Tel Aviv University

The Raymond and Beverly Sackler Faculty of Exact Sciences

The Blavatnik School of Computer Science

Face Recognition in Unconstrained Videos with Matched Background Similarity

This thesis is submitted as partial fulfillment of the requirements towards the M.Sc. degree in the School of Computer Science, Tel-Aviv University

by

Itay Maoz

The research work for this thesis has been carried out at Tel-Aviv University under the supervision of Prof. Lior Wolf

January 2012
Contents

1 Introduction 2

2 Existing Benchmarks 5
   2.1 Existing face recognition in still images benchmark databases . . . . . . 5
   2.2 Existing face recognition in videos benchmark databases . . . . . . 6

3 Existing Methods For Detecting And Encoding Face Images 8
   3.1 Face Detection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
   3.2 Face Image Encoding . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
      3.2.1 LBP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
      3.2.2 CSLBP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
      3.2.3 FPLBP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

4 Existing Methods For Face Recognition In Videos 14
   4.1 Previous Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
   4.2 Existing Methods For Comparing Sets . . . . . . . . . . . . . . . . . . 15
      4.2.1 All pairs comparisons . . . . . . . . . . . . . . . . . . . . . . . . 15
      4.2.2 Pose based methods . . . . . . . . . . . . . . . . . . . . . . . . . 16
      4.2.3 Algebraic methods . . . . . . . . . . . . . . . . . . . . . . . . . . 16
      4.2.4 Non-algebraic Set methods . . . . . . . . . . . . . . . . . . . . . 17

5 Existing Methods, The One Shot Similarity 18
   5.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
   5.2 The OSS score . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

6 The Matched Background Similarity (MBGS) 20

7 The ‘Youtube Faces’ set and benchmark 23
   7.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
   7.2 Collection setup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
   7.3 Database encodings . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
   7.4 Benchmark tests . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

8 Experiments 26
   8.1 All pairs comparisons . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
   8.2 Pose based methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
   8.3 Algebraic methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
   8.4 Non-algebraic Set methods . . . . . . . . . . . . . . . . . . . . . . . . 27
   8.5 Matched Background Similarity . . . . . . . . . . . . . . . . . . . . . . 27
   8.6 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

9 Discussion and Future Work 32

Appendices 33
Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor Prof. Lior Wolf from the Blavatnik School of Computer Science at Tel Aviv University, for his continuous support during the research and for his patience, motivation, availability, creativity and immense knowledge.

Not only that I greatly learned from his vast knowledge, he also increased my enthusiasm and curiosity and pushed me towards the edge of the unknown in the fascinating and exciting world of Computer Vision.

I would also like to thank Dr. Tal Hassner from the Department of Mathematics and Computer Science at The Open University of Israel, for his guidance and support, for his innovative ideas, and great knowledge.

I would like to thank my family for supporting me on this long journey and for their patience.
To my parents, for inspiring me and being a role model as M.Sc and Ph.D graduates.
To my kids, Barak and Mika, who gave up their father many times, while being too young to understand why.
And a very special thanks to my wife Hila, for enabling me to work on my research while taking care of everything in the background, and for her constant moral support.

Thank you all.
Abstract

Recognizing faces in unconstrained videos is a task of mounting importance. While obviously related to face recognition in still images, it has its own unique characteristics and algorithmic requirements. Over the years several methods have been suggested for this problem, and a few benchmark data sets have been assembled to facilitate its study.

However, there is a sizable gap between the actual application needs and the current state of the art. In this work we make the following contributions:
(a) We present a comprehensive database of labeled videos of faces in challenging, uncontrolled conditions (i.e., ‘in the wild’), the ‘YouTube Faces’ database, along with benchmark pair-matching tests. The database, image encoding, benchmark tests, paper results and the code of the baseline methods are available at www.cs.tau.ac.il/~wolf/ytfaces.
(b) We employ our benchmark to survey and compare the performance of a large variety of existing video face recognition techniques.
(c) We describe a novel set-to-set similarity measure, the Matched Background Similarity (MBGS). This similarity is shown to considerably improve performance on the benchmark tests.

This thesis is based on the paper “Face Recognition in Unconstrained Videos with Matched Background Similarity” by Lior Wolf, Tal Hassner and Itay Maoz [53].
Chapter 1

Introduction

On-line video repositories such as YouTube are expanding at a staggering pace. For example, in YouTube more than 60 minutes of video are uploaded every second. Searching through these collections is typically performed using text queries, often with names of people appearing in the desired videos. Of course, locating the videos of a particular person may be done by searching through user contributed meta-data, labels and tags. This information, however, is notoriously unreliable [36, 59]. Some examples are different people with the same name, wrong tagging or labeling, typos, user mistakes, etc.

Automatically analyzing videos to determine the identities of the people appearing in them, on the other hand, requires accurate face recognition techniques, designed to utilize information from multiple frames and remain robust in unconstrained videos.

Although face recognition is one of the most well studied problems in Computer Vision, recognizing faces in on-line videos is a field very much in its infancy. Videos naturally provide far more information than single images [27]. Indeed, several existing methods have obtained impressive recognition performances by exploiting the simple fact that a single face may appear in a video in many consecutive frames (e.g., [13, 32]). These methods, however, were primarily developed and tested using either strictly controlled footage or high quality videos from motion-pictures and TV shows. People appearing in these videos are often collaborative, are shot under controlled lighting and viewing conditions, and the videos themselves are stored in high quality.

Videos found in on-line repositories such as YouTube are very different in nature. Many of these videos are produced by amateurs, typically under poor lighting conditions, difficult poses, and are often corrupted by motion blur. In addition, bandwidth and storage limitations may result in compression artifacts, making video analysis even harder. In strictly controlled videos, the videos are being shot throughout a short period of time, where the face of the subject doesn’t change. In unconstrained videos, people are shot in different times of their life which causes large variability in age, wrinkles, haircut, with/without glasses/mustache/beard/hat/etc. This makes recognition much harder.

Recently, the introduction of comprehensive databases and benchmarks of face images, in particular images ‘in the wild’ (e.g., [19]), has had a great impact on the development of face recognition techniques. The LFW [19], for instance, contains 5,749 different subjects collected from the web, in uncontrolled challenging conditions; it is used as a standard benchmark for face recognition solutions, pushing forward the
technology for better results every year (see, for example, [24]).

In light of this success, we present a large-scale, database, the ‘YouTube Faces’ database, and accompanying benchmark for recognizing faces in challenging, unconstrained videos (e.g., Fig. 1.1). Following [19] our benchmark is a simple, yet effective, large scale pair-matching benchmark, allowing for standard testing of similarity and recognition methods.

We also provide a framework code, for easily testing new techniques on this benchmark, enabling the researchers to focus on the face recognition algorithm without worrying about the database or the benchmark. We use this benchmark to survey and test existing state-of-the-art techniques for face recognition in videos.

These techniques are sub-grouped to the following categories (see Chapter 4.2 for the details):

- All pairs comparisons.
- Pose based methods.
- Algebraic methods.
- Non-algebraic set methods.

Figure 1.1: Example frames from the spectrum of videos available in the YouTube Faces data set. The two bottom rows depict some of the challenges of this set, including amateur photography, occlusions, problematic lighting, pose, and motion blur.
We further present a novel set-to-set similarity measure, MBGS - Matched Background Similarity (see Chapter 6), used here to evaluate the similarity of face videos. This similarity is designed to utilize information from multiple frames while remaining robust to pose, lighting conditions and other misleading cues. We consequently demonstrate a significant boost in accuracy over existing methods.
Chapter 2

Existing Benchmarks

2.1 Existing face recognition in still images benchmark databases

Face recognition is one of the oldest and most well studied problems in Computer Vision and Biometrics. As such, the literature on this problem is vast. The progress made in this field is due, in no small part, to the availability of well designed image collections and benchmarks. Over the years these collections have progressed from small scale image sets acquired under controlled laboratory conditions to huge collections of unconstrained images. Some recent examples include:

- Facial Recognition Technology (FERET) Database [30]. Its primary mission was to develop automatic face recognition capabilities that could be employed to assist security, intelligence and law enforcement personnel in the performance of their duties. It consists of 14051 grayscale images of human heads with views ranging from frontal to left and right profiles, for 10465 different subjects.

- FERET’s successor, the Face Recognition Vendor Test (FRVT) database [31], with 37,437 images of human heads.

- FRVT’s successor, the Face Recognition Grand Challenge (FRGC) images and benchmarks [28, 29], which already consisted of 4000 subjects, and 50,000 recordings.

- CMU Pose Illumination and Expression (CMU-PIE) database [39], which contains 41,368 images of 68 people. Each person was imaged under 13 different poses, 43 different illumination conditions, and with 4 different expressions.

- CMU-PIE’s extension, the multi-PIE collection [16], which contains over 750,000 images of 337 subjects imaged under 15 view points and 19 illumination conditions in up to four recording sessions.

See a summary table for these databases at Table 2.1. These image sets were all designed to capture different sources of variability likely to be encountered by face recognition systems. These include illumination, pose, expression and more. However, these image sets databases were all produced in laboratory controlled settings. To provide researchers with a wider and more arbitrary range of viewing conditions, such as those available on-line [41], the Labeled Faces in the Wild (LFW) [19] and its extension, the Public Figure Face Database (PubFig) [21] sets were assembled. These image sets include over 13,000 images of over 5700 people, automatically collected from
<table>
<thead>
<tr>
<th>Database Name</th>
<th>Number of Subjects</th>
<th>Number of Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>FERET</td>
<td>10,465</td>
<td>14,051</td>
</tr>
<tr>
<td>FRVT</td>
<td>37,437</td>
<td>37,437</td>
</tr>
<tr>
<td>FRGC</td>
<td>4,000</td>
<td>50,000</td>
</tr>
<tr>
<td>CMU-PIE</td>
<td>68</td>
<td>41,368</td>
</tr>
<tr>
<td>multi-PIE</td>
<td>337</td>
<td>750,000</td>
</tr>
<tr>
<td>LFW</td>
<td>5,749</td>
<td>13,000</td>
</tr>
<tr>
<td>PubFig</td>
<td>200</td>
<td>58,797</td>
</tr>
</tbody>
</table>

Table 2.1: Existing still images benchmark databases.

internet web pages; images were added to these sets if they include a face detected by a Viola and Jones face detector [44]. The facial images included in the LFW data set therefore demonstrate quite a bit of variability. Since its recent publication, a lot of attention has been focused on improving performance on the benchmarks associated with the LFW database (see, [24]).

2.2 Existing face recognition in videos benchmark databases

Below are few examples of datasets designed to promote face recognition from video:

1. MPI face video database for biological cybernetics [5].
   This database contains 246 different videos of one subject, shot in a lab with six different view point for each video. This database contains videos of facial action units. Their Video lab technology enabled them to record facial movements from six different viewpoints at the same time while maintaining a very precise synchronization between the different cameras.
   This database is suited for classifying different facial action units and not for identity face recognition.

2. Automatic Naming of Characters in TV Video [12].
   The objective of this work is to label television or movie footage with the names of the people present in each frame of the video.
   The data is from episodes of Season 5 of the TV series 'Buffy the Vampire Slayer'. It contains both frontal and profile faces, with large variation due to changes in scale, pose, lighting, expressions, hair style etc., and some shots have also additional problems of poor image quality and motion blur.
   This database by its nature is limited to few different identities, and shot in a controlled environment.

3. The VidTIMIT Audio-Video Dataset [9, 35].
   The VidTIMIT dataset is comprised of video and corresponding audio recordings of 43 people, reciting short sentences. There are 10 sentences (videos) per person, all shot under controlled lab conditions.
   This database can be useful for research on topics such as automatic lip reading, multi-view face recognition, multi-modal speech recognition and person identification. Nevertheless it is quite small and contains very few identities (43), all shot under controlled environment.

4. A Video Database of Moving Faces and People [2].
   This is a database of static images and video clips of human faces and people that is useful for testing algorithms for face and person recognition, head/eye tracking, and computer graphics modeling of natural human motions. It contains
284 different subjects, each has 6 videos and still images. All the videos were shot in a lab, in a controlled environment.

5. The XM2VTSDB database [8].
The XM2VTSDB contains four recordings of 295 subjects taken over a period of four months. Each recording contains a speaking head shot and a rotating head shot. This database is not free of cost, and shot in a lab, in a controlled environment.

6. The Honda/UCSD Video Database [23, 22].
The goal of the Honda/UCSD Video Database is to provide a standard video database for evaluating face tracking/recognition algorithms. Each video sequence was recorded in an indoor environment at 15 frames per second, and each lasted for at least 15 seconds. There are total of 70 videos. All the videos were shot in a lab, in a controlled environment.

7. NRC-IIT Facial Video Database [14].
This video-based face database has been created in order to provide the performance evaluation criteria for the techniques developed and to be developed for face recognition in video. It contains 11 subjects, for each subject there is a pair of short video clips each showing a face of a computer user sitting in front of the monitor. The videos were all shot in the same environment, in front of a computer monitor in an indoor closed lab.

See a summary table for these databases at Table 2.2. However, contrary to face recognition in images, as far as we know, for face recognition in videos there is no widely used, large scale video data collection available for learning and testing recognition algorithms in uncontrolled, ‘wild’ conditions. All of the above face videos databases are relatively small, and shot in a controlled environment. We bridge this gap by publishing the YouTube Faces database described in this work in Chapter 7.

<table>
<thead>
<tr>
<th>Database Name</th>
<th># Subjects</th>
<th># Videos</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI [5]</td>
<td>1</td>
<td>246</td>
<td>A single person in a very controlled environment</td>
</tr>
<tr>
<td>Naming of Characters in TV Video [12]</td>
<td>14</td>
<td>27,504 frames</td>
<td>Different frames of the same TV show shot in a studio</td>
</tr>
<tr>
<td>VidTIMIT Audio-Video Dataset [9, 35]</td>
<td>43</td>
<td>430</td>
<td>Small, and shot under controlled environment</td>
</tr>
<tr>
<td>Video Database of Moving Faces and People [2]</td>
<td>284</td>
<td>1704</td>
<td>Shot in a lab, under controlled environment</td>
</tr>
<tr>
<td>XM2VTSDB database [8]</td>
<td>295</td>
<td>510</td>
<td>Not free, small, and shot under controlled environment</td>
</tr>
<tr>
<td>The Honda/UCSD Video Database [23, 22]</td>
<td>70</td>
<td>22</td>
<td>Small, and shot under controlled environment</td>
</tr>
<tr>
<td>NRC-IIT Facial Video Database [14]</td>
<td>11</td>
<td>22</td>
<td>Small, and shot under controlled environment</td>
</tr>
</tbody>
</table>

Table 2.2: Existing face videos benchmark databases.
Chapter 3

Existing Methods For Detecting And Encoding Face Images

Frames of a video showing the same face, are often represented as sets of vectors, one vector per frame. Representing images as vectors of descriptors is very common in many fields of computer vision, and in particular in face images. The common method is detecting a face in the image by using a face detector such as the Viola & Jones Face Detector [44], and then encoding the descriptors based only on the pixels inside the face’s bounding box.

3.1 Face Detection

The most commonly used face detector algorithm in the past few years has been the Viola & Jones face detection algorithm [44]. This is also the face detection algorithm used in OpenCV [26] (with over 2.5 million downloads), and the face detection algorithm we used in our work. It has been proven to be very fast and efficient.

In the high level, the Viola Jones face detector has two main steps:

1. Training.
   A classifier is trained using a variant of AdaBoost with a large set of positive and negative examples. The positive examples are images of faces, and the negative examples are non-faces images. For each image the features are extracted, and the most discriminating features are selected.

2. Classifying.
   On a given image, the algorithm runs on a sliding window over the entire image, with different scales, and on each window classifies if it has a face in it or not.

The Viola Jones Features

The Viola Jones features are simple and contain 4 different types of rectangle features, as explained in Fig. 3.1. The rectangle features can be computed very rapidly using an intermediate representation for the image which is called the integral image.

The integral image at location of \((x, y)\) contains the sum of the pixels above and to the left of \((x, y)\), inclusive as described in Fig. 3.2. Computing the integral image requires a single pass, \(O(height \times width)\), and then
Figure 3.1: The four Viola Jones rectangle features: A, B, C and D. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles.

extracting the 4 different features for a given square can be done in $O(1)$.

**The Viola Jones Learning Algorithm**

Within any image sub-window the total number of features is very large, far larger than the number of pixels. In order to ensure fast classification, the learning process must exclude a large majority of the available features, and focus on a small set of critical features. Feature selection is achieved through a simple modification of the AdaBoost procedure: the weak learner is constrained so that each weak classifier returned can depend on only a single feature. As a result each stage of the boosting process, which selects a new weak classifier, can be viewed as a feature selection process.

**The Viola Jones Attentional Cascading**

The classifiers are arranged in a cascading order which achieves increased detection performance while radically reducing computation time. Smaller (and hence faster) boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. Therefore simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates.

On the ROC curve, the first classifiers are very close to 100% detection with high false positive rate, and the next classifiers have each lower and lower false positive rates.

The overall form of the detection process is that of a cascading decision tree (see Fig. 3.3):

- A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates.
- A positive result from the second classifier triggers a third classifier, and so on.
- A negative outcome at any point leads to the immediate rejection of the sub-window.
Figure 3.2: The Viola Jones integral image at location \((x, y)\) contains the sum of the pixels above and to the left of \((x, y)\), inclusive.

![Viola Jones integral image](image)

Figure 3.3: The Viola Jones cascading architecture.

### 3.2 Face Image Encoding

Over the years several algorithms have been developed for the purpose of encoding face images. These algorithms run on the bounding box found by the face detector, and extract descriptors stored in features’ vectors. In the YouTube Faces Database we provided (see chapter 7), we used three types of image encoding descriptors:

1. **Local Binary Patterns (LBP)** encoding [1].
2. **Center-Symmetric Local Binary Patterns (CSLBP)** [18].
3. **Four-Patch Local Binary Patterns (FPLBP)** [54].

#### 3.2.1 LBP

In 2006 the Local Binary Patterns (LBP) encoding [1] became widely used for face recognition. The LBP descriptors have been proven to be very effective for face recognition.

The basic LBP operator is described in Fig. 3.4. The operator assigns a label to every pixel of an image by thresholding the 3x3-neighborhood of each pixel with the center pixel value and considering the result as a binary number. For a given pixel \(p\), check its surrounding 8 pixels \(p_1, p_2, \ldots, p_8\), starting from the pixel left to \(p\) and then counter clockwise. Then the examined pixel gets its value by a binary string \(b_1b_2\ldots b_8\), where:
Figure 3.4: Basic LBP operator.

\[ b_i = \begin{cases} 
1 & \text{if } p_i \geq p \\
0 & \text{otherwise}
\end{cases} \]

The binary string is then converted to a decimal value. More formally:

\[ LBP_{R,N}(p) = \sum_{i=0}^{N-1} b_i 2^i \]

The histogram of the labels is then used as a texture descriptor.

To be able to deal with textures at different scales, the LBP operator was extended to use neighborhoods of different sizes. Defining the local neighborhood as a set of sampling points evenly spaced on a circle centered at the pixel to be labeled allows any radius and number of sampling points. Bilinear interpolation is used when a sampling point does not fall in the center of a pixel. The notation \((P,R)\) means \(P\) sampling points on a circle of radius \(R\).

The basic LBP operator was farther extended by using the uniform patterns. A local binary pattern is called uniform if the binary pattern contains at most two bitwise transitions from 0 to 1 or vice versa. For example, the patterns 00000000 (0 transitions) and 01110000 (2 transitions) are uniform whereas the pattern 11001001 (4 transitions) is not.

In the computation of the LBP histogram, uniform patterns, which are typically almost 90% of the patterns, are used so that the histogram has a separate bin for every uniform pattern and all non-uniform patterns (which are a bit over 10% of the patterns) are assigned to a single bin.

When computing the LBP descriptor for a face image (inside the bounding box found by the face detector), several local descriptions of the face are used and then combined into a global description. For face descriptors retaining the information about spatial relations is important. Therefore the methodology is to divide the facial image into local rectangular regions (for instance 7x7, see figure 3.5), and then the above LBP histogram is computed for each region separately. For \(m\) facial regions, \(R_0, R_1, ..., R_m\), a histogram is computed independently within each of the \(m\) regions. The resulting \(m\) histograms are combined yielding the spatially enhanced histogram with size \(m \times n\) where \(n\) is the length of a single LBP histogram.

### 3.2.2 CSLBP

The Center-Symmetric Local Binary Patterns (CSLBP) [18], is a variant of LBP. The basic LBP operator is different and calculated as follows.

Using the same notation as in the LBP operator, for a given pixel \(p\) and its surrounding 8 pixels \(p_1, p_2, ..., p_8\), the examined pixel gets its value by a binary string \(b_1b_2b_3b_4\) where:

\[ b_1 = s(p_1 - p_5) \]
Figure 3.5: LBP: A facial image divided into 7x7, 5x5 and 3x3 rectangular regions.

\[ b_2 = s(p_2 - p_6) \]
\[ b_3 = s(p_3 - p_7) \]
\[ b_4 = s(p_4 - p_8) \]

\[ s(x) = \begin{cases} 
1 & \text{if } x \geq T \\
0 & \text{otherwise} 
\end{cases} \]

where \( T \) is some very small threshold, e.g. \( T = 0.01 \). The binary string is then converted to a decimal value like in LBP. More formally:

\[ \text{CSLBP}_{R,N,T}(p) = \sum_{i=0}^{N/2-1} s(p_i - p_{i+(N/2)})2^i \]

The histogram of the labels is then used as a texture descriptor same as in the LBP encoding described previously.

The main advantage of CSLBP over LBP is that in its core it compares only center-symmetric pairs of pixels, which means much smaller histograms. For example, for 8 neighbors it uses a 4 bit label for each pixel instead of 8 bit label produced by the LBP encoding. This produces significantly smaller amount of different binary patterns, resulting much smaller histograms, and hence much smaller descriptors.

### 3.2.3 FPLBP

The Four-Patch LBP (FPLBP) \[54\] is yet another variant of LBP. At its core, it differs from LBP and CSLBP by the fact that it compares patches with size \( w \times w \) instead of single pixels. For a given pixel \( p \), the FPLBP looks at two rings (instead of one ring in LBP and CSLBP) with different radii \( r_1 \) and \( r_2 \) centered on the pixel, and \( S \) \( w \times w \) patches spread evenly on each ring (see Fig. 3.6).

The FPLBP compares two center symmetric patches in the inner ring with two center symmetric patches in the outer ring positioned \( \alpha \) patches away along the circle. Each bit in FPLBP encoding for a given pixel is set according to which of the two pairs being compared is more similar.

More formally:

Given two patches \( C_1, C_2 \), and some distance function \( d(C_1, C_2) \) between the two patches (e.g. \( L_2 \) norm of their gray level differences), and we defined \( f(x) \) similar as in CSLBP:

\[ f(x) = \begin{cases} 
1 & \text{if } x \geq T \\
0 & \text{otherwise} 
\end{cases} \]
Figure 3.6: FPLBP: Four patches involved in computing a single bit value. The FPLBP code computed with parameters $S = 8$, $w = 3$, and $\alpha = 1$.

where $T$ is some very small threshold, e.g. $T = 0.01$.

Then FPLBP is defined as:

$$FPLBP_{r_1, r_2, S, w, \alpha}(p) = \sum_{i=0}^{S/2} f(d(C_{1,i}, C_{2,(i+\alpha)modS}) - d(C_{1,i+S/2}, C_{2,(i+S/2+\alpha)modS}))2^i$$

The histogram of the labels is then used as a texture descriptor same as in the LBP encoding described previously.
Chapter 4

Existing Methods For Face Recognition In Videos

4.1 Previous Work

As opposed to face recognition in still images, research of face recognition in video is a younger field. Some early work in video face recognition includes:

1. ‘Unsupervised face recognition from image sequences based on clustering with attraction and repulsion’ [33] which use manifolds to represent the time varying appearances of faces.

2. ‘A unified learning framework for real time face detection and classification’ [38] who focus on real-time face recognition in videos.

3. ‘Taking the bite out of automated naming of characters in TV video’ [13] and ‘Learning person specific classifiers from video’ [40] focus on the task of aligning subtitle information with faces appearing in TV shows.

4. In ‘Leveraging archival video for building face datasets’ [32] faces appearing in a TV show were clustered according to subject identity across 11 years of broadcast.

5. ‘Face recognition from long-term observations’ [37] and ‘Attribute-based people search in surveillance environments’ [43] focus on surveillance videos, where searching through people in surveillance videos is related to recognition from web videos. Surveillance videos often have poor quality and the subjects viewed are uncooperative.
   Video quality as well as viewing directions, however, typically remain constant, whereas web-videos vary greatly in these parameters.

6. ‘On person authentication by fusing visual and thermal face biometrics’ [3], ‘Video biometrics’ [6] and ‘Biometric Person Recognition: Face, Speech and Fusion’ [34] are about using face recognition in videos for biometrics and access control.

7. Finally, web-videos have begun to attract attention in the last couple of years. Both ‘Audiovisual celebrity recognition in unconstrained web videos’ [36] and ‘Large scale learning and recognition of faces in web videos’ [60] are exploring large-scale web-video collections for face recognition. They focus on recognition using unreliable, user-contributed, text labels [60] and audio-visual cues [36].
   Similar to the benchmark proposed here, the authors of ‘Boosted multi-task
learning for face verification with applications to web image and video search’ [49] briefly touch on the problem of face-verification from web-videos.

4.2 Existing Methods For Comparing Sets

As explained in Chapter 3, frames of a video showing the same face, are represented as sets of vectors, one vector per frame. Thus, recognition becomes a problem of determining the similarity between vector sets, which can be modeled as distributions [37], subspaces [55, 57], or more general manifolds [20, 33, 47]. Different choices of similarity measures are then used to compare two sets [47, 48, 55].

In this work we have tested several baseline face recognition methods on the YouTube Faces database we provided. Several types of methods were considered:

- All pairs comparisons. This group consists of methods employing comparisons between pairs of face images taken from the two videos.
- Pose based methods. Methods using the heuristic of detecting an approximately frontal pose in each frame sequence and comparing between them, or comparing the two frames with the most similar pose.
- Algebraic methods. This group consists of methods such as distances between projections, or comparing the principle angles between the two subspaces spanned by the videos’ vectors.
- Non-algebraic Set methods. This group includes the methods Pyramid Match Kernel and the Locality-constrained Linear Coding.

Below there is a detailed description on these methods.

The notation will be the following:
Each video is represented by a set of vectors, each one produced by encoding the video frames using one of a number of existing face descriptors (see Chapter 3).

When comparing the similarity of two videos, $X_1$ will represent the matrix whose columns are the encoding of the frames of the first video, and $X_2$ will represent the corresponding matrix for the other video.

4.2.1 All pairs comparisons

We compute a distance matrix $D$ where $D_{ij} = ||X_1(:,i) - X_2(:,j)||$, $X_1(:,i)$ denotes the i-th column of matrix $X_1$.

Four basic similarity measures are then computed using $D$:

1. The minimum distance of $D$.
2. The average distance of $D$.
3. The median distance of $D$.
4. The maximal distance of $D$.

In addition we also compute the ‘meanmin’ similarity in which for each image (of either set) we match the most similar image from the other set and consider the average of the distances between the matched pairs.
4.2.2 Pose based methods

1. Most frontal pose.

Presumably, the easiest image to recognize in each image set is the one showing the face in a frontal pose. We therefore locate the most frontal pose in each sequence by using the web API of www.face.com to obtain the three rotation angles of the head. Comparing two sequences then involves measuring the similarity between the descriptors of the representative frames of the two videos. The similarity measure in this case is

\[ ||X_1(:, \text{MostFrontalPose}_1) - X_2(:, \text{MostFrontalPose}_2)|| \].

In an earlier stage we approached this problem differently. We trained a classifier with 200 frontal images and 200 profile images. These images were manually labeled as ‘frontal’ or ‘profile’, and then we had a classifier for each descriptor type. For each video (frames sequence) the classifier returned the most frontal image (most confident classification), and this was the image used for comparing the similarity between the videos.

This approach gave similar results as using the API of www.face.com, but since we used the same pose descriptors (the three rotation angles of the head) for the ‘Similar pose’ method (see below) and later for MBGS (see Chapter 6), we decided to use it for the ‘most frontal’ method as well.

2. Similar pose.

This method uses one face image from each sequence by considering pairs of images with the smallest head rotation angle between them. Rotation is estimated for each image using the above API. Then all frames from one sequence are compared to all frames from the other sequence, finding the two with the most similar pose. The similarity measure in this case is:

Let \((i_1, i_2)\) be the indexes of the frames with the two most similar poses in \(X_1\) and \(X_2\) respectively, then the similarity is \(||X_1(:, i_1) - X_2(:, i_2)||\).

4.2.3 Algebraic methods

Algebraic methods view each matrix \(X_1\) or \(X_2\) as a linear subspace that is spanned by the columns of the matrix. A recent work [48] provides an accessible summary of large number of such methods. Many of the methods are based on the analysis of the principle angles between the two subspaces.

Let \(U_i, i = 1, 2\) be any orthogonal basis for the subspace spanned by the columns of \(X_i\). We can find these orthogonal matrices by using the singular value decomposition (SVD) of \(X_1\) and \(X_2\), such that \(X_i = U_i S_i V_i^\top\).

The SVD of \(U_1^\top U_2 = W S V_1^\top V_2^\top\) provides the principle angles between the column subspaces of the two matrices \(X_1\) and \(X_2\). Specifically, the inverse cosine of the diagonal of \(S\) are the principle angles, i.e., \(S = \text{diag}(\cos \Theta)\), where \(\Theta\) is the vector of principle angles of \(X_1\) and \(X_2\). Note that this vector is sorted from the least angle to the largest.

* In practice we used the eigenvalues of \(U_1^\top U_2\).

Several distances are defined based on these notations:

- The max correlation of [58] is defined by the minimal angle \(\theta(1)\).
- The projection metric [10] is given by \(||U_1 U_1^\top - U_2 U_2^\top||_F\).
- The norm \(||U_1^\top U_2||_F\) which seems to be relatively effective.
• The Procrustes metric \cite{7} is computed from the vector-norm $||\sin(\Theta/2)||$ (the \sin of a vector is taken element by element). In practice we used the norm of the eigenvalues vector.

Care should be taken when the number of frames differs between the sequences or if the number of samples is larger than the dimensionality of the representation. It is a good practice to restrict $U_1$ and $U_2$ to be the first $r$ singular vectors of the subspace spanned by the columns of $X_k$. This is justified by the fact that the projections $U_kU_k^\top$, $k = 1, 2$ provide the closest possible projection by a rank $r$ projection to the vectors of $X_k$.

In our experiments we found the value of $r = 10$ to provide relatively good performance. Using a larger constant did not increase performance, but just increased the calculation time.

The last algebraic method we compare to is the CMSM method \cite{58}. This method utilizes a training set and is essentially a max correlation method after the vectors have been projected to the subspace spanned by the smallest eigenvectors of the matrix that is the sum of all projection matrices of the training set. The projection is followed by a normalization step and an orthogonalization step. Lastly, the max correlation, sometimes called MSM, is computed as the score. Alternatively, as done in the code made available by the authors \cite{58}, the average of the largest $t = 10$ canonical correlations can be used.

4.2.4 Non-algebraic Set methods

We next consider methods that have emerged as effective ways to represents sets, not necessarily in the context of computer vision. The Pyramid Match Kernel (PMK) \cite{15} is an effective kernel for encoding similarities between sets of vectors. PMK represents each set of vectors as a hierarchical structure (‘pyramid’) that captures the histogram of the vectors at various levels of coarseness. The cells of the histograms are constructed by employing hierarchical clustering to the data, and the similarity between histograms is captured by histogram intersection.

In our experiments, we first construct the bins by hierarchical clustering on some representative videos, for each type of descriptor. Then when we had our vocabulary tree for this kind of descriptor, we used the vocabulary tree to create a pyramid for each face vector. The last step was to compute the kernel values from the pyramid.

We also test sparsity based methods, and specifically methods based on locality constrained encoding. In such methods, a face image is represented as a linear combination of a small subset of faces taken from a large dictionary of face images. Sparse representation methods were shown to enable accurate recognition despite of large variations in illumination and even occlusions \cite{56}.

As far as we know, such methods were not previously used for multi-frame identification. However, similar methods have been used for visual recognition based on sets of visual descriptors. The emerging technique, which we adopt in our experiments, is to represent the set by the maximal coefficients (one coefficient per dictionary vector) over all set elements. In order to maintain a reasonable run-time, in our experiments we employ the Locality-constrained Linear Coding (LLC) method \cite{46}, in which the sparse coefficients are computed based on the k-nearest dictionary vectors to each set element.

In practice we take descriptors from the non testing split in our database, and used them as the dictionary. We used the 5 nearest neighbors for this. Then the similarity was the $||maxCoefficientForX_1 - maxCoefficientForX_2||$. 

17
Chapter 5

Existing Methods, The One Shot Similarity

5.1 Motivation

The MBGS approach described in this work differs in that it models a set by a combination of a classifier and the set itself. At its core, the similarity is asymmetric and uses the classifier of one set to determine whether the vector set of another set is more similar to the first set or to a preselected subset background set. It is thus related to a recent family of similarities [42, 51, 52] based on a background set of examples and which employ classifiers.

The first recent background similarity method to emerge is the One-Shot-Similarity (OSS) [54, 51]. The problem it was designed to solve is the following: Given two vectors $x_1$ and $x_2$, for example two vectors of face descriptors, do they represent the same subject, or not the same subject?

This is the main benchmark for the LFW DB [19], where one needs to classify whether a pair of face images belong to the same subject or to different subjects. As explained in Section 7.4, our challenge is:

5.2 The OSS score

- Given two vectors $x_1, x_2 \in \mathbb{R}^d$, and a background sample set $B = \{b_1, \ldots, b_n\}$, $b_i \in \mathbb{R}^d$. This set of vectors, $B$, contains examples of descriptors with different identity from both $x_1$ and $x_2$. This background set is usually easy to assemble, by considering unlabeled examples, or in a designated benchmark by simply using some of the training set.

- First, a discriminative model is learned with $x_1$ as a single positive example and $B$ as a set of negative examples. This can be done by any classifier, such as SVM or LDA [51]. This discriminative classifier is trained to distinct between $x_1$ and a background which does not contain $x_1$.

So given another vector $x_t$, the classifier’s confidence level indicates whether $x_t$ is closer to $x_1$ than the background and therefore belongs to the same identity.
Similarity = One-Shot-Similarity(x₁, x₂, B)
Model1 = train(x₁, B)
Score1 = classify(x₂, Model1)

Model2 = train(x₂, B)
Score2 = classify(x₁, Model2)

Similarity = (Score1 + Score2)/2

Figure 5.1: Computing the symmetric One-Shot Similarity score for two vectors, x₁ and x₂, given a set B of negative examples.

as x₁, or whether it is closer to the background B and therefore doesn’t belong to the same identity as x₁.

- This model is then applied to the second vector, x₂, obtaining a classification score. In [54] an LDA classifier was used, and the score is the signed distance of x₂ from the decision boundary learned using x₁ (positive example) and B (negative examples).

- A second such score is then obtained by repeating the same process with the roles of x₁ and x₂ switched: this time, a model learned with x₂ as the positive example is used to classify x₁, thus obtaining a second classification score.

- The symmetric OSS is the average of these two scores. See Fig. 5.1.

As can be seen in LFW benchmark [24], using methods based on a background set improved performance, and thus the MBGS was motivated by trying to apply this technique to videos, or to any ‘set to set’ similarity.
Chapter 6

The Matched Background Similarity (MBGS)

A set-to-set similarity designed for comparing the frames of two face-videos, must determine if the faces appearing in the two sets are of the same subject, while ignoring similarities due to pose, lighting, and viewing conditions. In order to highlight similarities of identity, we train a discriminative classifier for the members of each video sequence. Doing so allows us the freedom to choose the particular type of classifier used, but more importantly, provides us with the opportunity to train a classifier using a ‘negative’ set selected to best represent misleading sources of variation. This negative set is selected from within a large set of background videos put aside for this purpose. For example, in our benchmark, the video sequences in the training splits are used as the background set (see Sec. 7.4).

As explained in Chapter 3, all the videos are encoded using a feature transform (e.g., LBP [25]) in $\mathbb{R}^d$. Assume a set $B = \{b_1, \ldots, b_n\}$ of background samples, $b_i \in \mathbb{R}^d$, containing a large sample of the frames in a ‘background-videos’ set. Given two videos, $X_1$ and $X_2$, likewise represented as two sets of feature vectors in $\mathbb{R}^d$, we compute their MBGS as follows (Fig. 6.1).

First we find the set $B_1 \subset B$ which represents the nearest neighbors of $X_1$. For this purpose we have tried four different approaches:

1. $K$ nearest neighbors:
   $B_1 = \{b \in B | \exists x \in X_1 \text{ so that } b \text{ is one of } x \text{’s first } K \text{ nearest neighbors in } B\}$
   [Find the $K$ nearest neighbors of for each $x \in X_1$ and group them.]

2. Fixed size $C$:
   (a) $B_1 = \emptyset$, $i = 1$
   (b) while $|B_1| < C$
      i. $T = \{b \in B | \exists x \in X_1 \text{ so that } b \text{ is } x \text{’s } i \text{th nearest-neighbor in } B\}$
         [Locate for each member (frame) of $X_1$ its $i$th nearest-neighbor in $B$.]
      ii. $B = B \cup T$
         [Aggregate all these matched frames discarding repeating ones.]
      iii. $i = i + 1$
   (c) trim $B_1$ to be at fixed size $C$ (remove the last matches)

3. Double fixed size $C$:
   (a) Calculate results for fixed size $C$ (as explained above)
Similarity = \text{MBGS}(X_1, X_2, B)

B_1 = \text{Find\_Nearest\_Neighbors}(X_1, B)
Model1 = \text{train}(X_1, B_1)
Confidences1 = \text{classify}(X_2, Model1)
Sim1 = \text{stat}(\text{Confidences1})

B_2 = \text{Find\_Nearest\_Neighbors}(X_2, B)
Model2 = \text{train}(X_2, B_2)
Confidences2 = \text{classify}(X_1, Model2)
Sim2 = \text{stat}(\text{Confidences2})

\text{Similarity} = \frac{\text{Sim1} + \text{Sim2}}{2}

Figure 6.1: Computing the symmetric Matched Background Similarity for two sets, \(X_1\) and \(X_2\), given a set \(B\) of background samples. The function \text{stat} represents either the mean, median, minimum or maximum over the confidences.

(b) Calculate results for fixed size \(2 \times C\)
(c) Average the results of both.

4. Fixed size \(C\) and KNN combined:

(a) Calculate results for fixed size \(C\) (as explained above)
(b) Calculate KNN (as explained above)
(c) Combine the results of both (average result of both).

In all four approaches we calculate the nearest neighbors by the Euclidean distance \(L_2\) metric, e.g. \(|x - b|, x \in X_1 \text{ and } b \in B\).

We have also tried applying all these approaches using the pose information, i.e. instead of calculating \(|x - b|\) we calculated the distance \(|p(x) - p(b)|, x \in X_1 \text{ and } b \in B\), where \(p(x)\) is the three rotation angles of the head as explained in Chapter 4.2. So using the pose distance instead of the regular \(L_2\) distance means choosing \(B_1\) as the neighbors with the nearest pose, instead of the neighbors with the nearest descriptor.

The method of using pose instead of regular \(L_2\) metric underperformed. The most likely reason is that the descriptor in \(\mathbb{R}^d\) contains more information than the pose in \(\mathbb{R}^3\).

The ‘FixedSize’ approach (using the L2 metric) outperformed the ‘\(K\) nearest neighbors’ approach and the other combined approaches. See experiments and results using all approaches in Section 8.5. From this point on we’ll refer to the ‘FixedSize, L2 metric’ approach only.

This set, \(B_1 \subset B, |B_1| = C\), is the set of background samples matching the vectors in \(X_1\). Provided that this set does not contain images of the same individual appearing in \(X_1\), \(B_1\) captures similarities to members of \(X_1\) resulting from factors other than identity (such as pose, lighting, and viewing conditions).

We now train a discriminative classifier (e.g., SVM) to distinguish between the two sets \(X_1\) and \(B_1\). A key observation is that the resulting discriminative model is trained to distinguish between similar feature vectors representing different identities.

There are two reasons for using the nearest neighbors and not the entire background:
• The main reason is to have the most discriminative classifier, discriminating between identities while being robust to all other factors. As seen in Chapter 8 the experiments confirm that using the entire background is suboptimal.

• In addition, training a classifier with a large number of samples take significantly more time as we saw in our experiments.

Using the model, we now classify all members of $X_2$ as either belonging to $X_1$ or $B_1$. For each member of $X_2$ we thus obtain a measure of classification confidence. These confidence values can be, for example, the signed distances from the separating hyperplane when using a SVM classifier. Each such confidence value reflects the likelihood that a member of $X_2$ represents the same person appearing in $X_1$.

We take the mean (or alternatively, the median, the minimum, or the maximum) of all these values, obtained for all of the members of $X_2$ as the one-sided MBGS. We mark it with $Sim_1$ which provides a global estimate for the likelihood that $X_2$ represents the same person appearing in $X_1$.

We then reverse the roles of $X_1$ and $X_2$ by selecting a set $B_2 \subset B$, $|B_2| = C$ of background samples matching the members of $X_2$. We train a discriminative classifier to distinguish between $X_2$ and $B_2$. Then using this second model, we classify all members of $X_1$ as either belonging to $X_2$ or $B_2$ and calculate their confidence level. Then we calculate $Sim_2$ which is the mean (or the median, or the minimum, or the maximum) of these confidence values.

The two-sided MBGS is obtained by the average of the two one sided similarities, which is the final MBGS score computed for the video pair.
Chapter 7

The ‘Youtube Faces’ set and benchmark

7.1 Motivation

In designing our video data set and benchmarks we follow the example of the ‘Labeled Faces in the Wild’ (LFW) image collection [19]. Specifically, our goal is to produce a large scale collection of videos along with labels indicating the identities of a person appearing in each video. In addition, we publish benchmark tests, intended to measure the performance of video pair-matching techniques on these videos. Finally, we provide descriptor encodings for the faces appearing in these videos, using well established descriptor methods. We next describe our database assembling process and associated benchmark tests.

7.2 Collection setup

1. We begin by using the 5,749 names of subjects included in the LFW data set [19] to search YouTube for videos of these same individuals.

2. The top six results for each query were downloaded.

3. We minimize the number of duplicate videos by considering two videos’ names with edit distance less than 3 to be duplicates.

4. Downloaded videos are then split to frames at 24fps.

5. We detect faces in these videos using a Viola and Jones face detector [44].

6. Automatic screening was performed to eliminate detections of less than 48 consecutive frames, where detections were considered consecutive if the Euclidean distance between their detected centers was less than 10 pixels. This process ensures that the videos contain stable detections and are long enough to provide useful information for the various recognition algorithms.

7. Finally, the remaining videos were manually verified to ensure that

   (a) The videos are correctly labeled by subject.
   (b) The videos are not semi-static, still-image slide-shows.
   (c) No identical videos are included in the database.
Table 7.1: YouTube faces. Number of videos available per subject.

<table>
<thead>
<tr>
<th># videos</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td># people</td>
<td>591</td>
<td>471</td>
<td>307</td>
<td>167</td>
<td>51</td>
<td>8</td>
</tr>
</tbody>
</table>

The screening process reduced the original set of videos from the 18,899 originally downloaded (3,345 individuals) to 3,425 videos of 1,595 subjects. An average of 2.15 videos are available for each subject (See Table 7.1 for a distribution of videos per subject). The shortest clip duration is 48 frames, the longest clip is 6,070 frames, and the average length of a video clip is 181.3 frames.

7.3 Database encodings

All video frames are encoded using several well-established, face-image descriptors. Specifically, we consider the face detector output in each frame. The bounding box around the face is expanded by 2.2 of its original size and cropped from the frame. The result is then resized to standard dimensions of 200 × 200 pixels. We then crop the image again, leaving 100 × 100 pixels centered on the face. Following a conversion to grayscale, the images are aligned by fixing the coordinates of automatically detected facial feature points [12], and we apply the following descriptors:

- Local Binary Patterns (LBP) [25].
- Center-Symmetric LBP (CSLBP) [18].
- Four-Patch LBP (FPLBP) [54].

These descriptors were selected as they were shown to be particularly useful when applied to unconstrained, face images [17]. In addition, all three descriptors are compact, which is an important consideration when producing a separate descriptor for each frame of each video. Other, less compact descriptors, known to be effective for representing images of unrestricted faces, were set aside for future work.

7.4 Benchmark tests

Following the example of the LFW benchmark, we provide standard, ten-fold, cross validation, pair-matching (‘same’/‘not-same’) tests. Specifically, we randomly collect 5,000 video pairs from the database, half of which are pairs of videos of the same person, and half of different people. These pairs were divided into 10 splits. Each split containing 250 ‘same’ and 250 ‘not-same’ pairs. Pairs are divided ensuring that the splits remain subject-mutually exclusive; if videos of a subject appear in one split, no video of that subject is included in any other split. The goal of this benchmark is to determine, for each split, which are the same and which are the non-same pairs, by training on the pairs from the nine remaining splits. We note that this split design encourages classification techniques to learn what makes faces similar or different, rather than learn the appearance properties of particular individuals.

One may consider two test protocols:

1. First, the restricted protocol limits the information available for training to the same/not-same labels in the training splits. This protocol is the one used in this work.

2. The Unrestricted protocol, on the other hand, allows training methods access to subject identity labels, which has been shown in the past to improve recognition results in the LFW benchmark [42].
There are several reasons for choosing a pair-matching benchmark rather than multi-identity classification. The same/not-same benchmarks are convenient as they provide a simple, binary interface to multiple class problems. However, since most real-world vision problems are multi-class in application as well as by nature, the suitability of such benchmarks as a key research tool is not obvious. Put differently, the applicability of the proposed benchmark to real-world face identification problems is not obvious, since a typical vision task requires naming a given video, not deciding if two videos are of the same person.

Some evidence demonstrating that the same performance pattern is observed for both tasks (same/not-same and multi-class identification) on the LFW data set was presented in [54].
Chapter 8

Experiments

We test the performance of several baseline face recognition methods (see Chapter 4.2) and compare them to the MBGS described in this work (see Section 6). Several types of methods were considered. One group consists of methods employing comparisons between pairs of face images taken from the two videos. Another group uses algebraic methods such as distances between projections. Such methods often appear in the literature as methods of comparing vector sets, particularly sets of face images. A third group includes the Pyramid Match Kernel and the Locality-constrained Linear Coding methods, which were proven extremely effective in comparing sets of image descriptors. We also include the performance obtained by using the straightforward heuristic of detecting an approximately frontal pose in each sequence and using this image to represent the entire set when comparing two videos. These methods are explained in Chapter 4.2, and next we relate specific experiments to the rows of Table 8.1.

As explained in Chapter 4.2, the notation is the following:
Let \( X_1 \) be the matrix whose columns are the encoding of the frames of one video, and let \( X_2 \) be the corresponding matrix for the other video.

8.1 All pairs comparisons

We compute a distance matrix \( D \) where \( D_{ij} = \|X_1(:,i) - X_2(:,j)\| \), \( X_1(:,i) \) denotes the \( i \)-th column of matrix \( X_1 \).
Five basic similarity measures are then computed using \( D \): min, average, median, max, and ‘meanmin’.

8.2 Pose based methods

1. Most frontal pose.
   Assuming that the easiest image to recognize in each set is the one showing the face in a frontal position, the most frontal face of each set is compared. Therefore the similarity measure is:
   \[\|X_1(:, \text{MostFrontalPose}_1) - X_2(:, \text{MostFrontalPose}_2)\|\].

2. Similar pose.
   This method uses one face image from each sequence by considering pairs of images with the smallest head rotation angle between them. Let \( (i_1, i_2) \) be the indexes of the frames with the two most similar poses in \( X_1 \) and \( X_2 \) respectively, then the similarity is:
   \[\|X_1(:, i_1) - X_2(:, i_2)\|\].
8.3 Algebraic methods

Algebraic methods view each matrix $X_1$ or $X_2$ as a linear subspace that is spanned by the columns of the matrix.

Let $U_i$, $i = 1, 2$ be any orthogonal basis for the subspace spanned by the columns of $X_i$. We can find these orthogonal matrices by using the singular value decomposition (SVD) of $X_1$ and $X_2$, such that $X_i = U_i S_i V_i^T$.

The SVD of $U_1^T U_2 = W S V_1^T$ provides the principle angles between the column subspaces of the two matrices $X_1$ and $X_2$. Specifically, the inverse cosine of the diagonal of $S$ are the principle angles, i.e., $S = diag(\cos \theta)$, where $\Theta$ is the vector of principle angles of $X_1$ and $X_2$. Note that this vector is sorted from the least angle to the largest.

In practice we used the eigenvalues: $d = eig(U_1^T U_2)$.

Several distances are defined based on these notations:

1. The max correlation is defined by the minimal angle $d(1)$.
2. The projection metric is given by $||U_1 U_1^T - U_2 U_2^T||_F$.
3. The norm $||U_1 U_2||_F$.
4. The Procrustes metric is computed from the vector-norm $||d||$.

As explained in Chapter 4.2, we restrict $U_1$ ($U_2$) to be the first $r = 10$ singular vectors of the subspace spanned by the columns of $X_i$.

The last algebraic is the CMSM method as explained in Chapter 4.2.

8.4 Non-algebraic Set methods

We next consider methods that have emerged as effective ways to represents sets, not necessarily in the context of computer vision.

1. Pyramid Match Kernel (PMK)

PMK represents each set of vectors as a hierarchical structure (‘pyramid’) that captures the histogram of the vectors at various levels of coarseness. We first construct the bins by hierarchical clustering on some representative videos, for each type of descriptor. Then once we had our vocabulary tree for this kind of descriptor, we used the vocabulary tree to create a pyramid for each face vector. The last step was to compute the kernel values from the pyramid.

2. Locality-constrained Linear Coding (LLC)

We represent the set by the maximal coefficients (one coefficient per dictionary vector) over all set elements. The sparse coefficients are computed based on the $k$-nearest dictionary vectors to each set element.

8.5 Matched Background Similarity

Since this method is new and one of the main contributions of this work, we’ll present its experiments in more details.

In MBGS samples from a large set $B$ of background samples are matched to each set sample. As there are 10 splits in the benchmark, each containing 500 pairs (1,000 different videos), we prepared in advanced a set of 10,000 background samples by randomly selecting one frame from each video. So samples 1..1000 are taken from split 1, samples 1001..2000 are taken from split 2 and so on.

Then we did two type of experiments:
• In the first experiment we used our 10 splits the following way:

1. 1 testing split.
2. 4 training splits.
3. 5 splits used as background.

This setup enabled us using a large background of up to 5,000 different background videos, and fine tuning the MGBS parameters. First we tried to find the optimal $K$ for the ‘$K$ nearest neighbors’ approach, and then tried to find an optimal $C$ value (the size of the nearest neighbors background set) for the ‘Fixed size’ approach. In addition we tested the other two approaches with several parameters, which underperformed as can be seen in Figure 8.1.

The conclusions from these experiments were that:

1. The best $K$ parameter is $K = 20$.
2. The best approach is ‘Fixed size’ with $C = 250$.
3. The combined approaches underperformed.

A graph showing the sensitivity of MBGS using the ‘Fixed size’ approach to the $C$ parameter is shown in Figure 8.2. As can be seen, plainly using the entire background set (without matching and selecting) is suboptimal.

A graph showing the sensitivity of MBGS to the $K$ parameter while using the ‘$K$ nearest neighbors’ approach is shown in Figure 8.3. Since the ‘fixed size’ approach gave the best results, from this point and on we only focused on this approach. We also note that the combined approaches took almost twice the running time without any added value and therefore neglected.

• We then use our splits as we did for testing all of the previous methods:

1. 1 testing split.
2. 8 training splits.
3. 1 split is used as background.

This enabled us to compare ‘apples to apples’, doing the same experiment for all similarity methods, with the same number of training splits. Background $B$ of 1,000 samples sufficed as the optimal $C$ was 250. The results of MBGS described in Section 8.6 are for the experiments done with this setup, using the ‘fixed size’ method with $C = 250$.

In all cases, the nearest neighbors set is performed by the smallest L2 metric of the descriptor.

Four methods are compared for combining the individual classification scores into one similarity value: mean, median, max and min.

To combine the two asymmetric scores, the average of the two scores is always used.

8.6 Results

Results are presented in Table 8.1. ROC curves of selected methods are presented in Figure 8.4. As detailed in Sec. 7.4, these results were obtained by repeating the classification process 10 times. Each time, we used eight sets for training, leaving out one for background (as previously explained) and evaluate the results on the tenth set. Results are reported by constructing the ROC curve for all splits together (the outcome value for each pair is computed when this pair is a testing pair), by computing statistics of the ROC curve (area under curve and equal error rate) and by recording
Figure 8.1: Success rate for MBGS using the LBP descriptor and the mean statistic for the different approaches, each approach with its best result, as described in Chapter 6.

average recognition rates ± standard errors for the 10 splits.

The results indicate that the newly proposed MBGS method outperforms the existing methods when considering all measures: recognition rate (‘accuracy’), area under curve and equal error rate.
The simplest min distance method is a solid baseline. The pose based heuristics, while popular, are not entirely effective. The algebraic methods are not better than the min-distance method. PMK and our adaptation of LLC underperform, although more exploration of their parameter space might improve their results.

To gain further insight into the challenges of the benchmark and the limitations of the current methods, Figure 8.5 presents the most confident cases according the variant of the MBGS method based on L2-norm matching and the mean operator. The most confident same-person predictions are of video sequences that are captured at the same scene, and the ‘easiest’ not-same are where there are multiple discriminating factors. The method might be fooled by motion blur, hats, and variation in illumination.

Figure 8.2: Success rate for MBGS using the LBP descriptor and the mean statistic as a function of the matched set size parameter C using the ‘Fixed Size’ approach.
Figure 8.3: Success rate for MBGS using the LBP descriptor and the mean statistic as a function of the matched set $K$ parameter using the ‘$K$ nearest neighbors’ approach.

Figure 8.4: ROC curves averaged over 10 folds. The plots compare the results obtained for LBP and FPLBP using the baseline mindist and $\|U_1^TU_2\|_F$ methods and the MBGS method, where the scores are combined by the mean operator.
Figure 8.5: Most confident MBGS results (L2 norm, mean operator). The Same/Not-Same labels are the ground truth labels, and the Correct/Incorrect labels indicate whether the method predicted correctly. For example, the top right quadrant displays same-person pairs that were most confidently labeled as not-same.

<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>CSLBP</th>
<th>FPLBP</th>
<th>LBP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy ± SE</td>
<td>AUC</td>
<td>EER</td>
<td>Accuracy ± SE</td>
</tr>
<tr>
<td>1</td>
<td>min dist</td>
<td>63.1 ± 1.1</td>
<td>67.3</td>
<td>37.4</td>
</tr>
<tr>
<td></td>
<td>max dist</td>
<td>56.5 ± 2.3</td>
<td>58.8</td>
<td>43.8</td>
</tr>
<tr>
<td></td>
<td>mean dist</td>
<td>61.1 ± 2.1</td>
<td>64.9</td>
<td>39.5</td>
</tr>
<tr>
<td></td>
<td>median dist</td>
<td>60.8 ± 2.1</td>
<td>64.8</td>
<td>39.4</td>
</tr>
<tr>
<td></td>
<td>&quot;meanmin&quot;</td>
<td>62.6 ± 1.5</td>
<td>66.5</td>
<td>38.3</td>
</tr>
<tr>
<td>2</td>
<td>most frontal</td>
<td>60.5 ± 2.0</td>
<td>63.6</td>
<td>40.4</td>
</tr>
<tr>
<td></td>
<td>nearest pose</td>
<td>59.9 ± 1.8</td>
<td>63.2</td>
<td>40.3</td>
</tr>
<tr>
<td>3</td>
<td>Max corr</td>
<td>58.4 ± 2.1</td>
<td>64.3</td>
<td>39.8</td>
</tr>
<tr>
<td></td>
<td>CMSM</td>
<td>61.2 ± 2.6</td>
<td>65.2</td>
<td>39.8</td>
</tr>
<tr>
<td></td>
<td>projection</td>
<td>50.1 ± 0.2</td>
<td>45.7</td>
<td>53.1</td>
</tr>
<tr>
<td></td>
<td>$|U_1^T U_2|_F$</td>
<td>63.8 ± 1.8</td>
<td>67.7</td>
<td>37.4</td>
</tr>
<tr>
<td></td>
<td>procrusts</td>
<td>62.8 ± 1.6</td>
<td>67.1</td>
<td>37.5</td>
</tr>
<tr>
<td>4</td>
<td>PMK</td>
<td>52.7 ± 2.2</td>
<td>53.1</td>
<td>47.0</td>
</tr>
<tr>
<td></td>
<td>LLC</td>
<td>51.5 ± 2.1</td>
<td>53.4</td>
<td>48.1</td>
</tr>
<tr>
<td>5</td>
<td>mean</td>
<td>72.4 ± 2.0</td>
<td>78.9</td>
<td>28.7</td>
</tr>
<tr>
<td></td>
<td>median</td>
<td>72.4 ± 1.9</td>
<td>78.9</td>
<td>28.5</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>70.5 ± 1.5</td>
<td>77.1</td>
<td>29.9</td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>67.1 ± 2.6</td>
<td>73.9</td>
<td>33.1</td>
</tr>
</tbody>
</table>

Table 8.1: Benchmark results obtained for various similarity measures and image descriptors. See text for the description of each method.
Chapter 9

Discussion and Future Work

There are few directions for future work.

In all the above experiments we used a single similarity measure for classification (same/not-same) and for ROC results. When combining several of these similarities into a similarity vector, it can outperform the single similarity results, by enabling the classifier to choose the weights of the different similarity measures. For example, using a vector combining all similarity measures with accuracy over 60% outperformed by 2% the best single similarity measure. This means there is place for future work in this direction.

In our benchmark database we used the LBP, CSLBP and FPLBP encodings. These encodings can vary by using different parameters, which may achieve better results. More efforts should be made in this direction.

In addition, there are some other known encoding descriptors used for face images, such as Three Patch LBP (TPLBP) [54], the Elastic Bunch Graph Matching (EBGM) [50], the LDA based descriptor [11], or some SIFT base encoding such as the dense Scale Invariant Feature Transform (d-SIFT) [45]. More work should be done encoding the database frames using these encodings, and perhaps outperforming the current results. More over, since face recognition in videos is relatively a young research field, it is likely that designing new designated encoding techniques for this purpose might also achieve better performance.

Some more future work can be done improving the MBGS itself. In its core it uses a built in classifier (e.g. SVM). Investing more efforts trying several sets of classifiers and fine tuning their parameters, can result better classification performance using the MBGS similarity. Possibly doing a variant of one of the known classifiers or designing a new classifier designated for MBGS, might also improve the classification performance. This effort can also be done in a future work.

It is interesting to compare the MBGS to the recent background similarities used in face recognition from static images, namely the One-Shot Similarity (OSS) [51], the Two-Shot Similarity (TSS) [52], and the Multi-shot similarity [42]. First, for the specific case of using a linear hard-margin SVM classifier, MBGS generalizes the OSS. This, since such a classifier employed between a single positive example and a negative set has only one negative support vector which is the closest negative point to that positive point [4]. The TSS examines the case where two points are separated from a background set B. A similar set-to-set similarity can be constructed. Initial tests with this TSS analog do not show promising performance. The TSS, however, has been shown to best perform in concert with the OSS and not by itself. Finally, the multi-
shot similarity is the extension of the OSS for the case where multiple background sets are used. It is easy to extend MBGS in a similar way. This effort, however, is left for future work.

Another future direction would be testing the unrestricted protocol. In addition to the same/not-same labeling, the unrestricted protocol also provides the identity for each video. Some algorithms may take advantage of this additional information, as seen in the LFW results. Testing the unrestricted protocol, and implementing an algorithm that can take advantage of this information is left for future work.

In our work our emphasize wasn’t done on efficiency. After achieving better classification performance, more efforts should be made on how it should be done more efficiently. In the future it will be possible to reduce the computation time of MBGS among other optimizations. Initial exploration in this direction indeed shows improved efficiency with minimal affect on recognition performance.

In a more general context, the problem of comparing sets of vectors is a cornerstone in modern object recognition. In the future we plan to test and adopt the MBGS method for tasks outside the face recognition domain.
Appendix A

The website, the dataset and the code package

The database, image encoding, benchmark tests, paper results and the code of the baseline methods are available at [www.cs.tau.ac.il/~wolf/ytfaces](http://www.cs.tau.ac.il/~wolf/ytfaces).

The Database:

1. `frame_images_DB.tar.gz`
   Contains the videos downloaded from youtube, broken to frames.
   The directory structure is:
   
   `subject_name/video_number/video_number.frame.jpg`

   For each person in the database there is a file called `subject_name.labeled_faces.txt`
   The data in this file is in the following format:
   
   filename, [ignore], x, y, width, height, [ignore], [ignore], [ignore]
   
   where:
   
   x, y are the center of the face and the width and height are of the rectangle that 
   the face is in.
   For example:
   
   ```
   $ head -3 Richard_Gere.labeled_faces.txt
   Richard_Gere/3/3.618.jpg,0,262,165,132,132,0.0,1
   Richard_Gere/3/3.619.jpg,0,260,164,131,131,0.0,1
   Richard_Gere/3/3.620.jpg,0,261,165,129,129,0.0,1
   ```

2. `aligned_images_DB.tar.gz`
   Similar to `frame_images_DB`, contains the videos downloaded from youtube 
   broken to frames, but after some manipulation:
   
   (a) face detection, expanding the bounding box by 2.2 and cropping from the 
   frame.
   (b) alignment.

   The directory structure is:
   
   `subject_name/video_number/aligned_detect_video_number.frame.jpg`

3. `descriptors_DB.tar.gz`
   Contains mat files with the descriptors of the frames.
   The directory structure is:
   
   `subject_name/mat files`
For each video there are two files:
aligned_video_1.mat
video_1.mat

The files contain descriptors per frame, several descriptors type per frame.
One contains the aligned version of the faces in the frame and the other contains
the not aligned version.
Each of the above file has a struct with the following (for example a video with
80 frames):

VID_DESCS_FILENAMES: {1x80 cell}
VID_DESCS_FPLBP: [560x80 double]
VID_DESCS_LBP: [1770x80 double]
VID_DESCS_CSLBP: [480x80 double]

These are the different descriptors and the file names.

4. meta_data.tar.gz
Contains the meta_and_splits.mat file, which provides an easy way for accessing
the mat files in the descriptors DB. The Splits is a data structure dividing the
data set to 10 independent splits.
Each triplet in the Splits is in the format of (index1, index2, is_same_person),
where index1 and index2 are the indexes in the mat_names structure. All to-
gether 5000 pairs divided equally to 10 independent splits, with 2500 same pairs
and 2500 not-same pairs.

video_labels: [1x3425 double]
video_names: [3425x1 cell]
mat_names: [3425x1 cell]
Splits: [500x3x10 double]

5. headpose_DB.tar.gz
Contains mat files with the three rotation angles of the head for each frame in
the data set.
The directory structure is:
headorien_apirun_subject_name_video_number.mat

Each mat file contains a struct with the following:
headpose: [3x60 double]

The Code Package:

We provide the source code for running the benchmark tests and implementation
of all methods. All in sources.tar.gz.

Use conf.txt to specify the location of the sources, e.g. the location of all external
libraries, the location of the results files, which similarity methods to run, etc.
The scripts directory contains all the relevant scripts to create the similarity
results.
The common usage is:

(a) run calc_similarity.pl to create the similarity distances.
(b) use progress.pl to monitor the progress of all the processes that run calc_similarity.pl.
(c) after calc_similarity.pl is done, use analyze_results.pl to analyze the results.
2. Scripts:

(a) `analyze_results.pl`
After running `calc_similarity.pl`, all the similarity distances per method are stored in the results files in the results directory defined in the configuration file. Analyze results will read all the results files, and output a CSV file with the analyzed results. The file is stored in the results directory.

(b) `calc_similarity.pl`
A script that calculates the similarities in all pairs in all methods, split by split. It enables to run several processes in parallel.
For example, to run 10 processes on the aligned descriptors run:
    `calc_similarity.pl 1 10 10`
for more info on this script run:
    `calc_similarity.pl -h`

(c) `create_composite_results.m`
An example script on how to take a sub set of the similarities measures and create a composite measure that in many cases gives better results. It contains few examples showing the main concept.

(d) `create_paper_results.m`
A matlab wrapper to calculate all the similarity measures, to unify the results from all the different instances, and to analyze the results. The scripts `calc_similarity.pl` and `analyze_results.pl` call it with the relevant parameters.

(e) `print_conf.pl`
Prints the current configuration.

(f) `progress.m`
A matlab script that unifies all the results created by `calc_similarity`, and outputs how many pairs were already calculated, showing the progress of all the processes of `calc_similarity.pl`.

(g) `progress.pl`
A wrapper script to run the `progress.m` - use it to monitor the progress of all the processes that run `calc_similarity.pl`.

(h) `run_matlab_bg.csh`
A script to run a matlab program in the background.

3. `src` directory:
Contains all the sources for all similarity methods, analyze results, etc.

4. `libs` directory:
Contains external libraries such as the classifiers, PMK, LLC, and ROC.
Some of these libraries were implemented by us and some of them were taken from open source projects. The relevant copy rights and licenses appear withing these directories.
Bibliography


