Facial Landmark Detection

By:
Satyam Nikhra
Hinal Shah
Thong Pham
Contents

- Introduction
- Techniques
- Live Emotion Detection
- Live Face Swapper App
- Face Recognition
Introduction

In this project we have done modules which are based on facial landmark detection such as facial emotion detection, face swapper, face recognition.

In our presentation we will going to explain the techniques which we used and high level process of our implementation.
Techniques to Detect Facial Landmarks

- HAAR Cascade Classifier
- Dlib `get_frontal_face_detector` and `face predictor` which is based on Face Alignment with an Ensemble of Regression Trees
HAAR Cascade Classifiers

What are HAAR Cascade Classifiers?

An object detection method that inputs Haar features into a series of classifiers (cascade) to identify objects in an image. They are trained to identify one type of object, however, we can use several of them in parallel e.g. detecting eyes and faces together.
1. HAAR Classifiers are trained using lots of positive images (i.e. images with the object present) and negative images (i.e. images without the object present).

2. We then extract features using sliding windows of rectangular blocks. These features are single valued and are calculated by subtracting the sum of pixel intensities under the white rectangles from the black rectangles. However, this is a ridiculous number of calculations, even for a base window of 24 x 24 pixels (180,000 features generated). So the researchers devised a method called Integral Images that computed this with four array references.
HAAR Classifiers Explained

3. However, they still had 180,000 features and the majority of them added no real value.

4. Boosting was then used to determine the most informative features, with Freund & Schapire’s AdaBoost the algorithm of choice due to its ease of implementation. Boosting is the process by which we use weak classifiers to build strong classifiers, simply by assigning heavier weighted penalties on incorrect classifications. Reducing the 180,000 features to 6000, which is still quite a bit features.
5. Think about this intuitively, if of those 6000 features, some will be more informative than others. What if we used the most informative features to first check whether the region can potentially have a face (false positives will be no big deal). Doing so eliminates the need for calculating all 6000 features at once.

6. This concept is called the Cascade of Classifiers - for face detection, the Viola Jones method used 38 stages.
Researchers are getting better at detecting facial landmarks

• In 2014, Swedish Computer Vision researchers Kazemi and Sullivan, created the One Millisecond Face Alignment with Ensemble of Regression Trees.

• They developed a method to quickly determine facial landmarks in almost real-time! This was a major milestone in Face Swaps!
DLIB

- Run a simple example to show facial landmarks on face

Facial Landmarks Number Order
- MOUTH_POINTS = 48 to 61
- RIGHT_BROW_POINTS = 17 to 21
- LEFT_BROW_POINTS = 22 to 27
- RIGHT_EYE_POINTS = 36 to 42
- LEFT_EYE_POINTS = 42 to 48
- NOSE_POINTS = 27 to 35
- JAW_POINTS = 0 to 17
Facial Emotion Detection

- Using Machine Learning
- Using Point Analysis
Using ML

- Used CNN
- VGG 16 Architecture
- Took Dataset from kaggle
- While testing results were not accurate
Using Point Analysis

● We consider the distances between all distinct pairs of detected 68 landmark points.

● In real time, we detect landmarks and check the displacement of distinct points and detects the emotions.
● The distance from the eyes to the corners of the mouth clearly has the greatest influence.

● This seems reasonable considering the degree to which a mouth is upturned or downturned is one of the clearest indicators of emotional state.

● For example, for smile detection we use relative distance between the lips endpoints.
● For surprise detection,
  ○ We consider lips distance between point 63 and point 67
  ○ And same way we consider distances between eyes and eyebrows points

● Similarly, for sad we check relative distances between lips endpoints 49 and 55 and eyes distance points 38,42 and 45,47
Result

Surprise Detected

Smile Detected
Future Work To Do

- Implementation of Real Time Facial Emotion using CNN after collecting test category images of high resolution
- Try to implement more emotions using point analysis
Face Swapper
Why is Advanced Face Morphing Hard?

- Previously we saw that HAAR Cascade Classifiers provide excellent results in Face Detection
- However, we can’t simply cut someone's face out of one picture and place it onto another and expect realistic results
Face morphing isn’t simply cutting and sticking a face on another image!
What was so hard about that?

1. Identifying Facial Features
2. Warping the image to fit the new and different facial expression
3. Color Matching
4. Creating seamless borders on the edges of the new swapped face
First Half of our Face Swapper

Face Image 1

**detector()** - dlib’s Face Detector
- **Input**: image
- **Output**: rectangles

Face Image 2

Uses dlib function: get_frontal_face_detector()

**predictor()** - dlib’s facial feature extractor
- **Input**: Bounding boxes (rectangles)
- **Output**: Array of x,y coordinates of facial features

(x,y) Landmark points for Face Image 1 & 2

**transformation_from_points()** – Ordinary Procrustes Analysis
- **Input**: (x,y) Landmark points for Face 1 & 2
- **Output**: Transformation Matrix ($M$) that maps points from one face to the next (3x3)
Continued..

get_face_mask() – Generates a mask for an image and a landmark matrix

**Input** = Image 2 and its Facial Landmarks (x,y)
**Output** = An image mask outlining which parts of image 2 will be overlaid on image 1

warp_im() – Maps one image onto another image using Transformation Matrix (M)

**Input** = Image 1, Transformation Matrix (M) & Image 2's shape/matrix dimensions
**Output** = Image 1 with Image 2’s face overlaid over it (warped_mask)

Generate a **Combined Masks** by taking an element-wise maximum

**Input** = Image 1 and is Facial Landmarks (x,y) & the warped_mask
**Output** = A combined mask which ensures the features from image 1 are covered up and the images for image 2 are visible

correct_colours() – Attempts to match skin tone and lighting between 2 images

**Input** = Images 1 & warped_im2 (reverse of warped_mask) and facial landmarks of image 1
**Output** = Color Corrected Warped image 2

Apply Mask to produce final image

**Input** = Color Corrected Warped image 2, the Combined Mask and the Original Image 1
**Output** = Final Face Swapped Image
Result
Demo
Future Work To Do

- Adding more filters
- Smoothening the images
- Develop app for mobile
Face Recognition
Facial Recognition

- Facial Recognition is a task that human abilities are absolutely brilliant, and even some animals, dogs, crows, sheep can do it.

General overview of facial recognition systems steps:

1. Obtain faces & normalize images (face alignment etc.)
   - Build a dataset of face images

2. Detect and record features of face
   - General global features
   - Geometric features such as spatial relations of eyes, nose, mouth etc.
   - PCA or LDA representations
   - Local feature extraction

3. Use features to then classify face or return a confidence/probability value
Facial Recognition with OpenCV

OpenCV comes with 3 facial recognition libraries, all of which operate similarly where they take our dataset of labelled faces, and compute features to represent the images. Their classifiers then utilize these features to classify.

1. Eigenfaces - createEigenFaceRecognizer()
2. Fisherfaces - createFisherFaceRecognizer()
3. Local Binary Patterns Histograms - createLBPHFaceRecognizer()
Facial Recognition with OpenCV

1. Eigenfaces - Uses Principle Component Analysis to reduce the dimensionality of the face. However, this neglects the class label into account and can represent variance from changes in illumination.

2. Fisherfaces – Solves this by using LDA (Linear Discriminant Analysis) that is a class-specific projection, which means it attempts to minimize variance within a class, while maximizing variance between classes.

3. Local Binary Patterns Histograms - Uses local feature extraction while persevering spatial relationships. It divides faces into cells and then compares each cell to face being classified. It then produces a histogram showing the matching values of an area.
Face Recognition

Create our Training Data

- Record 100 Images of your face using the HAAR Cascade Face Detector
- Normalize by gray scaling and re-sizing to 200 x 200 pixels
- Create an array of labels for recorded images (will you only one, yourself in this example)

Classify New Face

- Extract Face from Webcam using the HAAR Cascade Face Detector
- Normalize by gray scaling and re-sizing to 200 x 200 pixels
- Pass face to our model predictor to get label and confidence value
Demo
Challenges

- Tried to use CNN to make it but CNN does not work on 1 class
Future Work

- Try to implement it in using GAN (Generative Adversarial Network)
Questions??