

# Smile Detection

Sai Kumar Allam, Suma Chilukuri, Snigdha Reddy Dodla, Wu Peng, Praneeth Reddy and Jingli Zhang

## ABSTRACT

Smile detection in real-life face images is potential for many applications such as photo selection, patient monitoring. Many machine learning approaches have produced some high performance for smile recognition. However, extracting effective feature for smile face detection is still a challenging task for practical application. To detect smiles, we first extract facial features from Facial Landmark Detection by using Dlib, then append the bounding information to both positive and negative input images respectively and train the dataset using Haar cascade classifier. We trained a dataset of 2000 positive smile images and 1000 negative images. On the test dataset, we calculate a evaluation matrix and get a better results even for blurred, low intensity and images with comparatively low dimensions.

**Index Terms**— Smile detection, Feature extraction, Haar cascade classifier

o

## I. INTRODUCTION

Smile is the most important and common facial expression that occurs in human's daily life. It often indicates joy, happiness, satisfaction[1] or other positive emotions. Therefore, detecting smiles can be used to estimate a person's mental state. Film studios and ad agencies could test viewers' reactions in real-time for building powerful recommender systems which in turn can lead to adding market value for the firm. During the process of smile detection, smile face feature extraction is the most important part for detection later. Many algorithm[5] has been produced for extracting smile face features, but how to extract diverse and effective feature in smile face to improve detection accuracy is still a research interest.

Our system add facial features to input images first. Facial landmarks are the indexes used to localize and represent salient regions of the face, such as eyes, noses, mouth. After the landmarks are detected, a bounding rectangle is obtained for the pixels with those landmarks that localize the mouth part in the face. For each input image, its location, No. of positive samples and dimensions of the bounding box are appended for the text file meant for positive data. For negative data, we just add the image location to text file. Then we obtained our training dataset. For the training part, we used Haar cascade classifier for mouth features and got a harr xml file, which is used for the detection part.

## II. RELATED WORK

Driven by the need of smile detection application in mobile media and business, some academic research have developed some smile

detection technique. For example, in 2011, Caifeng Shan presented a approach of using the intensity differences as the feature[11]. In 2015, Tong zhang, el propose to extract high-level feature by a well designed deep convolutional neural networks, and on the GENKI dataset, the method reduce the error rate by 21% compared with the previous method.

Besides the techniques, many applications of smile detection in practice has been presented, such as interactive systems (e.g., gaming), product rating, distance learning systems, video conferencing, and patient monitoring. For example, Sensing component company Omron [2] released smile measurement software. It can automatically detect and identify faces of one or more people and assign each smile a factor from 0% to 100%. In the past years, many smile detection algorithms are proposed. For example, in [3], local binary patterns (LBP) were used as main image descriptors for smile detection. The authors reported a classification accuracy of 90% using support vector machines (SVM) and a small dataset of 5781 images.

However, the labelled dataset of smiling face is far less than other detections. An important contribution to the field of automatic smile detection was introduced by [4]. They collected the GENKI-4K database, which contains 4000 real-life face images, downloaded from publicly available Internet repositories, which have been labeled as either smiling or non-smiling by human coders. And this is the exact dataset we used.

In the next section, we would introduce our method of Methodology and implementation. In section 4, we would analyze the results by calculated the evaluation metrics separately for positive and negative images by randomly

considering 30 images each. Section 5 will draw a conclusion of our smile detection system.

### III. METHODOLOGY AND IMPLEMENTATION:

We have considered only frontal faces for this smile detection project.

Object Detection is implemented in three phases:

- A. First a model of algorithm is used to generate Regions of Interest(ROI). These regions proposals are a large set of bounding boxes spanning the full image.
- B. In the second step, visual features are extracted for each of the bounding boxes, they are evaluated and it is determined whether and which objects are present in the proposals based on visual features (i.e. an object classification component).
- C. In the final post-processing step, overlapping boxes are combined into a single bounding box (that is, non maximum suppression)

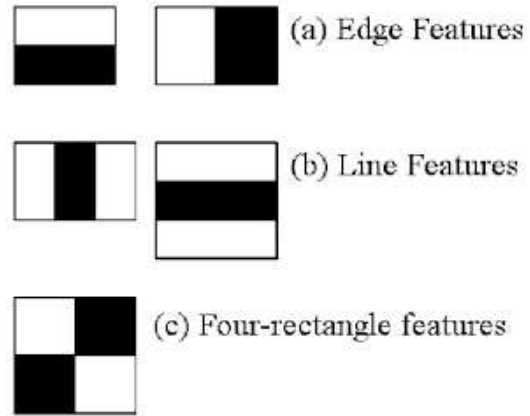
Here the dataset considered is “GENKI-4k”[6]. It contains 4K frontal-face images along with expression (smile=1, non-smile=0) labels and pose labels (yaw, pitch, and roll in radians). The file "labels.txt" contains these labels, and the Nth line of the file corresponds to N in the "files" directory.

In order to detect smile emotion on human faces, object detection technique is implemented for facial and mouth features using “HAAR” Feature[7] based Cascade classifier[8].

#### *HAAR Features and Cascade Classifier:*

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

For example, consider face detection. Initially, the algorithm needs a lot of positive images and negative images to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



Now, all possible sizes and locations of each kernel are used to calculate lots of features. For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. But among all these features calculated, most of them are irrelevant. To select the best features, we can use “Adaboost”.

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier.

To implement this idea, the following things need to be accomplished:

- A. Creating the training dataset by extracting the frontal face and facial feature “mouth” from the image.
- B. Train the Cascade classifier on the data using “Haar” features
- C. Face and Smile Detection
- D. Analysis.

#### *A. Creating the Dataset:*

Firstly, the face part in the image is localized using dlib library function “get\_frontal\_face\_detector” which yields a bounding box. Then facial landmarks[9] are detected with-in the extracted bounding box around the face.

### Facial Landmarks

Facial landmarks are the indexes used to localize and represent salient regions of the face, such as Eyes, Eyebrows, Nose, Mouth and Jawline. Detecting facial landmarks is a set of the shape prediction sub problem. Given an input image (and normally an ROI that specifies the object of interest), a shape predictor attempts to localize key points of interest along the shape.

Detecting facial landmarks is therefore a two step process:

Step #1: Localize the face in the image.

Step #2: Detect the key facial structures on the face ROI.

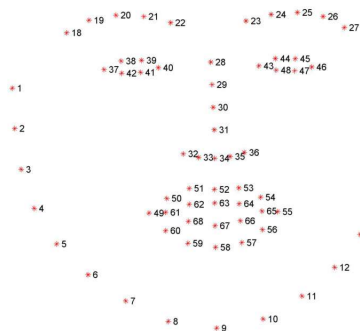
After localizing the face using dlib, the next step is to detect the key facial structures on face ROI. This method starts by using:

1. A training set of labeled facial landmarks on an image. These images are manually labeled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.
2. Priors, of more specifically, the probability on distance between pairs of input pixels.

Given this training data, an ensemble of regression trees are trained to estimate the facial landmark positions directly from the *pixel intensities themselves* (i.e., no “feature extraction” is taking place).

The pre-trained data inside dlib is based on i-BUG 300-W[10] dataset which yields 68 (x,y) coordinates on the localized face.

The indexes of the 68 coordinates can be visualized on the image below:



After the landmarks are detected, a bounding rectangle is obtained for the pixels with those landmarks that localize the mouth part in the face that is coordinates: “49-67” on the above image. Then, the mouth ROI is obtained with the boun

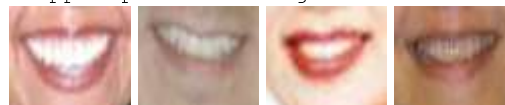
ding rectangle parameters capturing entire mouth features as follows:

```
roi = im[y-5:y + h+5, x-5:x + w+5]
```

Then the mouth ROI is resized to (60,50) and (100,100) dimensions for positive and negative images respectively with “Inter\_cubic” interpolation method.

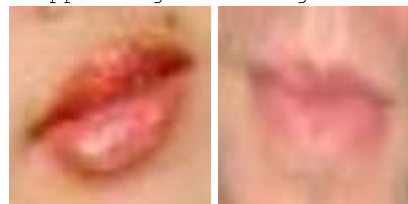
```
roi=cv2.resize(roi, (60,50), interpolation  
=cv2.INTER_CUBIC)
```

Cropped positive Images:



```
roi=cv2.resize(roi, (100,100), interpolati  
on=cv2.INTER_CUBIC)
```

Cropped Negative Images:



For each cropped image, it's location, no. of positive samples, top-left (x,y) coordinates and its size dimensions are appended to the text file meant for positive data if it has a smiling label “1” of the input image.

crop\_resize\_pos1.txt:

Crop\_resize\_pos/crop0.jpg 1 0 0 60 50

Crop\_resize\_pos/crop1.jpg 1 0 0 60 50

Else, just the cropped image location is appended to text file meant for negative data which corresponds to smiling label “0” of the original input image

Crop\_resize\_neg1.txt:

Crop\_resize\_neg/crop2162.jpg

Crop\_resize\_neg/crop2163.jpg

### B. Training the Dataset:

#### Cascade Classifier Training:

For training we need a set of samples. There are two types of samples: negative and positive. Negative samples correspond to non-object images. Positive samples correspond to images with detected objects

Info.dat file is created for the positive samples and bg.txt file(that contains path to each image by line) is created for the negative samples.

For training a cascade classifier, we need to create the vector file, which is basically where we stitch all of our positive images together. We will actually be using opencv\_createsamples command.

For training, the number of positives will be more in number when compared to that of no of negatives.

OpenCV offers two different applications for training a Haar classifier: opencv\_haartraining and opencv\_traincascade. We are going to use opencv\_traincascade since it allows the training process to be multi-threaded, reducing the time it takes to finish.

Eg Command:

```
opencv_traincascade -data data_haar -vec
positives.vec -bg bg.txt -numPos 2500 -numNeg
2000 -numStages 15 -w 20 -h 20
```

The more the no of stages, the longer it will take and it is exponential. The first stage is pretty fast usually, stage 5 much slower, and stage 10 is even slower. The haar training generates a xml file when the process is completely finished. Parse the cascade.xml file using cv2.CascadeClassifier().

#### C. Face and Smile Detection:

For detecting the faces, we have used built-in frontalface haar cascade classifier.

To load the webcam we used VideoCapture(0) of cv2 class

```
video_capture = cv2.VideoCapture(0)
```

For each frame, we will read it as video\_capture.read(). Then convert it from rgb to grayscale. detectMultiscale() is applied on each frame for detection with the following parameters(gray, 1.3,5) .Detection is performed on gray and with the described attributes. For drawing the rectangle we

use cv2.rectangle(). We will find the smile within the region of the space that the face is in, using the same detectMultiScale() method and rectangle is drawn for smile using (sx,sy,sw,sh) in smile. The to display cv2.imshow is used.

The detectMultiScale for face and smile are as follows:

```
smiles = smile_cascade.detectMultiScale(roi_gray,
1.3, 20) for positive images
```

```
smiles = smile_cascade.detectMultiScale(roi_gray,
1.3, 15) for negative images
```

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

#### D. Analysis:

Based on the detected rectangle around the mouth on the input image's face, the person's emotion is analyzed as "happy" if the smile is detected.

### IV. RESULTS AND DISCUSSIONS:

We have tested our smile detection system with evaluation metrics accuracy, precision and recall. We have calculated these metrics using true positives(TP), true negatives(TN), false positives(FP) and false negatives(FN).

TP = A person is smiling and his mouth is detected in smile detector.

TN = A person is not smiling and his mouth is not detected by smile detector.

FP = A person is not smiling but his mouth is detected by smile detector.

FN = A person is smiling but his mouth is not detected by smile detector.

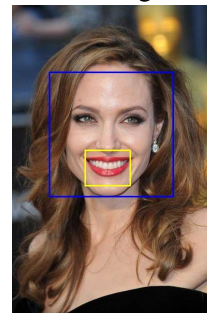
Accuracy =  $TP+TN/(TP+TN+FP+FN)$

Precision =  $TP/TP+FP$

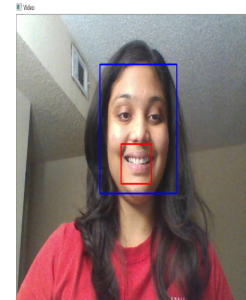
Recall =  $TP/TP+FN$

Results:

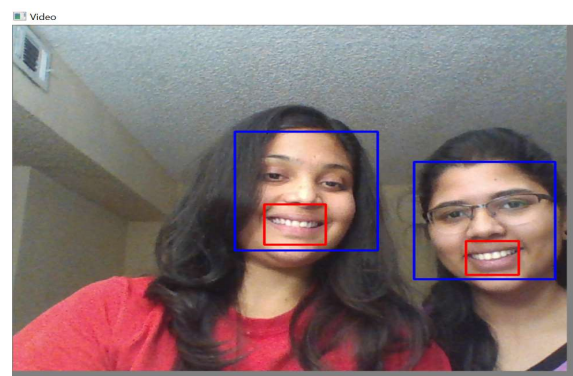
Normal Image






Video Capture



Video Result with two faces



We have calculated the evaluation metrics separately for positive and negative images by randomly considering 30 images each.

Test Image			
Type of Image	Positive	Positive	Negative
Smile Detected or not	Yes	Yes	No
TP for 30 random images	27	27	28
FP for 30 random images	0	0	2
Precision for 30 random images	100%	100%	100%
Recall for 30 random images	90%	90%	93.3%
Accuracy for 30 random Pos_test/neg_test images	90%	90%	93.3%

This smile detection system is giving better results even for blurred, low intensity and images with comparatively low dimensions. It doesn't work well if a person have heavy beard or mustache.

## V. CONCLUSION AND FUTURE WORK

We have proposed a relatively simple and accurate real time smile detection system that can easily run on a common personal computer and a webcam. We

have taken only frontal faces for face detection and we would like to extend this by considering multiple cues. Developing expression recognition systems that are robust to pose and feature variations which include beard etc will be an important challenge for the near future.

## VI. REFERENCES

- [1] P. Ekman, "Self-deception and detection of misinformation," Self deception: An adaptive mechanism, pp. 229–257, 1988.
- [2] Omron, "OKAO Vision," [http://www.omron.com/r\\_d/coretech/vision/okao.html](http://www.omron.com/r_d/coretech/vision/okao.html), 2009.
- [3] D. Freire-Obregon, M. Castrillón-Santana, and O. D'Eniz-Suárez, "Smile detection using local binary patterns and support vector machines," Proceedings of Computer Vision Theory and Applications (VISAPP09), 2009.
- [4] J. Whitehill, G. Littlewort, I. Fasel, M. Bartlett, and J. Movellan, "Toward practical smile detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 11, pp. 2106–2111, 2009.
- [5] Zhang T, Zheng W, Cui Z, et al. A Deep Neural Network-Driven Feature Learning Method for Multi-view Facial Expression Recognition[J]. IEEE Transactions on Multimedia, 2016, 18(12):2528-2536.
- [6] Xin Guo, Lusía F. Polanía, Kenneth E. Barner, Smile Detection in the wild based on transfer learning, 2018.
- [7] R. Leinhardt, Jay Maydt, An extended set of Haar-like features for rapid object detection, 2002.
- [8] Paul Viola, Michael Jones, Rapid Object Detection using a boosted cascade of simple features, 2001.
- [9] B.A. Efraty, M. Papadakis, S. Shah, A. Proffitt, I.A. Kakadiaris, Facial Component-Landmark Detection, 19 May 2011.
- [10] Christos Saganos, Georgios Tzimiropoulos, Stefanos Zafeiriou, Maja Pantic, 300 Faces in-the-Wild Challenge: The First Facial Landmark Localization Challenge, 2014.
- [11] Shan, C. An efficient approach to smile detection[C], IEEE International Conference on Automatic Face & Gesture Recognition and Workshops. IEEE, 2011:759-764.