Feature Selection for Fast Face Matching

Ke 'Colin' Zheng
kzheng@cs.washington.edu

Abstract

In this paper, we present a novel algorithm performing face image comparison. Ababoosting is applied to find the right features in the offline phase and use them to determine similarity quickly while online. We build hierarchical classifiers to speed up the online matching process. Extension to our method is described to count for misalignment images, thus shifted images can be handled correctly. Experiments show that we achieve good accuracy at a high speed.

I. Introduction

Given two images as a pair, can you quickly tell if they are different or similar? And further, can you tell how different they are? To put it another way, given an image as a query, can you find the most similar image to it from a database?

People have been trying to answer these questions for ages ever since computer vision was born. Various approaches and metrics were proposed [3, 4, and 5]. We can roughly categorize the approaches into pixel based and feature based methods where pixels or features are used for comparison. One of the reasons for having these two different approaches goes to the two sides of the coins that we concern about: speed and accuracy.

Feature based approaches are usually fast, since they work on the certain type of information extracted from images, such as edge or histogram information. Yet the features might be limited in a sense that it can't completely solve the problem. Sum of pixel difference is the other way to go, but speed problem arouses for this type method as accessing each pixel multiple times would consume significant amount of memory and computation. Image difference, or say image matching, is the basis for content based image retrieval [6]. Speed becomes a bigger issue for this type of application since the search space grows as the image database grows.

In this paper, we present a fast image comparison technique based on feature selection. We constrain ourselves to face images only, and even further, we focus on faces of a particular persons as we are interested in finding features that are not across person.

Inspired by Viola and Jones [2] work in object detection, we achieve feature selection using model ensemble. Given a big pool of single feature classifiers, we build a strong classifier from these weak ones using adaboosting. Further, we take the advantage of the hierarchical structure to build cascaded classifiers to speed up the matching process. But this approach breaks when faces are not aligned. So we extended the approach using a family of shiftable features instead of the previous single features. As results would show, this significantly improves the accuracy for cases of misalignment in translation.

In section 2, we show the setup for data collection. In section 3, the feature selection framework is presented. Section 4 covers the cascaded feature selection for speed up while Section 5 solves the misalignment problem by shiftable features. Results are show in section 6 with a short discussion for future work.

II. Data Acquisition

We collect face images from video streams of people talking with a state-of-art face tracker hooked in the front end. Based on the face box located by the tracker, face images are cropped, resized and stored. For each speaker, we build a database of faces of fixed image size. Our goal is to do face matching in the database as fast as possible. Note even the current best face tracker can't locate the face to pixel-wise accuracy, so face images are likely to be off by a few pixels for each other. As we would see in section 5, this turns out to be a big issue for our feature selection model.

Figure 1. Face Image in Database

III. Fast Matching via Learned Features

The feature set used here is very similar to Viola and Jones [2] used for fast object detection. A set of features is defined by differences of rectangular regions in a summed area table [1] which they have named an *integral image*. For an image, I, with pixel I_{uv}, the corresponding pixel in the integral image II_{uv} is equal to the sum of all pixels above (or below) and to the left (or right) of it.

$$II_{uv} = \sum_{i=0 \rightarrow u,\ j=0 \rightarrow v} I_{ij}$$

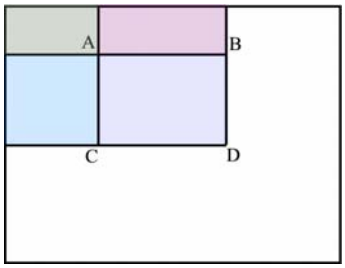

Figure 2: Rectangle ABCD = D + A - B – C

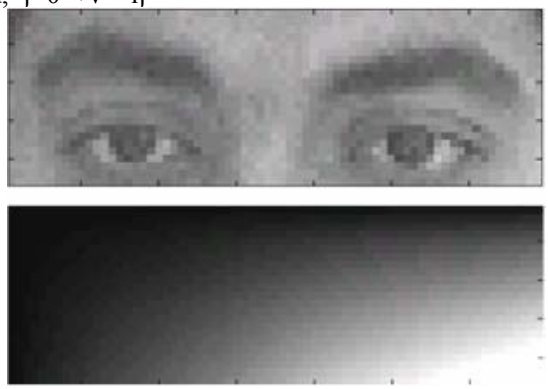

Figure 3: Eye image and corresponding integral image.

The integral image can be computed in a single pass through the image. It then allows the fast evaluation of the integral of any rectangle in the image by accessing and summing or differencing only four pixels. For example, the sum of all pixels in rectangle ABCD is equal to $II_D + II_A - II_B - II_C$. The feature set defined by Viola and Jones includes sums and differences of two, three, and four adjacent rectangles, which can be evaluated from six, eight, and nine integral image pixel references respectively. The score of a particular feature, *F*, applied to an image, *a*, is simply *F(a)* = the sum of the white (+) regions minus the sum of the black (-) regions.

The set of all possible features of any size is very large. They represent an over-complete basis of Haar wavelet-like filters.

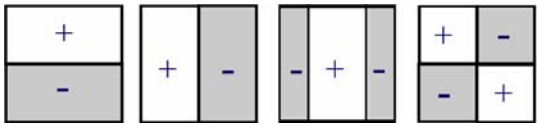

Figure 4: Features represent sums and differences of two, three, and four rectangles.

The hypothesis is that the values of particular small subset of the features can be used to distinguish between a face and a non-face. Viola and Jones [2] describe a machine learning approach to find such a set. They present the classifier with a set of positive (faces) and negative (non-face) examples and determine the best features to be used for the classifier and threshold values for each feature. They leverage the fact that most rectangles in an image do not contain a face by training a cascaded set of face rejection classifiers, each of which can reject most non-faces and pass on only likely candidates for the next phase of the cascade.

We then extend this work for the task of determining whether one face is sufficiently similar to the other. This new task requires some rethinking of the machine learning methodology. In our case, the classifier cannot act on a single image alone (face vs. non-face) but must act on a pair of images to classify them as similar or non-similar. To do so, we alter the score that a particular feature returns to be the *absolute difference* between the unary feature scores on each of the pair of images under consideration,

$$F(a,b) = | F(a) - F(b) |.$$

In addition, in Viola and Jones' work, the training set was carefully constructed and hand annotated. We wish to train our classifier automatically from video sequences. To do so, we annotate each pair as similar (or non-similar) based on the pixel luminance L_2 difference between them. More specifically, we examine a sequence of training images, and measure and record the L_2 difference between all pairs of images. We then annotate pairs to be similar if they are at least α standard deviations less different than the mean difference:

$$\mathbf{S(a,b) = 1 \quad if \quad \|a - b\|_2 < \mu - \alpha\sigma,}$$
$$\mathbf{0 \quad otherwise.}$$

where μ and σ are the mean and standard deviation of the L_2 distances between all pairs of images in a sequence from a single person.

Given a positive α we are (almost) assured that most pairs will be marked as non-similar. We will use this fact to construct an efficient classifier by training a cascade of classifiers, each of which is good at removing many non-similar pairs (and very few similar pairs) from contention, while possibly allowing some non-similar pairs to slip through. In other

words, each stage of the cascade should err on the side false positives over allowing false negatives. Taken together, the complete cascade should efficiently classify all pairs.

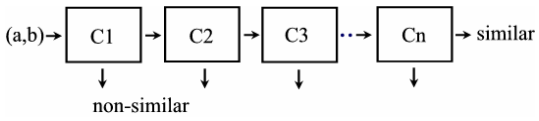

Figure 5: Each classifier in a cascade rejects non-similar images. Each stage thus acts on a smaller subset of possible pairs allowing only the best matches to pass all the way through.

For each stage of the classifier we seek one or more features and associated threshold values for the feature(s). To err on the side similarity we weight each false negative β times the cost of a false positive.

- Given pairs of images and their similarity (P_i,S_i), $S_i = 0, 1$ for (non-similar vs. similar)
- Initialize each pair's weight, $w_{i,0} = \frac{1}{N_S}$ *or* $\frac{1}{N_{NS}}$ depending on the pair's similarity, and N_S , N_{NS} = number of pairs annotated similar and non-similar.
- For as many features F_j as needed
  1. Normalize the weights to sum to 1
  2. For all possible features F_j, find a threshold T_j that minimizes the misclassifications = $\beta FN + FP$. We penalize false negatives (FN) more than false positives (FP) by a factor of $\beta=5$. For each pair $F_j(P_i)=1$ (similar) if F_j returns a score with absolute value less than T_j, or 0 otherwise.
  3. An error, e_j is defined to be $e_j = \Sigma_i w_{i,j} |F_j(P_i)-S_i|$. Given all possible (F_j,T_j) choose the one that minimizes e_j.
  4. Set the voting weight of F_j to be $W_j = log[(1-e_j)/e_j]$
  5. Sum the votes for each pair to classify it $V = \Sigma_j W_j F_j(P) / \Sigma_j W_j$
  6. If V > τ then P is marked as similar, otherwise non-similar. τ is set to 0.5 or is lowered to allow more pairs to be marked similar until the *accuracy* test (.98 times as accurate as the previous stage) is passed.
  7. If the chosen feature(s) pass the FP test (.4 times lower % false positives) go on to the next stage of the cascade, otherwise
  8. Reset all the weights to decrease the importance of the pairs that were properly classified $w_{i,j+1} = w_{i,j} [e_j / (1-e_j)]$. Weights of misclassified pairs remain unchanged (until normalization in step 1).
  9. Go to 1 and choose an additional feature for this stage of the cascade.

How many stages should the cascade contain? More specifically, how can one know when each stage is trained? Ideally each stage will find the minimum number of features to reject some fraction of non-similar pairs while incurring minimal error. What should the fraction be and what kind of error should be tolerated?

In other words, there are some parameters to be chosen, but most can be guided by intuition, empirical experience, and experience gleaned from the literature. We train each stage of the cascade until the cascade's false positive rate is lowered by 40%. In other words, of all pairs that pass through the first stage, a maximum of 40% can have been marked as non-similar. After each successive stage the false positive (*FP*) rate must be at most 0.4 times the rate after the previous stage. At the same time, each stage is only allowed to lower its *accuracy* (1 – false negative rate) by 98%. In other words, the first stage can only allow the rejected pairs to contain at most 2% that have been annotated as similar by the L_2 norm test. The accuracy can drop by 98% each stage thereafter.

The cascade is stopped when a desired false positive rate is met or there is no more improvement. Once the cascade is trained and the features and their thresholds and voting weights set, new image pairs can be evaluated as similar or non-similar extremely efficiently. For images that are inserted in the database, we store all the unary feature scores.

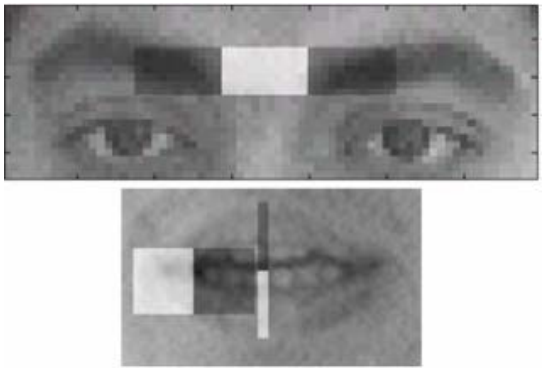

Figure 6: First feature for eyes helps locate bright bar between eyes. First two features found for mouth help locate and measure relative shape of mouth.

We then need only to determine the feature scores for the new image and compare their difference from the database image under consideration against the threshold for that feature.

The setting for query in face database is slightly different. We pair the query image with every image in the database and send them to the cascade classifier. For image pairs that pass through the whole cascade, we add up their scores for each layer to be the final score. We sort this score for all pairs survived the cascade, and return the minimum one to be the most similar one.

VI. Shiftable Features

The approach described above has an important assumption implied that all face images are aligned perfectly as we would apply the same features to all images at the exact same location, with exact same dimension. What's more, L2 norm which we used as the ground truth assumes pixel-wise alignment as well.

But the actual data we collect violates this assumption heavily, due to the instability of the face tracker. As the face box located by the tracker might shake along the stream, quite a number of faces are shifted a few pixels off. We extend the previous framework to make it applicable for misaligned data.

The first extension is about the L2 norm which we used as ground truth. L2 norm is very sensitive to alignment, so we extend it to shift window based L2 norm where we shift one face image around and calculate the L2 difference. We pick the minimum L2 norm among all shifted versions as the L2 difference. This gives us a better ground truth to start with.

The other extension is about the training process itself. In our adaboosting framework, instead of finding one feature every cycle, we find a family of features with the same dimension, but shifted by 1 pixel in location with each other. Among this series of features, we pick the one with minimum error to represent the whole family. And the training framework as well as the classification framework remains the same. The intuition behind it is that by having families of shifted features, we gain much more flexibility which in turns gives much better results, for the simple reason that the features doesn't have to be at the exact location for all face images, they can be off by a few pixels yet still be picked.

As a whole, we need to spend more time in training as well as classification since we need to count for all shifted cases, but the classification is still very light weighted as the number of feature families are quite small.

V. Results

We have carried out experiments with two different types of data, one with face tracker and one without it. For the one without face tracker, we simply fixed the location to crop face out. Our fixed features approach work well on data with manual cropping as the pictures are better aligned while our shifted feature approach work well on data cropped by face tracker. The following is the result for shifted feature approach.

We tested the feature training on data from a single person consisting of about 9000 pairs of images. Half were used for training with the other half reserved for testing. We first trained a single classifier consisting of 100 features. This resulted in 92% classification accuracy and 2.7% false positives.

A cascaded-classifier produced 94.2% classification accuracy and 4.1% false positives. This classifier is extremely simple with three layers of only 2, 4 and 4 features respectively. The result is comparable to a single classifier with many more features.

We also tested for finding the most similar one out of the database with the same experiment setup. We use shifted L2 norm to find the most similar one as a comparison. 26% of them return the same face from database and if we increase to find the most similar three, then 67% of return at least one same face from database.

Our approach is faster in order of magnitude than normal pixel based method. This can be seen by a simple complexity analysis.

In terms of time complexity, given image size to be S and the number of images to be N, finding the most similar one in database needs time $O(NS)$ for pixel based approach. For our method, given the number of features to be M, we need time $O(NM+S)$ to do the same task. $O(S)$ is the time to build the integral image and $O(NM)$ is the time to do the searching. M is usually way smaller than S, that's why it's faster.

In terms of space complexity, we need space *O(NM+S)* as we only need to store the feature scores for all the images in database, not the whole images. Pixel based method would need *O(NS)* space for storing everything, which is quite expensive once the database gets large.

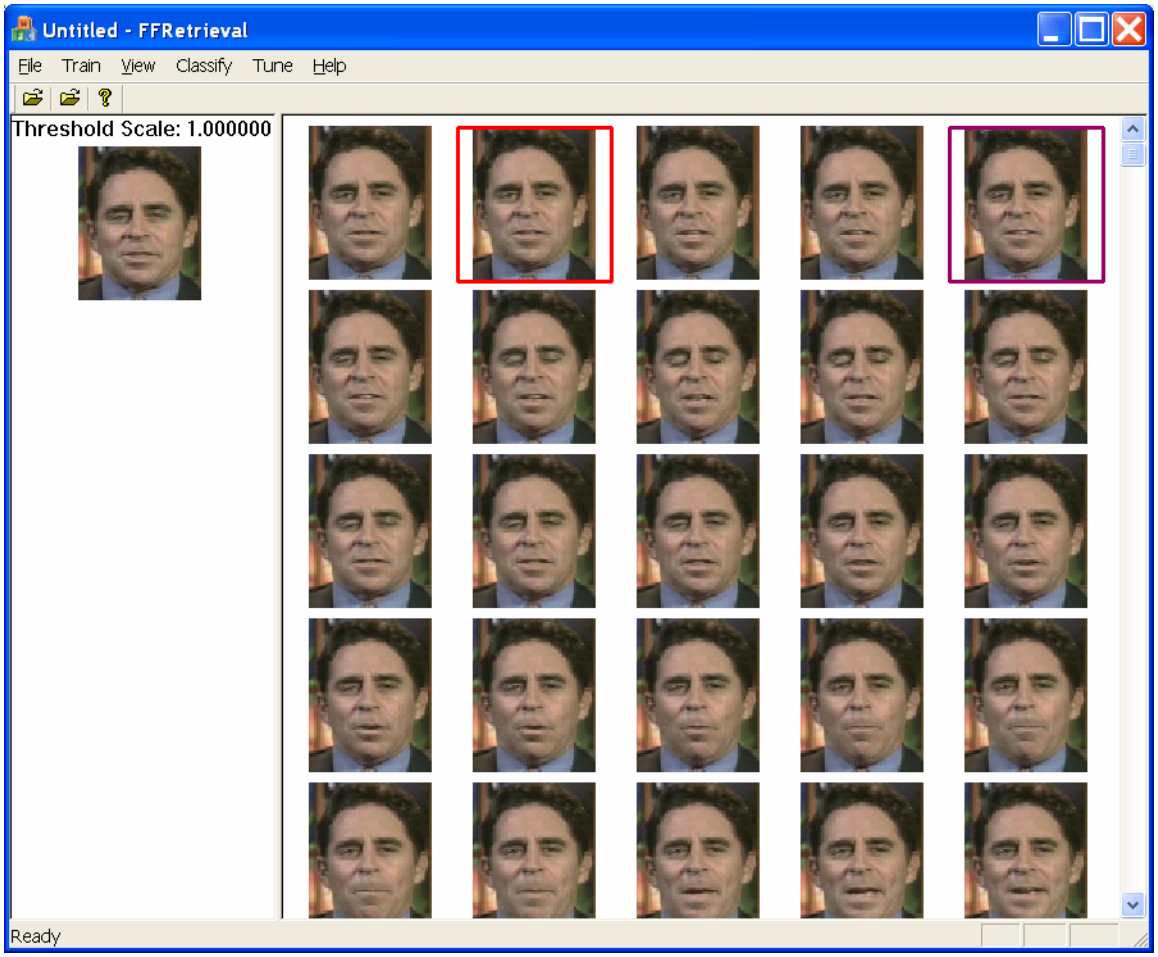

Figure 7. Snapshot of the System, left view is the query while the right view is the database, faces that are boxed are the similar ones returned.

To summarize, we proposed a way to compare images quickly and accurately. We applied ababoosting to find the right features in the offline phase and used them to determine similarity quickly while online. We extend our method to count for misalignment images and we are able to handle shifting. We still can't solve scaling and rotation misalignment, yet it's promising as our model ensemble framework is very flexible.

Reference:

1. Crow, F. C., "*Summed-Area Tables for Texture Mapping* ", Proceedings of SIGGRAPH `84, Computer Graphics, Vol. 18, No. 3, July 1984, pages 207-212
2. Viola, P. & Jones, M., Robust real-time object detection, International workshop on statistical and computational theories of vision, 2001.

3. B. Girod, ``What's Wrong with Mean-squared Error?'', in A. B. Watson, editor, Digital Images and Human Vision, pp. 207-220, MIT Press, 1993.
4. C. E. Jacobs, A. Finkelstein, D. H. Salesin, ``Fast Multiresolution Image Querying'', Computer Graphics (Proceedings of Siggraph '95), pp. 277-286, (1995)
5. H. Rushmeier, G. Ward, C. Piatko, P. Sanders, B. Rust, ``Comparing Real and Synthetic Images: Some Ideas About Metrics'', Sixth Eurographics Workshop on Rendering, Dublin, Ireland, pp. 82-91, (1995)
6. R. C. Veltkamp, M. Tanase, “Content-Based Image Retrieval Systems: A Survey”, Technical Report UU-CS-2000-34,(2000)