Facing the differences between Facebook and OpenCV

A facial detection comparison between Open Library Computer Vision and Facebook

Staffan Blomgren
Marcus Hertz

Degree Project in Computer Science, DD143X
Supervisor: Richard Glassey
Examiner: Örjan Ekeberg

CSC, KTH, May 7, 2015
Abstract

Face detection is used in many different areas and with this thesis we aim to show the difference between Facebooks face detection software compared with an open source version from OpenCV. By using the simplest implementation of OpenCV we want to find out if it is viable for use in personal applications and be of help for others wanting to implement face detection. The dataset was meticulously checked to find the exact number of faces in each image so that the optimal result is given. The conclusion of this study is that Facebooks algorithm is better trained and thus has better results, however OpenCV is still a viable choice for your own applications.
Sammanfattning

Ansiktsigenkänning används i många olika områden och i den-
da rapporten presenteras skillnaderna mellan det system Facebook
använder på uppladdade bilder och OpenCV, en öppen programva-
ra utvecklad av Intel. Genom att använda den enklaste versionen av
OpenCV, vill vi ta reda på ifall det är användbart på personlig nivå.
Data kontrollerades för hand för att finna ett exakt antal ansikten
i varje bild, som sedan jämfördes med de båda systemen. Resultaten
pekar tydligt på att Facebooks algoritm är bättre tränad, men att
OpenCV fortfarande är ett dugligt alternativ för egna applikationer.
## Contents

1 Introduction .................................................. 4

2 Background .................................................... 5
   2.1 Haar-like features .................................. 6
   2.2 Variations and other features .................... 7
   2.3 Learning algorithm ................................. 9
      2.3.1 Boosting and learning ...................... 9
      2.3.2 Weighting .................................. 11

3 Method ......................................................... 12
   3.1 The detection approaches ......................... 12
   3.2 Face detection dataset ........................... 12
   3.3 Defining face regions ............................. 13
   3.4 Result assessment ................................ 14

4 Result presentation ........................................ 16

5 Discussion .................................................... 19
   5.1 Limitations ........................................ 19
   5.2 Deviations and special cases ................... 20
1 Introduction

The process of detecting and recognizing a face is one that happens naturally and unconsciously for us humans. We can look at an image and without much difficulty determine whether or not it actually contains one or several faces. We can also decide at which way the face is looking, the gender of the person, various details such as eyes and nose, etc; we can perform this detection process very quickly, despite differences in lighting, obstructions and scaling. Ever since the middle to the late 1970s, that process has been simulated by computers. As more and more people came to realise the potential of computers and computer vision, the field expanded and today, everyone with a smartphone can easily perform face detection through their phones camera.

This paper introduces a broad comparison between the face detection class Cascade Classifier in the OpenCV Python library and the face detection used by Facebook on uploaded pictures. OpenCV stands for Open Library Computer Vision and is an open-source cross-platform library developed by Intel specifying in computer vision, making it one of the easiest libraries to implement and handle in regards to face detection. Facebook, containing what is one of the largest facial image databases on the planet, has implemented their own system for detecting and recognising faces in the uploaded pictures. With its funding and personnel talent, it is interesting to see how Facebook compares against a project that has been in development since the very beginning of the millennia.

An overview of the background in face detection, as well as some technical details to the most common approach is given in section 2. Detailed descriptions of the various aspects of comparison used in this paper, dataset collection and execution are given in section 3. In section 4, the performances of the two systems are displayed and comparisons made. Finally, the comparison is evaluated, concluded and discussed in section 5.
2 Background

The very earliest attempts at face detection was performed as early as the 1970s. Back then, only simple heuristics and human body proportion/measurement techniques were used, which gravely limited the application pool [1]. At this time, assumptions such as direct facial angle and plain background were required. These problems caused more work in the facial detection area to lay largely dormant until the 1990s, when further hardware and software developments enabled practical facial detection to become a reality. Since then, interest has grown significantly due to several different reasons including the need for surveillance-related technologies, hardware development, the emergence of the Internet and neural networks, etc [2].

At first, face detection as a process required information about the human face. This led to the categorization of different techniques, based on how they utilized the knowledge of the human face [1]. For example, one of the earliest approaches was named the feature-based approach and just as the name implies, the techniques makes explicit use of the face knowledge and detect low-level features before analysing further. This was the main method being used and accounted for the majority of research prior to the early 2000s [1]. As pattern recognition theory advanced, another category called the image-based approach was spawned. This technique approaches face detection as a general pattern recognition problem and uses face knowledge implicitly instead of explicitly. Since then, several different techniques and approaches has been reported and been grouped into the four following categories [3]:

- Knowledge-based methods
- Feature invariant methods
- Template matching methods
- Appearance-based methods

Knowledge-based methods work in the same way as the older techniques; they use already known knowledge about the human face (such as proportions, symmetries, etc) to detect a face amongst other things in a picture. Feature invariant methods searches for robust face structure features that can be detected regardless of pose and lighting variations; template matching uses already stored templates of faces and compares them to the image; appearance-based methods learns different models of the face and uses them to deduce whether or not an image contains a face.

Along with the quick increase in computational power and data storage, appearance-based methods have the last decade been performing better than the others [3].
Appearance-based methods

When using an appearance-based method, the software performs facial detection by learning what is a face and what is not. This is done by using a large dataset of faces and non-face examples, and then implement a learning algorithm in order to sort the two possibilities out. In this process, there are two key decisions: what features to extract from the dataset, and what learning algorithm to apply. In order to achieve optimal detection, optimization of both problems must be issued. The following section will examine the recent progress in feature extraction.

2.1 Haar-like features

The one face detection algorithm that it is generally accepted had the most effect on the area of face detection in the 2000s was "Rapid object detection using a boosted cascade of simple features", [4], by Viola and Jones. This algorithm introduced what’s known as Haar-like features, which were quickly calculated using the integral image algorithm (also known as summed area table). The Haar-like features quickly became the most popular method in face detection research and, with variations and improvements, is still being widely used.

![Figure 1: Haar-like features](image)

The integral image is computed using the following formula:

$$I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

where $I(x, y)$ is the integral image for the pixel at location $(x, y)$, and $i(x', y')$ is the original image in its whole. What this essentially means is that the integral image of a single pixel is the sum of all the pixels above and to the left of that pixel. This enables calculation of the Haar-like features shown in Fig 1. For example, the integral image of the rectangle ABCD can be calculated as follows:

$$\sum_{x, y \in ABCD} i(x, y) = I(C) + I(A) - I(B) - I(C)$$
This calculation requires only four lookups in our original image, which enables the Haar-like features to be calculated with very few memory references. These features are defined as the difference in pixel intensity between the rectangles within the feature. Since many of the inner rectangles share corners, it is possible to add rectangles without significantly increasing the number of references (for example, feature (d) from Fig.1 requires eight lookups while feature (e) and (f) requires nine). Fig 2 shows the use of Haar-like features in action with standard face features.

2.2 Variations and other features

This integral image technique quickly showed not to be sufficient as developers started looking at more realistic photographs and pictures - the integral image worked well for frontal face detection, but was limited when it came to angles, lighting, background, etc [3]. This led to several variations of the Haar-like images. One research team led by Lienhart et al. [5] introduced a 45 degree rotation as well as new, center-surrounded, features. These are shown in Fig. 3.

![Figure 3: Rotated and extended Haar-like features](image)

Along with the new features, an integral image table for calculating the rotated features was introduced as follows:

1. Edge features
   1. (a) 2. (b) 3. (c) 4. (d)

2. Line features
   1. (a) 2. (b) 3. (c) 4. (d) 5. (e) 6. (f) 7. (g) 8. (h)
\[ rI(x, y) = \sum_{x' \leq x, |y-y'| \leq x-x'} i(x', y') \]

More detail regarding these features can be found in [6].

Another extension on the Haar-like features was made in [7], in which the research was focused on joint features, basing them on the co-occurrence of several Haar-like features at once. These are represented by combining the values computed from multiple features, in binary.

Fig. 4 shows an example of a joint Haar-like feature based on the co-occurrence of three Haar-like features at once. The value \( j \) represents an index of \( 2^F \) combinations, where \( F \) is the number of combined features. The number \( F \) is of course limited to avoid unrelia-

cibility and keep computations quick. It is then possible to compute and classify faces and non-faces by using this index combined with statistical analysis on big datasets.

Several other methods of feature extraction has been used for face and object detection. One such example is the pixel-based feature approach, in which the software can use a pair of pixels or a set of control points as features. These are generally computationally quicker than the Haar-like features, but since the features themselves are also smaller, the discrimination power is greatly reduced.

Another popular approach is using statistical-based features. This method takes advantage of regional statistics to find edges and other features. By being mainly undeviated by overall brightness changes, it has an easier time detecting structural attributes compared to linear edge filters such as the Haar-like feature. Edge based histograms are, however, not indifferent to scaling and the images must therefore be scaled to ensure reliability in these kind of features.

Table 1 shows some feature extraction methods and their variations.
<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Variations</th>
</tr>
</thead>
</table>
| Haar-like features | - Haar-like features  
- Rotated Haar-like features  
- Rectangular features with structure  
- Haar-like features on motion filtered image |
| Pixel-based features | - Pixel pairs  
- Control point set |
| Statistics-based features | - Edge orientation histograms  
- Spectral histogram  
- Region covariance |
| Composite features | - Joint Haar-like features  
- Sparse feature set |
| Shape features | - Boundary/contour fragments  
- Edgelet  
- Shapelet |

*Table 1: Features and variations*

### 2.3 Learning algorithm

Another way of improving facial detection is to improve the boosting learning algorithm. Boosting is a process in which the program reaches a strong conclusion by combining several weak ones. In other words, in order to achieve a strong classifier we need to combine multiple weak ones. In this section, we expand on the recent progress within boosting algorithms and their applications within the field of facial detection.

#### 2.3.1 Boosting and learning

Along with the newly introduced Haar-like features, Viola et al. used the AdaBoost (Adaptive Boosting) algorithm in what is generally considered the original face detection paper. This algorithm is widely accepted as the first step towards more functional and efficient boosting algorithms. What follows is the derivation of AdaBoost performed in [8].

Assume a dataset $S = (x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$ in which each item $x_i$ (in this case, an image) is classified with a value $y_i \in \{-1, 1\}$ (which in this case responds to non-face/face). We also have a set of classifiers $k_1, k_2, \ldots k_T$, which when used on an item in our dataset produces a classification $k_j(x_i) \in \{-1, 1\}$. In order to achieve the strong classifier $C_T$, we add together all the weak classifiers with a weight for each one as follows:

$$C_T(x_i) = \alpha_1 k_1(x_i) + \alpha_T k_T(x_i)$$

At iteration number $m$ we have successfully included classifier number $m - 1$ and now seek to include the next one:
\[ C_m(x_i) = C_{m-1}(x_i) + \alpha_m k_m(x_i) \]

The problem is now to choose the best classifier for \( k_m \) and what the weight \( \alpha_m \) should be. Granted, expanding the strong classifier comes with a cost - a loss on some datapoints - which we seek to minimize. We define this cost, or error, \( E \) of \( C_m \) as the exponential loss on each data point:

\[ E = \sum_{i=1}^{N} e^{-y_i C_m(x_i)} \quad (1) \]

since we are interested in determining \( k_m \), we can rewrite the above formula as:

\[ E = \sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha_m k_m(x_i)} \]

in which

\[ w_i^{(m)} = e^{-y_i C_{m-1}(x_i)} \]

with this, we can split (1) into two sums:

\[ E = \sum_{y_i = k_m(x_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{y_i \neq k_m(x_i)} w_i^{(m)} e^{\alpha_m} \]

which essentially means that the cost of expanding the formula is the sum of the total weight of all hits and the total weight of all misses. In order to simplify, we denote the first summand as \( W_c e^{-\alpha_m} \) and the second as \( W_e e^{\alpha_m} \), which leads to

\[ E = W_c e^{-\alpha_m} + W_e e^{\alpha_m} \quad (2) \]

In order to now choose \( k_m \), the exact value of \( m \) is of little interest, as long as its greater than 0. This is because minimizing \( E \) with a fixed value of \( \alpha_m \) is the same thing as minimizing \( e^{\alpha_m} E \) and by rewriting (2) we obtain:

\[ e^{\alpha_m} E = (W_c + W_e) + W_e (e^{2\alpha_m} - 1) \]

We can now see that the cost of increasing the strong classifier consists of two parts; first, the total sum of the weights (which is not dependant on which weak classifier we choose). The second part of the equation is minimized when we choose the classifier with the lowest total cost \( W_c \), the lowest cost of the weighted error. In hindsight, this might be an obvious conclusion that the best candidate is the one with the lowest penalty given to our current strong classifier.
2.3.2 Weighting

The next task in improving our strong classifier is to choose what weight our new weak classifier should have. Deriving from (2) shows that

$$\frac{dE}{d\alpha_m} = -W_c e^{-\alpha_m} + W_e e^{\alpha_m}$$

Through equating the derivative to 0 and then multiplying with $e^{\alpha_m}$, we see that

$$0 = -W_c + W_e e^{2\alpha_m}$$

which simply leads us to the optimal weight:

$$\alpha_m = \frac{1}{2} \ln\left(\frac{W_c}{W_e}\right)$$
3 Method

3.1 The detection approaches

In regards to face detection, each approach can be categorized based on the range of acceptable head poses into one of the four following categories [9]:

- **Single Pose**: the head is assumed to be in a single, upright pose - either frontal or profile
- **Rotation invariant**: in-plane rotations of the head are allowed
- **Multi-view**: out-of-plane rotations are binned into a pre-determined set of views
- **Pose-invariant**: no restrictions on the orientation of the head

As this paper seeks to challenge and find the limits of the two systems, it intends to focus on the most real-life and general pictures - the pose-invariant ones.

One complication when comparing different face detection systems is the differences in desired output. To clarify, many systems agree upon marking an image region (rectangular regions or image patches of various shapes) for each hypothesised face and others instead use the location of apparent facial landmarks such as the eyes or the mouth. The research presented in this paper is limited to evaluating region-based output alone, as the two systems compared both use this method.

3.2 Face detection dataset

The images used in the study by Berg et al. [10] used a dataset of images extracted from news articles. This allowed for a large range of lighting, poses, background and appearance. Some of the variations in appearance are a result of motion, obstruction, facial expressions and focus - all both necessary and unavoidable when testing face detection in an uncontrolled environment.

Since Berg et al. amassed the images for their dataset from the Yahoo! news website, which collects news from several sources - most of these sharing the same photograph sources such as Reuters or the Associated Press, it contained several near-duplicated images (small differences in editing, cropping, etc.). As it would be uninteresting to see real-life value in comparing systems over near-duplicated images, the dataset was later improved upon by Jain and Learned-Miller in [9] by removing multiple of these copies. What resulted was a rich collection of images perfectly suited for comparing expected performance of face detection systems in unconstrained settings.
Though the original dataset used by Jain and Learned-Miller contained over 2800 images, the research performed in this paper limits itself to 203 pictures with a total of 389 faces. As we sought to challenge both systems to find their limits, images with the most variance in regards to lighting, poses, background and appearance were chosen.

### 3.3 Defining face regions

For images in real-life scenarios and in general unconstrained settings, it is often challenging to decide whether or not some regions actually contain a face or not. For example, backgrounds with lower resolution due to camera focus, occlusion of the face or strange head-poses may make a face vague and inconclusive.

A potential way to handle these issues is to establish some computable measurement regarding the region in question and automatically dismiss regions that do not fulfill the criteria. This is, however, impractical as it is problematic to define a required resolution, what fraction of the face must be included or at what angle the face should be looking at. Because of these limiting factors, the pictures used was instead
characterized using human judgement. All images were showed to each member of a group of different people in order to gather several independent decisions about each picture. These people were volunteers chosen at random in order to achieve as many different viewpoints as possible. A collective group discussion on the questionable pictures and regions was had, followed by a statistical analysis on the images with differing judgements.

In order to be as objective as possible as well as achieving consistency throughout multiple people, guidelines regarding what to consider a face or not was given to the group performing the judgement. The guidelines were inspired by the guidelines used in [9], where Jain and Learned-Miller gave instructions to their annotators about how to draw distinguishing face region ellipses. Moving forward from their guidelines, these instructions was given to the evaluators:

1. Reject a face if it is impossible to determine either its position, size or orientation.
2. A face region must contain at least three of the following: left eye, right eye, nose and mouth.
3. Neither of the eyes need to be open, and can be occluded by glasses.
4. The majority of the facial features (eyes, nose, mouth, etc) must be clearly visible.
5. Distinguishing between close facial features must be possible: eyes to eyebrows, mouth to chin, etc.

From these instructions some following constraints can be drawn. First of all, if neither of the eyes are visible, a region cannot contain a face. This also means that parts of the face can be obstructed and it should still count as a face. For example, faces in profile with ears occluded can very well count as a face.

Secondly, faces with low resolution due to camera focus and/or distance from camera will not be counted as a face. Since most real-life face detection scenarios will not aim to find these faces in the background or very far away, they should not be prioritized by the system and it would merely be a bonus in the case of a detection.

3.4 Result assessment

In order to create a base method for evaluating the performances, some assumptions about the output must first be made. The following assumptions were made prior to investigating the detections:
• A detection corresponds to a contiguous image region.

• Each detection corresponds to exactly one face - no more, no less. This means that no image region can contain more than one face and that a combination of multiple regions together cannot make up one face. In the case of multiple regions detecting the same face or parts of one face, only one should be counted as a positive detection and the rest be recognized as false positives.

Concluding that a detection is a placement of a rectangle in the input image, we need to make a decision regarding how accurate that subimage rectangle must be. When determining correct detections and false positives, we use the assumptions above together with the instructions given in section 3.3.

We derive from instruction number 2 that in order for a detection to be considered accurate and correct, it needs to contain at least three of the facial features mentioned (left eye, right eye, nose and mouth) - otherwise, it should count as a false positive.
4 Result presentation

The results from the face detection systems by Facebook and the open source library OpenCV are displayed below in table 5.1. From the dataset tested in this experiment True Positives (TP), False Negatives (FN) and False Positives (FP) are found after comparing with the established amount of faces in each image. As stated earlier in section 3.2, a total of 389 faces were to be found in the pictures.

<table>
<thead>
<tr>
<th>Face detection system</th>
<th>Facebook</th>
<th>OpenCV</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positives</td>
<td>366</td>
<td>319</td>
</tr>
<tr>
<td>False Negatives</td>
<td>23</td>
<td>70</td>
</tr>
<tr>
<td>False Positives</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Faces found percentage</td>
<td>94.087%</td>
<td>82.005%</td>
</tr>
</tbody>
</table>

Table 2: Face detection results

The data from table 2. is presented in figure 7.
Figure 7: Visualization of true positives, false negatives and false positives
From this, we can further calculate the recall (R) and the precision (P) of both systems. The formulas for these are:

\[ R = \frac{TP}{TP + FN} \]

\[ P = \frac{TP}{TP + FP} \]

Figure 8: Recall and precision
5 Discussion

The result in Figure 7 show that Facebooks face detection algorithm is significantly better at finding correct faces and did not find any false positives for our dataset whatsoever. This compared to OpenCV which had 17 false positives shows the difference between these two. However, OpenCV did perform quite well with 82,005% found faces and it did even outperform Facebook on a couple of images (see section 5.2). This taken into consideration Facebooks score of 94,087% is considerably better and it is most likely due to the fact that they have an enormous database of images they can use to calibrate their algorithm with. The haar-cascade used in OpenCV is trained, yet it can not possibly be as extensively trained as the one Facebook possesses.

The most impressive finding from these results is probably Facebooks ability to not find any false positives. With the variation in images with faces being blurry, sharp, hand-drawn and obscured it was surprising that Facebooks algorithm was so generally refined. This is the end result of millions of people uploading photos and tagging the faces of people in these pictures. From the beginning the tagging (and by that, the face detection) was entirely made by hand and you can still tag the photos were the algorithm was unable to detect a face. This constantly feeds Facebook more data they can use to improve their software. This is simply an amount of data that open source system can not achieve.

With all things considering and with the results given, OpenCV is still a very well performing face detection system that is perfect for private use when a face detection application is required. It is also possible to adjust and customize the algorithm in OpenCV to work better for some pictures in order to to find more faces, however this is far more complicated. The OpenCV algorithm has thus room for improvement - however, the goal of the research performed in this study was to test the most basic configuration of the detection algorithm.

5.1 Limitations

The main limitation this paper performed was the number of images tested. As with all statistics, any conclusion that can be reached will undoubtedly be reliant on the number of results - with more results and more numbers, a more convincing conclusion can be drawn. This means that, while a 0% rate in false positives - as with Facebook - is impressive, the question of whether or not the reason behind this was the input data remains ambiguous.

Another limiting factor was the pictures chosen. Since the every picture
were chosen based on its potential to challenge the system limitations, it is possible that OpenCV will outperform Facebook for less demanding data. One could argue that the rate of true positives would rise, while the number of false negatives would most certainly drop with less difficult data.

The research performed in this paper also limits itself to the actual detection results. While this is the main reason one turns to face detection systems, other benchmarks such as memory allocation and time complexity could very well be a deciding factor. Testing these would be simple for OpenCV, as measuring instruments are available and can be implemented, while it would be a difficult test for Facebook - unless they were to unveil their own implementation.

It remains, however, that with a relatively small number of tests, it is uncertain whether or not the results are statistically applicable to other areas of research and practice. Further research would include a larger data set with images of varying difficulty, together with further benchmarking memory allocation and time complexity of both algorithms.

5.2 Deviations and special cases

Some special cases not related to the statistical analysis of the results need mentioning. Despite the fact that Facebook did outperform OpenCV for the majority of the pictures, this was not the case for all of them. On 5 of the images tested, OpenCV did find more faces than Facebook did. These were mostly difficult pictures with faces in low resolution, which spawns an interesting question - could OpenCV be more efficient at detecting faces in low resolution than Facebook? As the statistical results for such pictures were neither conclusive nor very well measured, the question is left for future research.
References


