combining multiple sources of information, both visual (using concepts of facial detection) and textual (Subtitles).



Method

The method of naming the characters include:

1. automatic generation of time stamped annotation by aligning subtitles and transcripts.
2. strengthening the supervisory information by identifying when characters are speaking.
3. using complementary cues of face matching (and clothing matching) to propose common annotations for face tracks



Goals

We are planning as mentioned below:

- **First week**: Planning, designing and solidifying the pipeline of implementation.
- **Second and Third week**: Detecting all faces on the screen.
- **Fourth week**: Implementing script processing and detecting characters in scene from script.
- **Fifth week**: Detecting mouth movements to find who is speaking in the scene and match the characters name from the processed script.

OUR GOAL

START

Week 1
Planning and design

Week 2 and 3
Detecting all faces on screen

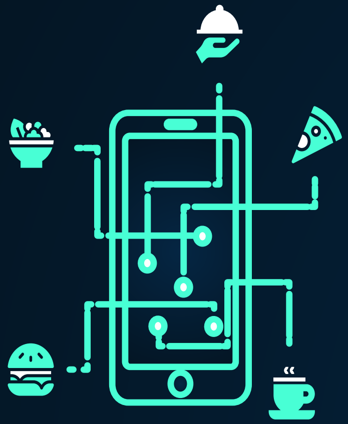
Week 4
Script Processing, detecting characters from scene

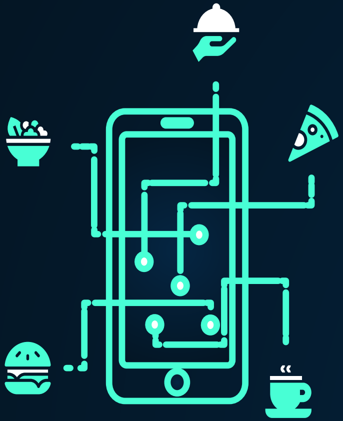
Week 5
Detecting which character is speaking and assign name from script

MID SUBMISSION

Mid submission deliverable

1. Detecting all faces on screen
2. Detect mouth movements in these characters
3. Use this to find which character is speaking

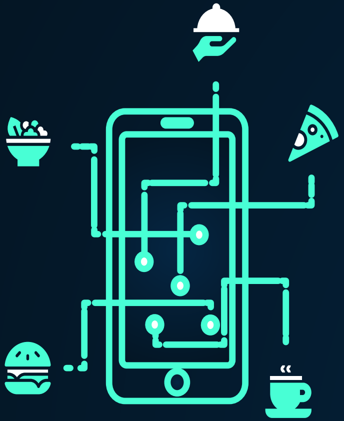




Detecting all faces on screen

Experiments:

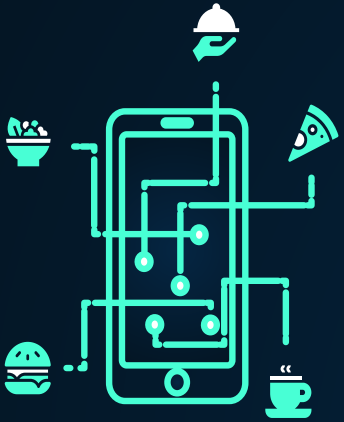
1. First we tried using a Haar Cascade Classifier specifically for full frontal faces, as mentioned in the paper. But this method was slow and unreliable as it gave many False-Positives. In addition to this these, the Haar cascade classifier did not give any distinction between the facial features (eg. mouth movements). Thus, to make the process faster and more reliable, we shifted to a Deep Neural Network-based architecture.



Detecting all faces on screen

Experiments:

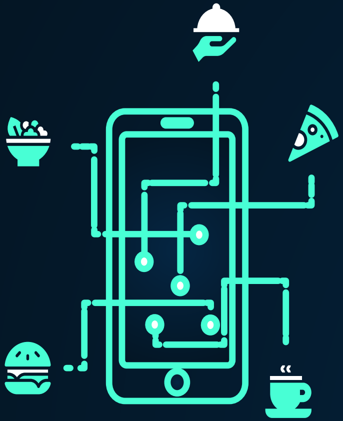
2. The deep neural network (dnn) method was the fastest method of all the methods we experimented with. But the results were unreliable, the output had lots of false positives and took many many iterations to learn facial features to detect. It could not predict and detect distinct facial features, similar to the Haar implementation. Thus, we had to tradeoff speed to accuracy and detecting facial features and choose an implementation which uses dlib data and functions.



Detecting all faces on screen

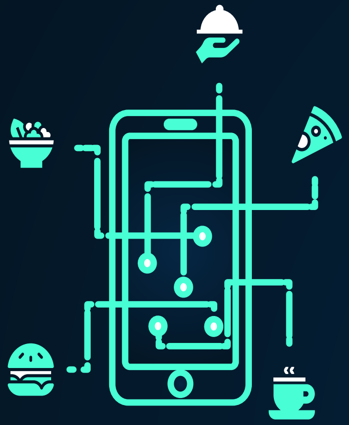
Experiments:

3. At last, we decided to work on the dlib implementation, which even though was slow in comparison to the other methods, had a high accuracy in detecting faces. In addition dlib had much more robust functions which worked on detecting specific keypoints of the faces, which allowed us to detect and localize different facial features and detect independent movement of these facial features (eg. lip movement, explained in the following slides). Using shape priors trained on the 300W dataset, we were able to leverage HOG features in our implementation when it came to face and feature detection and localization.



Detecting Lip movements

1. Dlib allowed us to localise facial features with keypoints. Using those keypoints, we used the keypoints around the mouth to detect lip movement in order to identify the speaker. However, this started giving false positives as the keypoints moved even when the head was moving or when the camera moves.

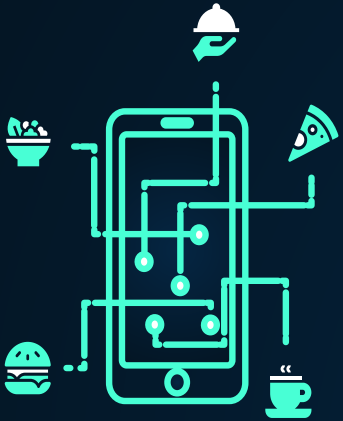


Detecting Lip movements

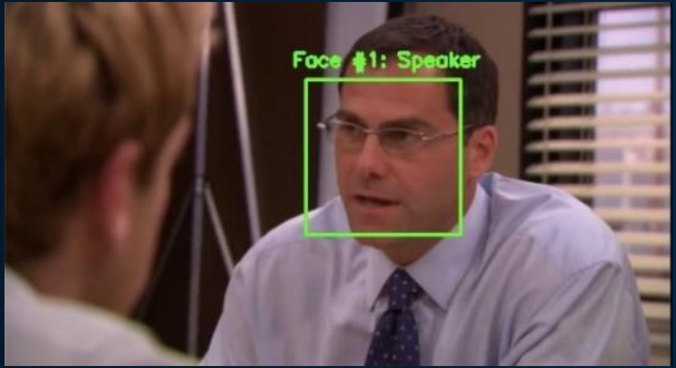
2. In order to tackle this issue, we cropped the mouth region and used it as our region of interest. Now we compute the changes in keypoints position within the ROI to detect lip movement. This solved the previous problem

Detecting Lip movements

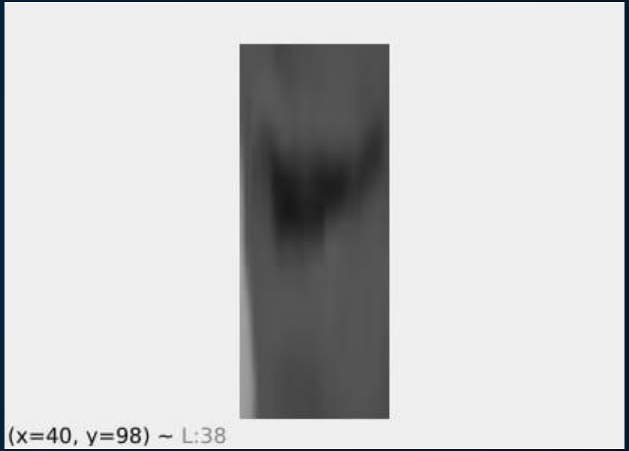
3. The metric used for identifying lip movements is the Euclidean distance of the key points. As of now, the decision rule we use to identify the speaker is to find the lip on the current frame with the highest movement. We are exploring alternate ways of doing this in the form of thresholding etc.



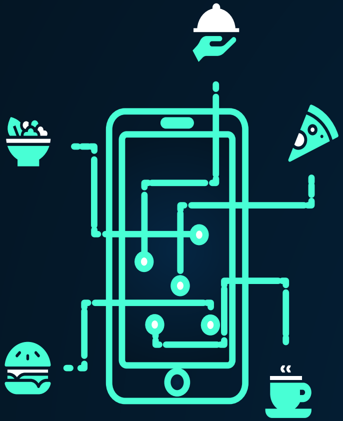
Detecting Lip movements



Face of the speaker detected

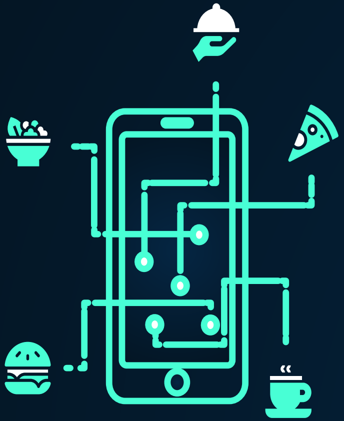


Cropped mouth region



Further improvements to be implemented

1. **THRESHOLDING**:
   As of now the speaker of a scene/frame is detected as the person with the most mouth movement in that frame, this works on the assumption that only one person is speaking per frame and that every frame has a speaker whenever faces are involved, but this might not be the case. To solve this issue we plan on creating a threshold specific to each face which is learnt by the system and a person is detected to be speaking only if he/she crosses the defined threshold in terms of his/her mouth movement.



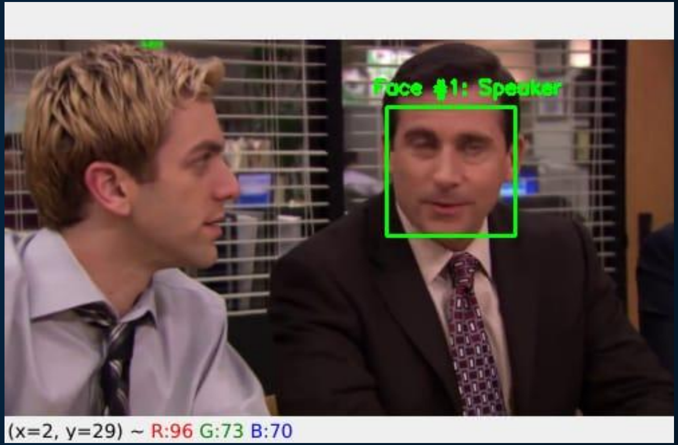
Further improvements to be implemented

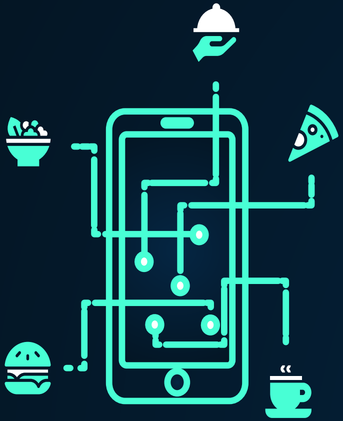
2. **TRACKER:**
   The present algo only detects full frontal faces and leaves out the ones which are even slightly turned to a side, this leads to many false negatives and might not detect the correct speaker. To fix this issue we are planning on implementing a tracker which tracks a face once detected and can determine whether he/she is speaking even when the face is turned a bit.

Further improvements to be implemented

2. **TRACKER:**

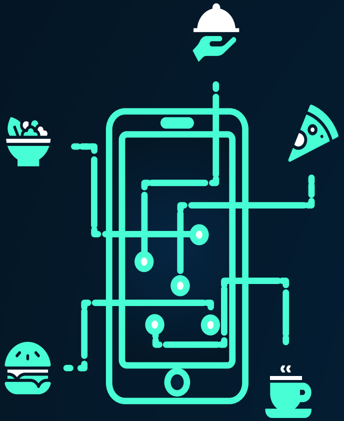
   Here Ryan's face is not detected because his face is turned sideways. This issue could be fixed using the tracker.





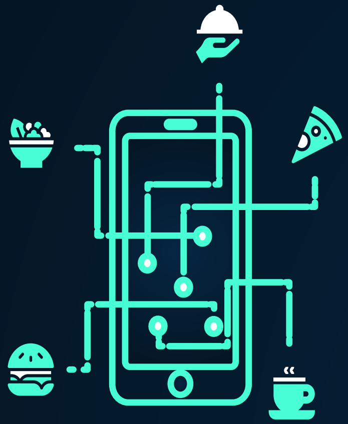
Script Processing

1. We make use of transcript and subtitle files for script processing. The transcript gives information as to who speaks what. The script gives us information as to when each dialogue is spoken.
   We use the transcript to name the speaker on screen, and use the subtitles to match the appropriate dialogue to the speaker.



Script Processing

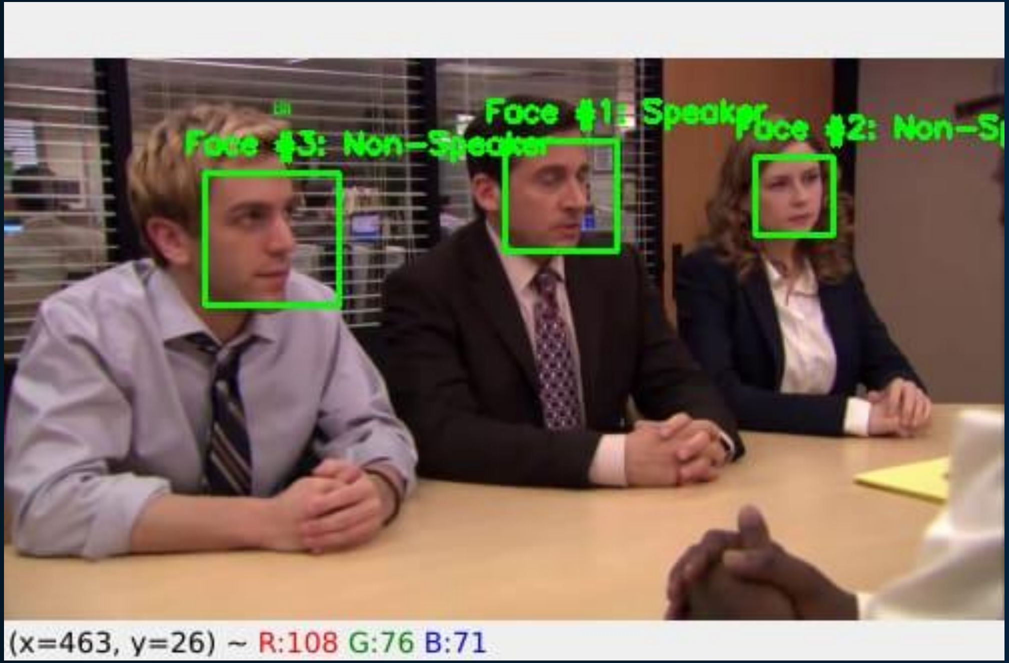
2. We write a function to convert subtitle (srt) and transcript (txt) files into a data type, which is used to depict when a person is talking, and using our mouth detection algorithm, we can name the speaker accordingly.



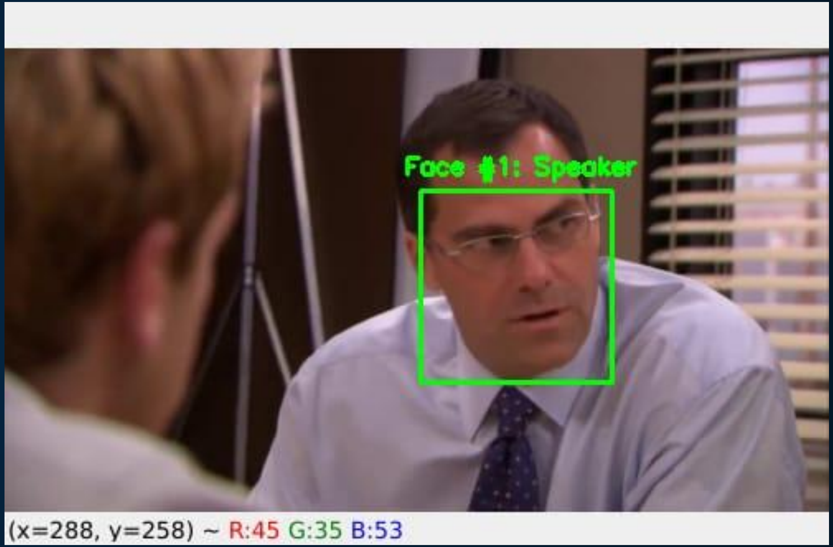
Script Processing

3. After processing, the name of the speaker will be displayed along with a bounding box around his/her face.

Working Examples

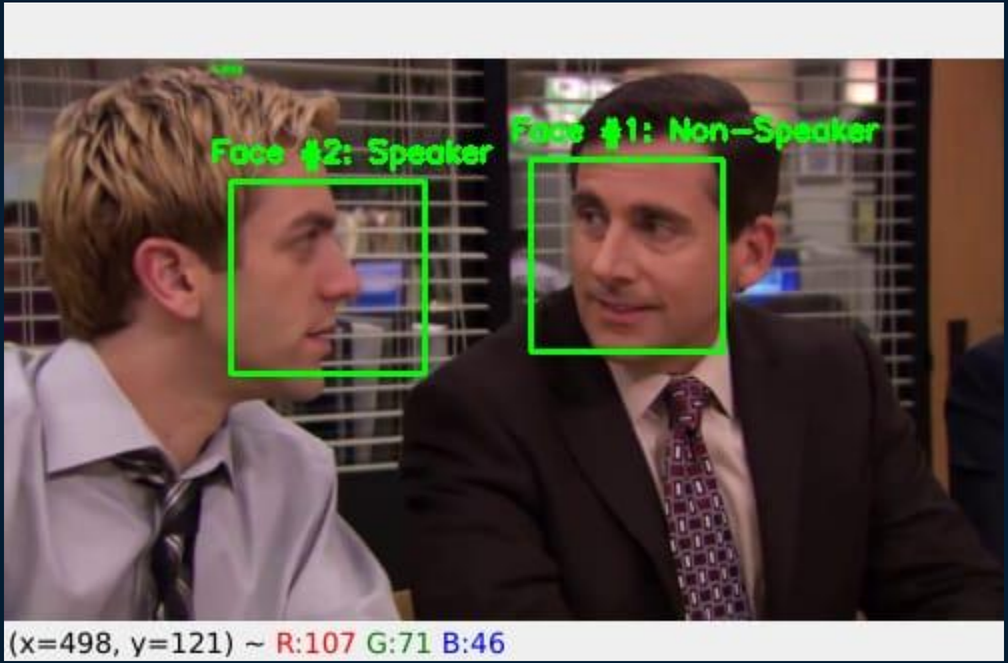


(x=463, y=26) ~ R:108 G:76 B:71



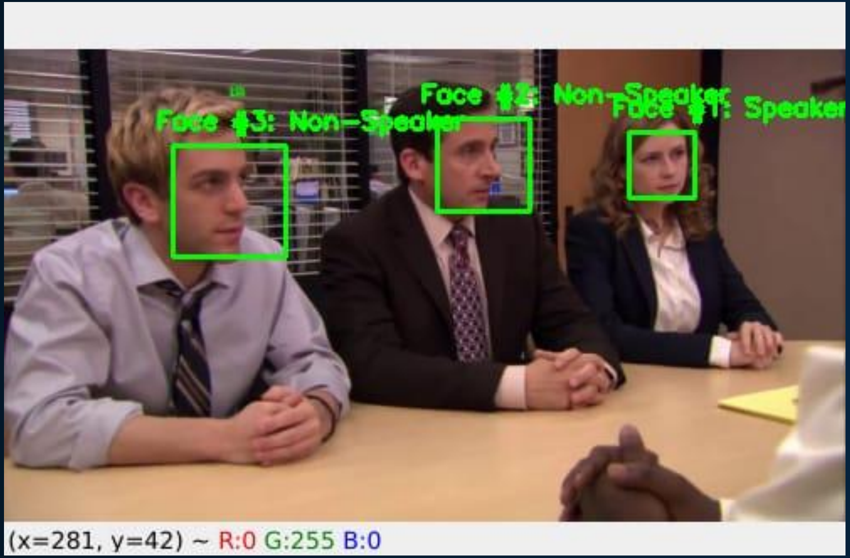
(x=288, y=258) ~ R:45 G:35 B:53

Working Examples

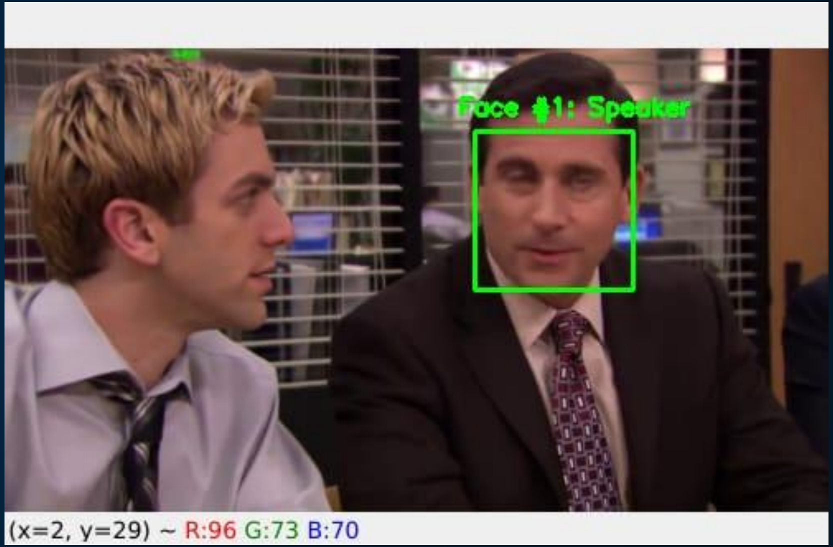


(x=498, y=121) ~ R:107 G:71 B:46

Exceptions



False positive on speaker identification



Ryan's face not detected because it is turned sideways