Automatic Lip Detection

Miniproject report

A. Ciontos & L.M. Fenoy

Aalborg University
Faculty of IT and Design

Contents

Chapter 1

Introduction

In the past decades, detection and tracking of facial features has been receiving a lot of attention. In this category falls also mouth detection. This is particularly important as the mouth gives important information about the overall facial expression of the subject. At mouth level it is possible to describe emotions or spoken messages, essential for various applications. The goal of this project is to develop a mouth detection solution. Some of the applications for this are face expression recognition or visual speech recognition (lip reading).

This project presents a way of detecting lips in a pre-recorded video file as well as on live webcam video stream. This implies two stages of processing. First, the lips have to be detected, then they have to be tracked throughout a stream of images (video). This project is implemented using the Python programming language and OpenCV [1]. Haar Cascades are used for feature extraction, with pre-trained models. Further on, OpenCV specific algorithms are used for mouth detection. Nonetheless, this project also presents a method for building ground truth with the use of color segmentation.

Chapter 2

Solution design

This chapter presents background knowledge on object detection, detection methods, as well as the implementation of a lip detection algorithm and tests of the software.

2.1 Object detection

Object detection is a process which enables a computer to recognise a specific object in digital images or videos. Detection methods can be divided into different categories depending on the approach used to design the algorithm. [2]

- **Knowledge based methods**. They depend on a set of rules based on human knowledge. For example, if we wanted to detect a face, some of the rules would be: two eyes, one nose, one mouth, etc. And usually, the rules capture the relationships between facial features, such as distance, proportion, etc. This method however, is not very efficient.
- **Feature invariant methods**. This method relies on extracting different structural features which are independent from pose, viewpoint, or lighting conditions.
- **Template matching based**. They use pre-defined or parameterised face templates to locate or detect the faces by the correlation between the templates and input images. They define the concept of face as a function, like a general template for all faces.
- **Appearance based**. This method is used to find out face models. They use classification methods and rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face images.

2.2 Haar Cascades

In order to do mouth detection, we chose Haar Cascade Classifier, which is a machine learning algorithm based on the concept of features. On a high level, the classifier is trained with images which either represent the object of interest (positive examples) or do not (negative examples). And after it is trained, it can be presented with a new input, and the classifier will output a 1 if it detects the object of interest, and a 0 otherwise.

Taking a closer look, this is how the algorithm works at a lower lever. The classifier uses Haar-like features, which can be visualized in Figure 2.1. These features represent a single numerical value, obtained from subtracting the sum of pixel values from the white regions, from the sum of the pixel values under the black regions. [3]

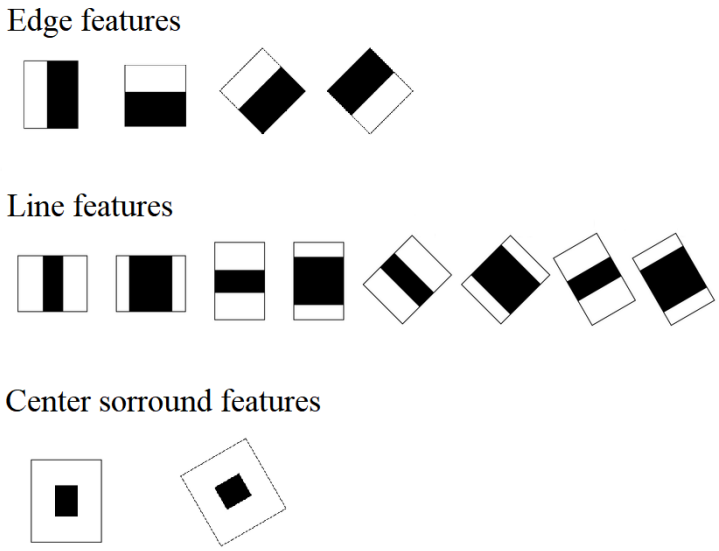

Figure 2.1: Common Haar features

As it can be seen in Figure 2.1, certain features are used for specific purposes, such as edge or line detection.

In order to make the classification, Adaptive Boosting (AdaBoost) is used 2.2. Adaboost is a machine learning algorithm which has the following characteristics: it is composed by a collection of stumps (decision trees with just two leafs). Each stump has an individual weight (calculated in the training), which will make them more or less decisive in the classification. In AdaBoost, the stumps are codependent. Decisions of previous stumps are taken into consideration for the current

stump.

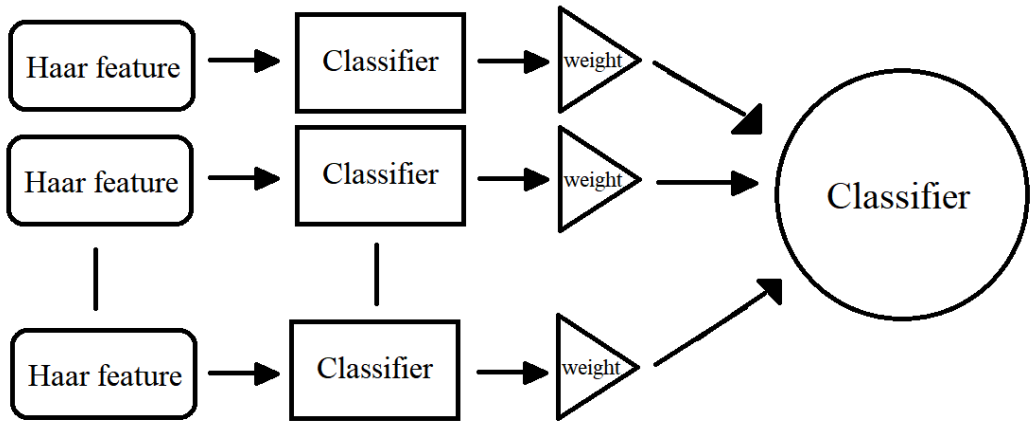

Figure 2.2: AdaBoost Classifier

The classifier is called a cascade because it consists of several simpler classifiers. These weak classifiers depend on only a single feature and are applied subsequently to the same region in a sliding window manner. Therefore, the first classifier is applied (which corresponds to the stump with the higher weight) and it continues to run through the rest of the classifiers until one of them is rejected (meaning the object is not present in this region) or all of them are accepted. Once it finishes with that region, it slides to the next one and repeats the process until the whole image is covered. This process can be visualised in Figure 2.3.

Figure 2.3: Visualisation of how Haar Cascade features are applied to an image

2.3 Implementation

For lip detection, two different types of implementations have been developed. First the Haar cascades were used for smile detection. The pre-trained algorithm with positive and negative samples is imported via OpenCV, and applied to a video stream. This video stream can come from either an *.mp*4 file saved on the

disk or direct webcam video capture. The video is being read frame by frame and the `detectMultiScale` function is used to find the object of interest, in this case the lips. This function returns four values that represent a rectangle containing the region of interest (ROI), these values are the x and y coordinates of the rectangle and w and h which are the rectangle width and height respectively.

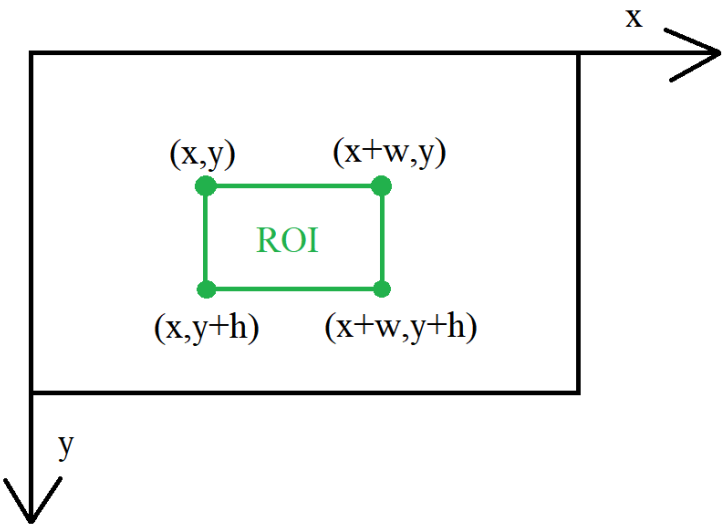

Figure 2.4: Figure showcasing the ROI rectangle coordinates

However, at this level, only the numerical values are returned, and an easy way to test whether or not these values are correct is to project or draw a rectangle on the video stream. To do this we use the built in `rectangle` function, which takes as parameters the frame, the coordinates, a color specified in BGR (blue-green-red) and a value for the line thickness. Figure 2.4 is a representation of how the coordinates make up the rectangle for the ROI in a given frame.

The results of this implementation have not been very successful. The smile detection has been very inconsistent and erroneous. It has been speculated that it could be because of the lack of any type of context in the image, where it is difficult to find a specific feature with no background information. To fix this issue, another implementation has been developed. This time, apart from the lip detection, face detection is also performed.

In this case, following the same algorithm, both face and smile Haar cascades are used. First the face is detected with coordinates x,y and w,h for width and height, then the smile coordinates are derived relative to the face as sx and sy as well as sw and sh for width and height respectively. Different rectangles are drawn for both ROIs. The algorithm performed much better in this case, with a lot fewer

false positives regarding the detection of the lips.

An additional feature for this implementation is detection of smile ratio, meaning the ratio between the ROI width and height. This ratio can be set to a certain threshold and the lips are detected only when they are within the preset values. This is especially important when lip reading, as the mouth is shaped in different forms when it comes to pronouncing words. For example, when pronouncing 'a' the width of the lip ROI becomes lower than the height, but when pronouncing 'e', the width becomes larger than the height.

Further on, an implementation for red color segmentation was developed. The purpose of doing this is to follow the true position of the lips in the frame. This is done by setting BGR threshold values for the shade of red to be detected lower threshold : `([161, 155, 84])`, upper threshold: `([179, 255, 255])`. In a similar fashion as the lip detection using Haar cascades, a rectangle is drawn on the frame containing the red color. Assuming that the frame is in a control environment, and the user uses some lip color enhancement (i.e. red lipstick), the software performs very well, it is very precise, thus it is easy to measure the true movement of the lips in the frame. To eliminate the need for any enhancements, such as lipstick, the color thresholds can be adjusted to be more adequate to natural lip color. Figure 2.5 showcases the functionality of the color segmentation.

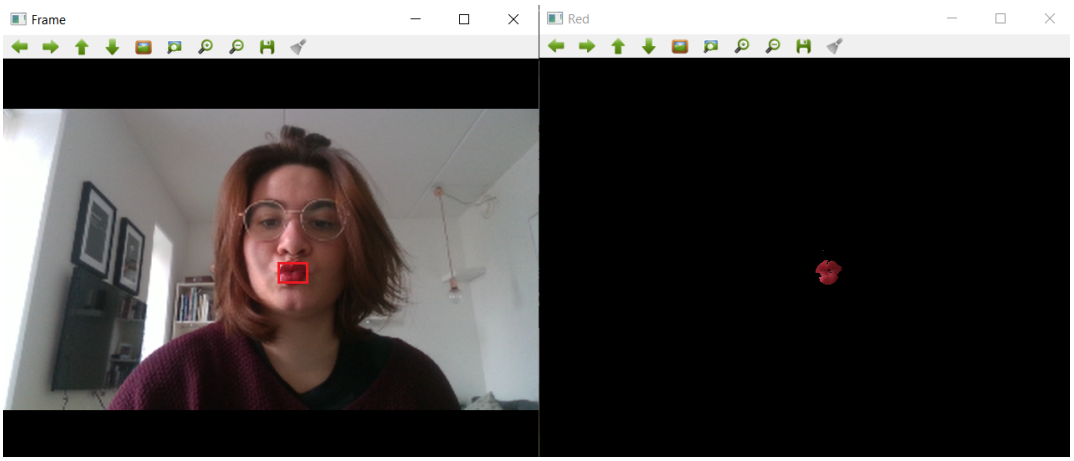

Figure 2.5: The Figure shows the colour segmentation

2.4 Testing

As Section 2.3 suggests, different types of implementations have been developed. To determine the performance of each implementation, several tests have been

carried out.

1. Test performance of mouth detection.
2. Test performance of mouth detection together with face detection.
3. Test performance of color segmentation.
4. Compare performance of Test 1 and Test 2.
5. Get the software with best recognition rate and compare it to Test 3.

Test 1

Purpose: The purpose of this test is to see the detection rate of the mouth. This includes also the presence of false positives which appear as extra ROI rectangles in each frame, where the system thinks there is a mouth, but there is none.

Protocol: To perform this test, 3 random frames are extracted from a 6 second video using the lip detection software.

Results: In Figure 2.6 it is observable that the detection rate is faulty, with the presence of a lot of false positives.

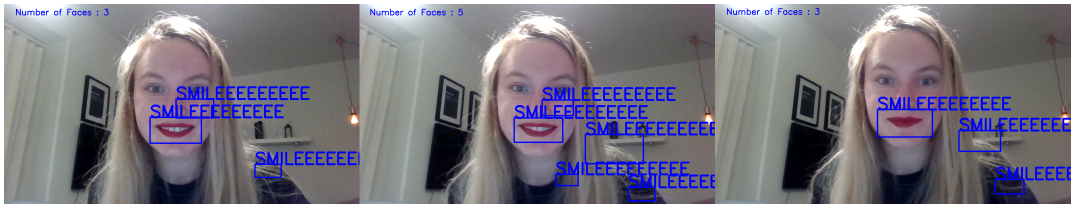

Figure 2.6: Figure showcasing the performance of mouth detection without face information

Test 2

Purpose: The purpose of this test is to see the detection rate of the mouth given that the face is also being detected. The idea is to detect presence of false positives, as well as accurate detection of the mouth.

Protocol: To perform this test, 3 random frames are extracted from a 6 second video using the face and lip detection software.

Results: In Figure 2.7 it is observable that the detection rate is accurate. The face is detected correctly, subsequently, the lip detection is also correct.

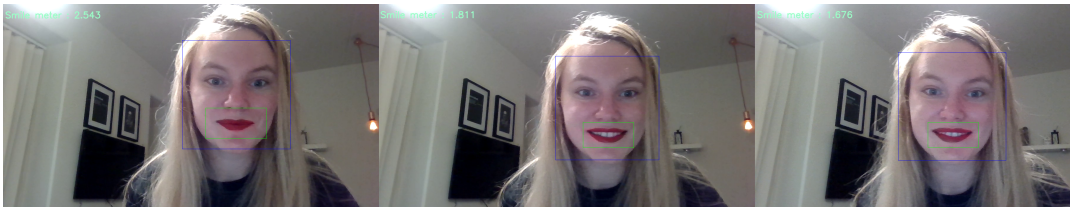

Figure 2.7: Figure showcasing the performance of mouth detection together with face detection

Test 3

Purpose: The purpose of this test is to see how the color segmentation behaves, noise detection and lip color detection.

Protocol: To perform this test, 3 random frames are extracted from a 6 second video using red color segmentation software.

Results: In Figure 2.8 it is observable that the detection is accurate, with no significant noise in the image mask or in the segmentation. However, this software is prone to light changes and is highly sensitive to it, because different type of lighting might interfere with the color, which then might be falling outside the threshold values.

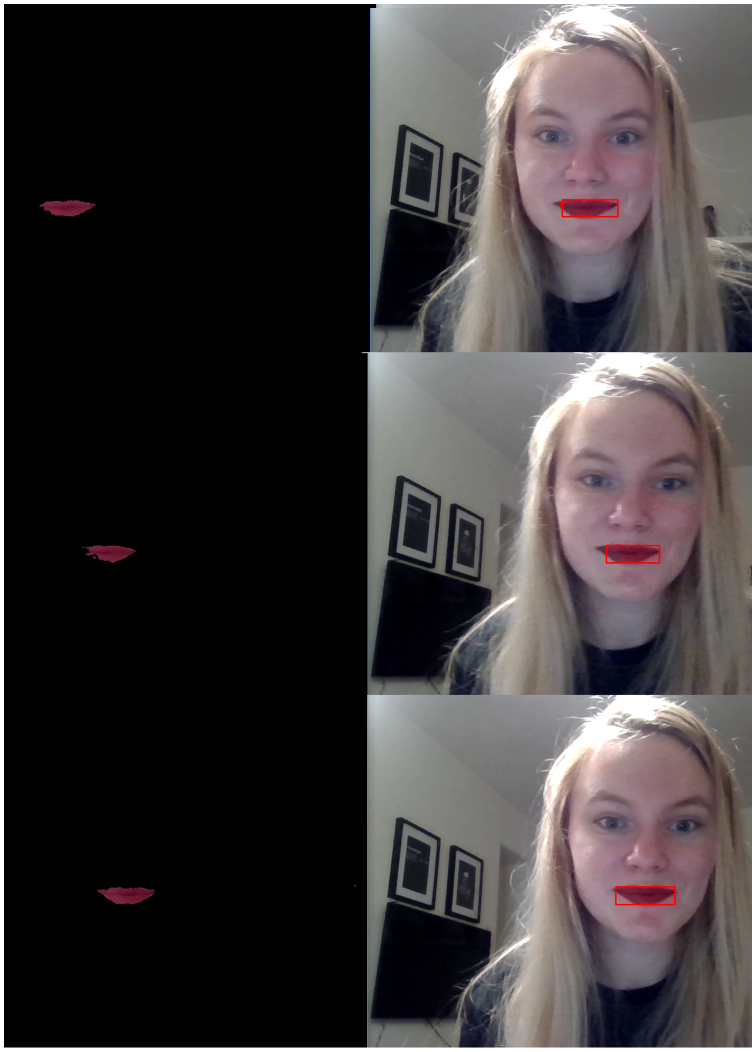

Figure 2.8: Figure showcasing the performance of mouth detection together with face detection

Test 4

Purpose: The purpose of this test is to compare the performance of mouth detection with and without face detection.

Protocol: To perform this test, we compare Figure 2.6 and Figure 2.7.

Results: The mouth detection together with face detection performs better than its counterpart which does only mouth detection. This is because it is easier to detect a mouth when there is some information about facial features. It is more likely to find a mouth where there is already a face.

Test 5

Purpose: The purpose of this test is to compare the performance of face and mouth detection presented in Test 2 to the performance of the red color segmentation. The latter one serves as ground truth for the measurements.

Protocol: To perform this test, we record the sx and sy coordinates of the bounding box from face and smile detection which are the coordinates of the box surrounding the lips and compare them to the x and y coordinates of the bounding box surrounding the lip area in the color segmentation. All coordinates are extracted from frames in the same 6 second video.

Results: The x and y values of both software are very close, given the fact that the bounding box area is slightly smaller for the red color segmentation (red detection) than the bounding box for the mouth detection. Given the values in Table 2.1 it can be concluded that the mouth detection together with the face detection performs very well when comparing it to the ground truth.

Mouth Detect. x	Mouth Detect. y	Red Detect. x	Red Detect. y
607	390	766	437
606	391	659	430
601	392	715	461
597	390	647	436
592	389	759	434
589	390	650	428
581	389	643	428
594	403	644	427
588	402	637	428
582	401	676	427
575	403	670	456
573	404	639	427
565	403	624	435
559	403	639	426
553	404	697	464
548	404	736	440
540	404	736	437
535	402	624	434
530	401	628	431
528	403	630	426
528	404	669	426
527	404	686	465
528	406	650	456
528	406	648	442
529	405	622	442
534	405	732	440
541	407	624	439
546	407	622	439
555	408	734	437
563	406	618	434

Table 2.1: Mouth Detect. x and y = coordinates of the mouth detection; Red Detect. x and y = coordinates of red detection

Testing conclusion

To conclude the tests, the results have been mixed regarding the different implementations, however, when detecting lips together with the face, the detection rate performs accurately, even when comparing it to ground truth. The difference in coordinate values is given by the difference in bounding box sizes. In this case they are negligible, as they can also be tested on direct video stream, which is a success.

Furthermore, the software can detect multiple faces and lips as it can be seen in Figure 2.9

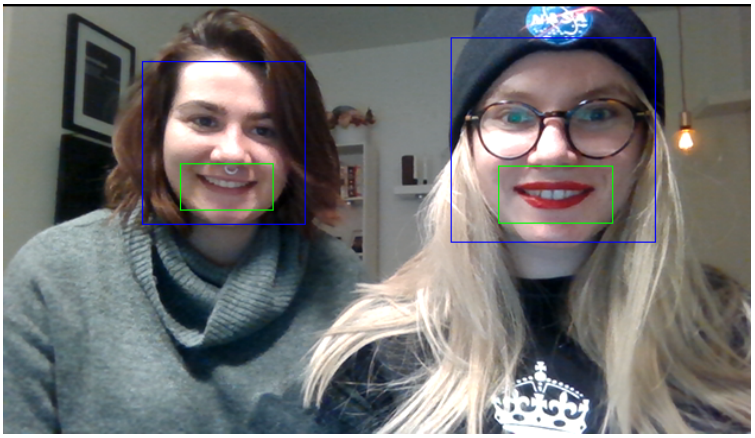

Figure 2.9: The algorithm can track multiple instances

Chapter 3

Conclusion

In conclusion, the lip detection software yielded good results when combined with face detection. However, the detection method runs an image frame by frame and is continuously trying to find the lips in each frame. This is a slow process, and sometimes faulty. In order to make the algorithm more robust and computationally less expensive, the detection could be combined with a tracking method, such as the Kalman filter[4]. It could predict the position of the mouth for the next frames and the detection algorithm would only have to be applied to the region where the mouth is expected to appear next.

Bibliography

[1] OpenCV. *The OpenCV Library*. https://opencv.org/.

[2] Dr Qaim Mehdi Rizvi. "A Review on Face Detection Methods". In: *Journal of Management Development and Information Technology* 11 (Feb. 2011).

[3] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[4] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. Tech. rep. Chapel Hill, NC, USA, 1995.