Automatic Facial Symmetry Analysis & Rating

CS 534: Computational Photography - Fall 2015

University of Wisconsin - Madison

Brant Birrenkott, Brett Brown, Connor Neff
Table of Contents

1. Abstract

2. Introduction
   2.1 Motivation
   2.2 Theory
      2.2.1 Eyes and Nose Detection
      2.2.2 Eye Pupil Detection
      2.2.3 Bilateral Symmetry Detection
      2.2.4 Feature Point Detection
      2.2.5 Symmetry Percentage Detection
   2.3 Related work
   2.4 Mission Statement

3. Method
   3.1 Detecting the Face and Nose
   3.2 Calculating the Bilateral Line of Symmetry
   3.3 Mirroring the Images
   3.4 Detecting Facial Landmarks
   3.5 Calculating the Symmetry Percentage

4. Results

5. Discussion

6. Conclusion

7. References
1. Abstract

Symmetry is a significant factor in determining the objective attractiveness of a face. The purpose of this project was to answer the question can the symmetry of a face be objectively measured and improved upon by creating a perfectly symmetrical face in a portrait style image? The project detected the center point of the nose and pupils of the eyes of a portrait style image in order to create the line of bilateral symmetry for the face. It then mirrored one half of the face onto the other in order to create a perfectly symmetrical face using both the left and right halves to make two new images. Lastly, the project detected the facial landmarks of the face and measured the Euclidean norm of each facial landmark in order to compare the location of that feature point on the other half of the face. The ratio of the Euclidean norms was then calculated in order to obtain a percentage. This application could be used for entertainment purposes as well as a face beautification product for portrait style images

2. Introduction

Attractiveness of an individual is often thought of as a subjective measure; however, recent research [5] is bringing to light the brain's subconscious analyzation of a variety of facial features to come to a conclusion of how attractive one's face is. We focus
on the feature of \textit{symmetry} in the face, as numerous studies \cite{11} \cite{7} point to this being a significant factor when determining the objective attractiveness of a face. Using this knowledge, by calculating the symmetry of the face in an image, this should positively correlate to the attractiveness of this face.

\subsection*{2.1 Motivation}

There is a huge industry which revolves around editing photos to make the subject “more attractive”. For better or worse, nearly every image one will find in a magazine or news site would have been retouched in some way. If there were a program which could objectively inform the user with a quantitative numerical scale of how attractive the subject was, this would increase the efficiency of photo-making, and it would allow for higher quality images to be created. Rather than rely on the subjective opinion of a group of individuals, computer analysis is impartial and mathematical.

\subsection*{2.2 Theory}

\subsubsection*{2.2.1 Face and Nose Detection}

The face and nose detection is based on the Viola-Jones algorithm. We used a threshold of 100 in order to make sure facial detection was accurate \cite{9}.

\subsubsection*{2.2.2 Eye Pupil Detection}

The pupil detection is based on the detection of Eigen Pupils in each eye. The Viola Jones algorithm is used to initially find each eye in order to easily detect the Eigen Pupils. \cite{9} Then it uses the Camus & Wildes’ method to select the possible center coordinates of the pupil and the iris. The method consists of thresholding followed by checking if the selected points correspond to a local minimum in their immediate neighbourhood. These points serve as the possible center coordinates for the iris. Then Daugman’s method is used to detect the iris. Lastly, using Daugman’s integrodifferential operator, the pupil’s center coordinates are found by searching a 10x10 neighbourhood around the iris center and varying the radius until a maximum is found \cite{18}. 
2.2.3 Bilateral Symmetry Detection

Bilateral symmetry was detected by using the nose detected by the Viola-Jones algorithm [9]. A point was placed at the center of the nose to represent the point of symmetry. It then detected the eye pupils and created a line between the two center points of the pupils. The line of bilateral symmetry was then detected by creating a line perpendicular to the line between the eye pupils that ran through the point of symmetry on the nose [3].

2.2.4 Feature Point Detection

The feature point detection method used the supervised descent method in order to trace the facial landmarks on the image. The facial landmarks that were traced include the eyes, eyebrows, nose, and mouth [17].

2.2.5 Symmetry Percentage Detection

We accomplished symmetry percentage detection by manipulating the values output from running a feature point detector on both versions of the face: the original image, and the two symmetrical versions. The feature point detector outputs the x,y coordinates of each of the feature points so we use the norm() Matlab function on the difference between the original and symmetrical vector of points to acquire the Euclidean norm. By raising the negated Euclidean norm on Euler’s number multiplied by 10, we bound our function such that as the Euclidean distance approaches ∞, our function approaches 0. Conversely, if the norm is equal to 0 (meaning there is no distance between the right and left feature points, thus the face is perfectly symmetrical), our program will score the face as a 10/10. The equation score for a single side of the face is defined as:

\[ 10e^{-\|a\|} = \text{score} \]

Put plainly, the greater the distance between the feature points once the halves have been mirrored on top one another, the lower the score will output (down to a theoretical minimum of 0). The smaller the distance, the higher the score will be (up to a max of 10 if the distance is 0).
2.3 Related work

Symmetry of the face is a reliable trait for attractiveness however there has been few other programs which measure percent of symmetry. One example that does is [http://www.symmeter.com/](http://www.symmeter.com/) which describes itself as a symmeter or a tool which rates the symmetry of your face out of ten. Another method that measure facial symmetry is the Golden Ratio. This method needs a person to manually measure several lengths (i.e. Top of head to chin, Top of head to pupil, etc.) and form ratios from those measured lengths. The golden ratio is said to be 1.62 and the closer you are to that number the more scientifically attractive your face is [6].

2.4 Mission Statement

The goal of our project was to formulate and compute some metric that would serve an objective identifier to the degree of attractiveness of a face. By combining numerous approaches of facial detection and feature analysis, we were able to combine the results into a cohesive whole, with which our algorithm can rate the face on a scale of 1-10 based on the measured symmetry of the face.

3. Method

The overall method to accomplish our goal was a two step process. First our project created an output file which contains the perfectly symmetrical face of an image created by mirroring by the right and left half of the face. Second, our project measured the symmetry of the original face by outputting a value out of ten. To start, an input image must be selected. This input image must be a portrait style photo which has the person completely facing the camera with no head rotation. Head rotation will skew symmetry results because one half of the face will be more evident in the photo.
3.1 Detecting the Face and Nose

Figure 2 & 3: Detection of the face and nose using Computer Vision System Toolbox's Cascade Object Detector [8]

Our project first reads the image and uses the visual cascade object detector from the computer vision system toolbox in order to detect the face in the image. Then it uses the visual cascade object detector again to detect the area where the nose is in the image. The nose detection uses a threshold number of 100 in order to decrease the chances of reading other objects in the image as noses however is not too large where it misses the actual nose itself.
3.2 Calculating the Bilateral Line of Symmetry

Figure 4 & 5: Detection of the center point of the nose and the pupils of the eyes [8]
Figure 6 & 7: The line of bilateral symmetry and the rotation of the head to make the line between the pupils horizontal [8].

From the detection of the face and nose, our project calculates the line of bilateral symmetry of the face. The line of bilateral symmetry is defined as a line “which the left and right sides of the organism can be divided into approximate mirror images of each other...”. To find the line the project first finds the centerpoint of the nose by plotting a point which has an X coordinate that is half the width of the previously defined rectangle added with the starting X coordinate of the rectangle and a Y coordinate that is calculated with the same parameters except using half the height. The point of the nose is used as the centerpoint of the face.

With the center point of the face found, the project then detects head tilt in order to create a line which is perpendicular to a line between the pupils. To calculate the head tilt the project first detects the points at which the eye pupils are located. The project uses the visual cascade object detector to find the eyes and then creates two images, one for each eye, in order to detect the pupils. The two images use the Camus & Wildes’ method to detect the iris and the pupil and uses that detection to create a point for the eye pupils in order to create a line between them. This line is then used to calculate an
angle which is the amount that the image needs to rotate in order for the line between
the pupils to be horizontal. To calculate the angle that is needed to rotate the image we
first need three different points on the image. We need the point of the left pupil, the
right pupil, and the point that is on the horizontal line with the left pupil and the vertical
line with the right pupil. With these three points we can create two vectors, a line vector
between the left and right pupils, and a line vector between the left pupil and the other
calculated point. Once the vectors are created we used the calculation found below to
determine the angle that the image needs to be rotated. After the angle is calculated we
lastly needed to check if the right pupil was higher or lower than the left. If it was higher
we tilt to image down and if it is lower we tilt the image up, we do so by making the angle
of rotation positive or negative depending on the case.

% Creates the two vectors needed to calculate the angle
DirVector1=[p2,q2]−[p1,q1];
DirVector2=[p4,q4]−[p3,q3];

% Uses the vectors to calculate the angle
angle= radtodeg(acos( dot(DirVector1,DirVector2)/norm(DirVector1)/norm(DirVector2) ));

Figure 8: Method of image rotation angle calculation

After the image is rotated according to the head tilt, the bilateral line of symmetry
is plotted. The bilateral line is perpendicular to the line between the eye pupils and
contains the centerpoint of the face as a coordinate. This creates the approximate
mirroring of the left and right sides of the image.
3.3 Mirroring the Images

Figure 9 & 10: The bisection of the face [8]

Figure 11 & 12: The mirroring of the left (11) and right (12) halves of the face [8]
With the creation of the bilateral line of symmetry, the left and right side of the face are then mirrored onto the opposite side in order to create a perfectly symmetrical face. To create these new images the project first bisects the two halves of the face using the bilateral line as one parameter and the rectangle of the face detected by the visual cascade object detector as the other. This creates two new rectangular images which will be written to the output file and used in order to mirror the images.

The images are mirrored by using the bilateral line of symmetry and reflecting one side of the image and translating it to the other. This is done for both halves of the face and written to the output file.

3.4 Detecting Facial Landmarks
After the two mirrored images are written to the output file, the project reads the facial landmarks of the images. This captures several points that highlight the eyes, eyebrows, nose, and lips by using the supervised descent method. The points are recorded for the original tilted image as well as the two mirrored images.

### 3.5 Calculating the Symmetry Percentage

![Method of symmetry percentage calculation](image)

Lastly the percentage of facial symmetry is calculated using the facial points detected in the three images. The project uses the norm() Matlab function to find the difference between the original and symmetrical vector of points. This calculation acquires the Euclidean norm. The score is then derived by raising Euler’s number by the negated Euclidean norm and then multiplying that number by 10. This calculation is done for both symmetrical images and then averaged to find the final symmetry percentage.
4. Results

<table>
<thead>
<tr>
<th>Mirrored Left</th>
<th>Original</th>
<th>Mirrored Right</th>
</tr>
</thead>
</table>

Figure 15: Natalie Portman, best result we have 8.75/10 [12]

Figure 16: Taylor Swift, 8.1/10 [16]
5. Discussion

The results were best for mirroring when hair was either short or symmetrical and the head was not rotated at all. For the calculation of symmetry, the results were best when the head was not rotated in any way and head tilt was not extreme so that calculation between the original and the mirrored images at each feature point was based entirely on the symmetry of the face.

Issues that occurred during the project were due to selecting images that didn’t fit into our criteria. For instance:
Model Olivia Munn should rank fairly high on a facial symmetry scale, but our program ranks her as a 5.95/10. In the original picture (Center), her face is awkwardly rotated to left. This exposes more of the right side of the face so when our program outputs the Left symmetrical (Left) and Right Symmetrical (Right) pictures they give a very skewed image of her. The skewed images as a result create a poor symmetry percentage.

Other issues that occurred were due to image size. In the image below (Left), the size of the photo is so large that the visual cascade object detector is able to detect other faces or noses. The detection of multiple faces or noses creates an error in the program. To correct this either the merge threshold needs to be raised or the image needs to be cropped. In the example below (Right) the image was cropped which resolves the issue.
6. Conclusion

The goal of this project was to compute an objective metric which served as an identifier to the degree of attractiveness of a face. This was accomplished by calculating the Euclidean distances between feature points of the original images and its created mirrored images in order to find a percentage out of 10. The results of these measurements were most accurate when a face was not rotated to one side or the other and head tilt was not too large. Issues did occur when the head was rotated to one side or the image was too large. This project measured a reliable trait of attractiveness which few others have explored. In the future it could serve as a new way to modify or enhance images for entertainment purposes as images become more and more prevalent because of social media and the internet.
7. References


Our team obtained these functions from outside sources with the location referenced by the url's below.
+cv
  ○ private
    ■ CascadeClassifier_.mexw64
    ■ HOGDescriptor_.mexw64
    ■ VideoCapture_.mexw64
  ○ buildPyramid.m
  ○ buildPyramid.mexw64
  ○ CascadeClassifier.m
  ○ cvtColor.m
  ○ cvtColor.mexw64
  ○ HOGDescriptor.m
  ○ resize.m
  ○ resize.mexw64
  ○ VideoCapture.m
  ● detect_image.m
  ● DetectionModel-xxsift-v1.4.mat
  ● drawcircle.m [1]
  ● getEigenEye.m [1]
  ● getEye.m [1]
  ● getEyes.m [1]
  ● haarcascade_frontalface_alt2.xml
  ● InitialTest.fig
  ● lineint.m
  ● Mirror.m [2]
  ● mirrorImage.m [2]
  ● mirrorImageRight.m [2]
  ● opencv_core244.dll
  ● opencv_ffmpeg244_64.dll
  ● opencv_highgui244.dll
  ● opencv_imgproc244.dll
  ● opencv_objdetect244.dll
  ● partiald.m [1]
● search.m [1]
● thresh.m [1]
● TrackingModel-xxsift-v1.8mat
● xx_initializem
● xx_sift.mexw64
● xx_track_detect.m
● xx_track_detect.p

Url's

1)  http://www.mathworks.com/matlabcentral/fileexchange/25056-fast-eyetracking
2)  https://www.mathworks.com/matlabcentral/newsreader/view_thread/166191
3)  

Our team created the following functions

● EyeCrop.m
● FaceCrop.m
● main.m
● tilt.m

Our team approximately wrote **250 lines of code.** Here is how the work broke down.

**Connor Neff:**

● Implemented the Face and Nose Detection of the image
● Modified Mirror Image code to perform face mirroring
● Wrote the Majority Paper

**Brett Brown:**

● Modified the Feature point detection code into our program
● Constructed the Symmetry percentage code
● Developed the majority of the website

**Brant Birrenkott:**

● Constructed the image tilt code by figuring out how to calculate the angle
- Created the code to implement Bilateral symmetry line
- Commented the majority of the code