Identifying Speech Bubbles:

creating automatic speech bubbles for characters on video

HAYUN CHONG

Background

With an increasing use of the internet, there is a proliferation of videos and multimedia content. In order to try and organize these, one important task is to transcribe the events and speech that occur in the video. One way of doing this is to identify the speakers and what they say in the video.

Currently, there are many programs that try to automate the subtitling process when creating videos. Youtube has an auto-captioning feature that allows for users to submit a video and return subtitles that are automatically generated. However, one of the problems with such programs is that they are not able to accurately transcribe the speaker's' words. Because it relies on speech recognition software, many of the words are inaccurately transcribed to a similar sounding word or phrase.

Another important challenge is to identify the speaker in a video. The initial step is to use face detection on the video. However, face detection is not straightforward, because there are lots of variation to image appearance, such as different poses, motion blurring, different lighting, and the speaker being off stage. There have been many methods used to try to accurately identify faces, including the template-matching methods, feature invariant approaches for feature detection, appearance-based methods for using eigenface, and neural network methods. Another problem is getting the computer to accurately identify the person speaking using mouth detection.

In this project, a program was created to make the automated subtitling process more accurate by using a transcript of the video as an input. It also displayed the words of the speaker above the character in the form of speech bubbles.

Methodology

Voice Detection:

- Detect when people are speaking using Google WebRTC Voice Activity Detector
- Audio clips transcribed using Google Speech Recognition software
- Compare lines in the transcript to the script to determine the exact words used

Face and Mouth detection:

- Use OpenCV and both the Haar Cascades and the Dlib facial landmark detection libraries
- Compare successive frames to determine speaking mouth

Results

Voice Detection

When the program was run for a sample audio clip, most of the files were less than 10 seconds each, and was successful in detecting human voices. However, the program also inaccurately detected sounds as positive that are similar to human voices, such as the sound of crumpling paper.

Following the voice activity detection, the program implemented Google Speech Recognition to transcribe each audio frame, returning -1 if no speech was recognized. The resulting text file consisted of both accurately and mistranscribed words compared to the actual audio. As seen in Table 1, the program returned -1 for the sound of paper turning, and -1 for the last segment, “His name was Wesley.” There were also several errors in the speech recognition, such as mistaking “Florin” for “War” and “Her” for “My.”

Face/Mouth Detection

In order to detect the faces and people in the video, Haar Cascade Classifier object detection was first implemented. The Haar Cascade Classifier was normally able to detect the object it was looking for, in this case a person’s face, but there were also several false positives that appeared. For example, in Figure 1, the face of the person in the video is correctly detected with a green bounding box on the right. However, there is also another green box on its left that indicate that a face was detected where the bricks are located. In addition, when the cascade classifier was used for mouth detection as shown in Figure 2, the classifier correctly detected the mouth, but also falsely detected the person’s eye and an object in the background. Another issue with this method of detecting the speaker’s mouth was that it only gave the bounding box of the mouth. Because none of the features of the mouth itself are detected or analyzed, it made it very difficult for it to be used to figure out whether the mouth was moving to recognize the speaker.

To move around this issue of detecting the features of a person’s face, a facial landmark detection algorithm was implemented. Using the Dlib library, a facial landmark detection algorithm was applied to the video. As seen in Figure 3 in blue, the algorithm detects and outlines the main features of the face, such as the eyebrows, eyes, nose, and mouth. In addition, it is also able to locate the actual face with a red bounding box, similar to what the Haar Cascade Classifier was able to do.

Table 1
Comparison of the actual transcript versus what the program returned for a 20 second clip from The Princess Bride

Transcript	Program Results
-The Princess Bride -(Paper Turning) -By S Morgenstern. Chapter 1. -Buttercup was raised on a small farm in the country of Florin. -Her favorite pastimes were riding a horse and tormenting the farm boy that worked there. -His name was Wesley	-The Princess Bride --1 -is Morgenstern chapter one -Buttercup was raised on a small farm in the country of War -my favorite pastimes were riding a horse and tormenting the farm boy that worked there --1

Table 2
Sample start and end times for the audio clips created for a 20 second clip from The Princess Bride

Start Time	End Time
0.03	3.18
3.39	4.05
105.21	105.99
60.48	61.53

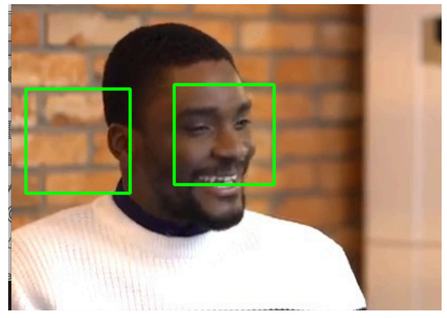

Figure 1. Example of an implementation of haar cascade classifier for face detection.

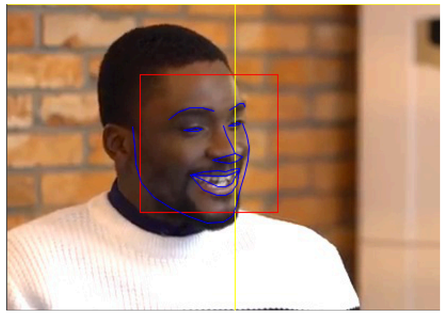

Figure 2. Example of an implementation of facial landmark detection using the Dlib library.