# Cosine Similarity Metric Learning for Face Verification

**Conference Paper** · November 2010

DOI: 10.1007/978-3-642-19309-5_55 · Source: DBLP



**2 authors**, including:

Li Bai
University of Nottingham
**204** PUBLICATIONS **2,833** CITATIONS

SEE PROFILE

# Cosine Similarity Metric Learning for Face Verification

Hieu V. Nguyen and Li Bai

School of Computer Science, University of Nottingham,
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
{vhn,bai}@cs.nott.ac.uk
http://www.nottingham.ac.uk/cs/

**Abstract.** Face verification is the task of deciding by analyzing face images, whether a person is who he/she claims to be. This is very challenging due to image variations in lighting, pose, facial expression, and age. The task boils down to computing the distance between two face vectors. As such, appropriate distance metrics are essential for face verification accuracy. In this paper we propose a new method, named the Cosine Similarity Metric Learning (CSML) for learning a distance metric for facial verification. The use of cosine similarity in our method leads to an effective learning algorithm which can improve the generalization ability of any given metric. Our method is tested on the state-of-the-art dataset, the Labeled Faces in the Wild (LFW), and has achieved the highest accuracy in the literature.

Face verification has been extensively researched for decades. The reason for its popularity is the non-intrusiveness and wide range of practical applications, such as access control, video surveillance, and telecommunication. The biggest challenge in face verification comes from the numerous variations of a face image, due to changes in lighting, pose, facial expression, and age. It is a very difficult problem, especially using images captured in totally uncontrolled environment, for instance, images from surveillance cameras, or from the Web. Over the years, many public face datasets have been created for researchers to advance state of the art and make their methods comparable. This practice has proved to be extremely useful.

FERET [1] is the first popular face dataset freely available to researchers. It was created in 1993 and since then research in face recognition has advanced considerably. Researchers have come very close to fully recognizing all the frontal images in FERET [2,3,4,5,6]. However, these methods are not robust to deal with non-frontal face images.

Recently a new face dataset named the Labeled Faces in the Wild (LFW) [7] was created. LFW is a full protocol for evaluating face verification algorithms. Unlike FERET, LFW is designed for unconstrained face verification. Faces in LFW can vary in all possible ways due to pose, lighting, expression, age, scale, and misalignment (Figure 1). Methods for frontal images cannot cope with these variations and as such many researchers have turned to machine learning to


**Fig. 1.** From FERET to LFW

develop learning based face verification methods [8,9]. One of these approaches is to learn a transformation matrix from the data so that the Euclidean distance can perform better in the new subspace. Learning such a transformation matrix is equivalent to learning a Mahalanobis metric in the original space [10].

Xing et al. [11] used semidefinite programming to learn a Mahalanobis distance metric for clustering. Their algorithm aims to minimize the sum of squared distances between similarly labeled inputs, while maintaining a lower bound on the sum of distances between differently labeled inputs.

Goldberger et al. [10] proposed Neighbourhood Component Analysis (NCA), a distance metric learning algorithm especially designed to improve kNN classification. The algorithm is to learn a Mahalanobis distance by minimizing the leave-one-out cross validation error of the kNN classifier on a training set. Because it uses softmax activation function to convert distance to probability, the gradient computation step is expensive.

Weinberger et al. [12] proposed a method that learns a matrix designed to improve the performance of kNN classification. The objective function is composed of two terms. The first term minimizes the distance between target neighbours. The second term is a hinge-loss that encourages target neighbours to be at least one distance unit closer than points from other classes. It requires information about the class of each sample. As a result, their method is not applicable for the restricted setting in LFW (see section 2.1).

Recently, Davis et al. [13] have taken an information theoretic approach to learn a Mahalanobis metric under a wide range of possible constraints and prior knowledge on the Mahalanobis distance. Their method regularizes the learned matrix to make it as close as possible to a known prior matrix. The closeness is measured as a Kullback-Leibler divergence between two Gaussian distributions corresponding to the two matrices.

In this paper, we propose a new method named Cosine Similarity Metric Learning (CSML). There are two main contributions. The first contribution is

that we have shown cosine similarity to be an effective alternative to Euclidean distance in metric learning problem. The second contribution is that CSML can improve the generalization ability of an existing metric significantly in most cases. Our method is different from all the above methods in terms of distance measures. All of the other methods use Euclidean distance to measure the dissimilarities between samples in the transformed space whilst our method uses cosine similarity which leads to a simple and effective metric learning method.

The rest of this paper is structured as follows. Section 2 presents CSML method in detail. Section 3 present how CSML can be applied to face verification. Experimental results are presented in section 4. Finally, conclusion is given in section 5.

## 1 Cosine Similarity Metric Learning

The general idea is to learn a transformation matrix from training data so that cosine similarity performs well in the transformed subspace. The performance is measured by cross validation error (cve).

### 1.1 Cosine similarity

Cosine similarity (CS) between two vectors $x$ and $y$ is defined as:

$$CS(x, y) = \frac{x^T y}{\|x\| \, \|y\|}$$

Cosine similarity has a special property that makes it suitable for metric learning: the resulting similarity measure is always within the range of $-1$ and $+1$. As shown in section 1.3, this property allows the objective function to be simple and effective.

### 1.2 Metric learning formulation

Let $\{x_i, y_i, l_i\}_{i=1}^{s}$ denote a training set of $s$ labeled samples with pairs of input vectors $x_i, y_i \in R^m$ and binary class labels $l_i \in \{1, 0\}$ which indicates whether $x_i$ and $y_i$ match or not. The goal is to learn a linear transformation $A : R^m \rightarrow R^d (d \leq m)$, which we will use to compute cosine similarities in the transformed subspace as:

$$CS(x, y, A) = \frac{(Ax)^T (Ay)}{\|Ax\| \, \|Ay\|} = \frac{x^T A^T A y}{\sqrt{x^T A^T A x}\sqrt{y^T A^T A y}}$$

Specifically, we want to learn the linear transformation that minimizes the cross validation error when similarities are measured in this way. We begin by defining the objective function.

### 1.3 Objective function

First, we define positive and negative sample index sets $Pos$ and $Neg$ as:

$$Pos = \{i | l_i = 1\}$$

$$Neg = \{i | l_i = 0\}$$

Also, let $|Pos|$ and $|Neg|$ denote the numbers of positive and negative samples. We have $|Pos| + |Neg| = s$ - the total number of samples.

Now the objective function $f(A)$ can be defined as:

$$f(A) = \sum_{i \in Pos} CS(x_i, y_i, A) - \alpha \sum_{i \in Neg} CS(x_i, y_i, A) - \beta \left\| A - A_0 \right\|^2$$

We want to maximize $f(A)$ with regard to matrix $A$ given two parameters $\alpha$ and $\beta$ where $\alpha, \beta \geq 0$. The objective function can be split into two terms: $g(A)$ and $h(A)$ where

$$\begin{aligned} g(A) &= \sum_{i \in Pos} CS(x_i, y_{i,} A) - \alpha \sum_{i \in Neg} CS(x_i, y_i, A) \\ h(A) &= \beta \left\| A - A_0 \right\|^2 \end{aligned}$$

The role of $g(A)$ is to encourage the margin between positive and negative samples to be large. A large margin can help to reduce the training error. $g(A)$ can be seen as a simple voting scheme from each sample. The reason we can treat votes from samples equally is that cosine similarity function is bounded by 1. Additionally, because of this simple form of $g(A)$, we can optimize $f(A)$ very fast (details in section 1.4). The parameter $\alpha$ in $g(A)$ is to balance the contributions of positive samples and negatives samples to the margin. In practice, $\alpha$ can be estimated using cross validation or simply be set to $\frac{|Pos|}{|Neg|}$. In the case of LFW, because the numbers of positive and negative samples are equal, we simply set $\alpha = 1$.

The role of $h(A)$ is to regularize matrix $A$ to be as close as possible to a predefined matrix $A_0$ which can be any matrix. The idea is both to inherit good properties from matrix $A_0$ and to reduce the training error as much as possible. If $A_0$ is carefully chosen, the learned matrix $A$ can achieve small training error and good generalization ability at the same time. The parameter $\beta$ plays an important role here. It controls the tradeoff between maximizing the margin ($g(A)$) and minimizing the distance from $A$ to $A_0$ ($h(A)$).

With the objective function set up, the algorithm can be presented in detail in the next section.

**Algorithm 1** Cosine Similarity Metric Learning

**INPUT**

- $S = \{x_i, y_i, l_i\}_{i=1}^{s}$: a set of training samples ($x_i, y_i \in R^m$, $l_i \in \{0, 1\}$)
- $T = \{x_i, y_i, l_i\}_{i=1}^{t}$: a set of validation samples ($x_i, y_i \in R^m$, $l_i \in \{0, 1\}$)
- $d$: dimension of the transformed subspace ($d \leq m$)
- $A_p$: a predefined matrix ($A_p \in R^{d \times m}$)
- $K$: $K$-fold cross validation

**OUTPUT** - $A_{CSML}$: output transformation matrix ($A_{CSML} \in R^{d \times m}$)

1. $A_0 \leftarrow A_p$
2. $\alpha \leftarrow \frac{|Pos|}{|Neg|}$
3. **Repeat**
   (a) $min\_cve \leftarrow \infty$ // store minimum cross validation error
   (b) **For** each value of $\beta$ // coarse-to-fine strategy
      i. $A^* \leftarrow$ the matrix maximizing $f(A)$ given $(A_0, \alpha, \beta)$ evaluating on $S$
      ii. **if** $cve(T, A^*, K) < min\_cve$ **then** // Algorithm 2
         A. $min\_cve \leftarrow cve(T, A^*, K)$
         B. $A_{next} \leftarrow A^*$
   (c) $A_0 \leftarrow A_{next}$
4. **Until** convergence
5. $A_{CSML} \leftarrow A_0$
6. **Return** $A_{CSML}$

### 1.4 The algorithm and its complexity

The idea is to use cross validation to estimate the optimal values of $(\alpha, \beta)$. In this paper, $\alpha$ can be simply set to 1 and suitable $\beta$ can be found using coarse-to-fine search strategy. Coarse-to-fine means the range of searching area decreases over time. Algorithm 1 presents the proposed CSML method. It is easy to prove that when $\beta$ goes to $\infty$, the optimized matrix $A^*$ approaches the prior $A_0$. In other words, the performance of learned matrix $A_{CSML}$ is guaranteed to be as good as that of matrix $A_0$. In practice, however, the performance of matrix $A_{CSML}$ is significantly better in most cases (see section 3).

$f(A)$ is differentiable with regard to matrix A so we can optimize it using a gradient based optimizer such as delta-bar-delta or conjugate gradients. We used the Conjugate Gradient method. The gradient of $f(A)$ can be computed as follows:

**Algorithm 2** Cross validation error computation

**INPUT**

- $T = \{x_i, y_i, l_i\}_{i=1}^{t}$: a set of validation samples ($x_i, y_i \in R^m$, $l_i \in \{0, 1\}$)
- $A$: a linear transformation matrix ($A \in R^{d \times m}$)
- $K$: $K$-fold cross validation

**OUTPUT** - cross validation error

1. Transform all samples in $T$ using matrix $A$
2. Partition $T$ into $K$ equal-sized subsamples
3. $total\_error \leftarrow 0$
4. **For** $k = 1 \rightarrow K$
   // using subsample $k$ as testing data, the other $K-1$ subsamples as training data
   (a) $\theta \leftarrow$ the optimal threshold on training data
   (b) $test\_error \leftarrow$ error on testing data
   (c) $total\_error \leftarrow total\_error + test\_error$
5. **Return** $total\_error/K$

$$\frac{\partial f(A)}{\partial A} = \sum_{i \in Pos} \frac{\partial CS(x_i, y_i, A)}{\partial A} - \alpha \sum_{i \in Neg} \frac{\partial CS(x_i, y_i, A)}{\partial A} - 2\beta(A - A_0) \tag{1}$$

$$\begin{aligned} \frac{\partial CS(x_i, y_i, A)}{\partial A} &= \frac{\partial (\frac{x_i^T A^T A y_i}{\sqrt{x_i^T A^T A x_i}\sqrt{y_i^T A^T A y_i}})}{\partial A} \\ &= \frac{\partial (\frac{u(A)}{v(A)})}{\partial A} \\ &= \frac{1}{v(A)} \frac{\partial u(A)}{\partial A} - \frac{u(A)}{v(A)^2} \frac{\partial v(A)}{\partial A} \end{aligned} \tag{2}$$

where

$$\frac{\partial u(A)}{\partial A} = A(x_i y_i^T + y_i x_i^T) \tag{3}$$

$$\frac{\partial v(A)}{\partial A} = \frac{\sqrt{y_i^T A^T A y_i}}{\sqrt{x_i^T A^T A x_i}} A x_i x_i^T - \frac{\sqrt{x_i^T A^T A x_i}}{\sqrt{y_i^T A^T A y_i}} A y_i y_i^T \tag{4}$$

From Eq (1, 2, 3, 4), the complexity of computing $f(A)$'s gradient is $O(s \times d \times m)$. As a result, the complexity of CSML algorithm is $O(r \times b \times g \times s \times d \times m)$ where $r$ is the number of iterations used to optimize $A_0$ repeatedly (at line 3 in Algorithm 1), $b$ is the number of values of $\beta$ tested in cross validation process (at line 3$b$ in Algorithm 1), $g$ is the number of steps in the Conjugate Gradient method.

## 2 Application to Face Verification

In this section, we show how CSML can be applied to face verification on the LFW dataset in detail.

### 2.1 LFW dataset

The dataset contains more than 13,000 images of faces collected from the web. These images have a very large degree of variability in face pose, age, expression, race and illumination. There are two evaluation settings by the authors of the LFW: the restricted and the unrestricted setting. This paper considers restricted setting. Under this setting no identity information of the faces is given. The only information available to a face verification algorithm is a pair of input images and the algorithm is expected to determine whether the pair of images come from the same person. The performance of an algorithm is measured by a 10-fold cross validation procedure. See [7] for details.

There are three versions of the LFW available: original, funneled and aligned. In [14], Wolf et al. showed that the aligned version is better than funneled version at dealing with misalignment. Therefore, we are going to use the aligned version in all of our experiments.

### 2.2 Face verification pipeline

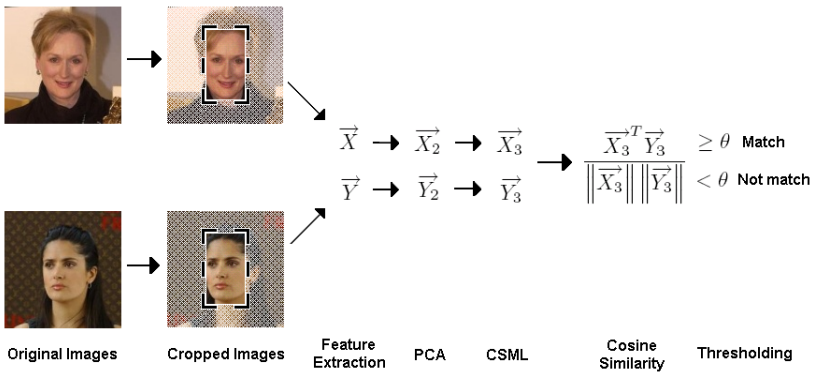


**Fig. 2.** Overview of Face verifiction process

The overview of our method is presented in Figure 2. First, two original images are cropped to smaller sizes. Next some feature extraction method is used to form feature vectors $(\overrightarrow{X}, \overrightarrow{Y})$ from the cropped images. These vectors are passed to PCA to get two dimension-reduced vectors $(\overrightarrow{X_2}, \overrightarrow{Y_2})$. Then CSML is used to transform $(\overrightarrow{X_2}, \overrightarrow{Y_2})$ to $(\overrightarrow{X_3}, \overrightarrow{Y_3})$ in the final subspace. Cosine similarity

between $\overrightarrow{X_3}$ and $\overrightarrow{Y_3}$ is the similarity score between two faces. Finally, this score is thresholded to determine whether two faces are the same or not. The optimal threshold $\theta$ is estimated from the training set. Specifically, $\theta$ is set so that False Acceptance Rate equals to False Rejection Rate. Each step will be discussed in detail.

**Preprocessing** The original size of each image is $250 \times 250$ pixels. At the preprocessing step, we simply crop the image to remove the background, leaving a $80 \times 150$ face image. The next step after preprocessing is to extract features from the image.

**Feature Extraction** To test the robustness of our method to different types of features, we carry out experiments on three facial descriptors: Intensity, Local Binary Patterns and Gabor Wavelets.

Intensity is the simplest feature extraction method. The feature vector is formed by concatenating all the pixels. The length of the feature vector is $12,000$ $(= 80 \times 150)$.

Local Binary Patterns (LBP) was first applied for Face Recognition in [15] with very promising results. In our experiments, the face is divided into non-overlapping $10 \times 10$ blocks and LBP histograms are extracted in all blocks to form the feature vector whose length is $7,080$ $(= 8 \times 15 \times 59)$.

Gabor Wavelets [16,17] with 5 scales and 8 orientations are convoluted at different pixels selected uniformly with the downsampling rate of $10 \times 10$. The length of the feature vector is $4,800$ $(= 5 \times 8 \times 8 \times 15)$ .

**Dimension Reduction** Before applying any learning method, we use PCA to reduce the dimension of the original feature vector to a more tractable number. A thousand faces from training data (different for each fold) are used to create the covariance matrix in PCA. We notice in our experiments that the specific value of the reduced dimension after applying PCA doesn't affect the accuracy very much as long as it is not too small.

**Feature Combination** We can further improve the accuracy by combining different types of features. Features can be combined at the feature extraction step [18,19] or at the verification step. Here we combine features at the verification step using SVM [20]. Applying CSML to each type of feature produces a similarity score. These scores form a vector which is passed to SVM for verification.

### 2.3 How to choose $A_0$ in CSML?

Because CSML improves the accuracy of $A_0$, it is a good idea to choose matrix $A_0$ which performs well by itself. There are published papers concluding that Whitened PCA (WPCA) with Cosine Similarity can achieve very good performance [3,21]. Therefore, we propose to use the whitening matrix as $A_0$. Since we

reduce the dimension from $m$ to $d$, the whitening matrix is in the rectangular form as follows:

$$A_{WPCA} = \begin{bmatrix} \lambda_1^{-\frac{1}{2}} & 0 & ... & 0 & 0\,0 \\ 0 & \lambda_2^{-\frac{1}{2}} & ... & 0 & 0\,0 \\ ... & ... & ... & ... & 0\,0 \\ 0 & 0 & ... & \lambda_d^{-\frac{1}{2}} & 0\,0 \end{bmatrix} \in R^{d \times m}$$

where $\lambda_1, \lambda_2, ..., \lambda_d$ are the first $d$ largest eigen-values of the covariance matrix computed in the PCA step.

To compare, we tried two different matrices: non-whitening PCA and Random Projection.

$$A_{PCA} = \begin{bmatrix} 1 & 0 & ... & 0 & 0\,0 \\ 0 & 1 & ... & 0 & 0\,0 \\ ... & ... & ... & ... & 0\,0 \\ 0 & 0 & ... & 1 & 0\,0 \end{bmatrix} \in R^{d \times m}$$

$$A_{RP} = random\,matrix \in R^{d \times m}$$

## 3 Experimental Results

To evaluate performance on View 2 of the LFW dataset, we used five of the nine training splits as training samples and the remaining four as validation samples in CSML algorithm (more about the LFW protocol in [7]). These validation samples are also used for training the SVM. All results presented here are produced using the parameters: $m = 500$ and $d = 200$ where $m$ is the dimension of the data after applying PCA and $d$ is the dimension of the data after applying CSML. In this section, we present the results of two experiments.

In this first experiment, we will show how much CSML improves over three cases of $A_0$: Random Projection, PCA, and Whitened PCA. We call the transformation matrices of these $A_{RP}$, $A_{PCA}$, and $A_{WPCA}$ respectively. Here we used the original intensity as the feature extraction method. As shown in table 1, $A_{CSML}$ consistently performs better than $A_0$ about $5 - 10\%$.

| | $A_{RP}$ | $A_{PCA}$ | $A_{WPCA}$ |
|---|---|---|---|
| $A_0$ | $0.5752 \pm 0.0057$ | $0.6762 \pm 0.0075$ | $0.7322 \pm 0.0037$ |
| $A_{CSML}$ | $0.673 \pm 0.0095$ | $0.7112 \pm 0.0083$ | $0.7865 \pm 0.0039$ |

**Table 1.** $A_{CSML}$ and $A_0$ performance comparison

In the second experiment, we will show how much CSML improves over cosine similarity in the original space and over Whitened PCA with three types of features: Intensity (IN), Local Binary Patterns (LBP), and Gabor Wavelets

(GABOR). Each type of feature is tested with the original feature vector or the square root of the feature vector [8,14,20].

| | | Cosine | WPCA | CSML |
|---|---|---|---|---|
| IN | original | $0.6567 \pm 0.0071$ | $0.7322 \pm 0.0037$ | $0.7865 \pm 0.0039$ |
| | sqrt | $0.6485 \pm 0.0088$ | $0.7243 \pm 0.0038$ | $0.7887 \pm 0.0052$ |
| LBP | original | $0.7027 \pm 0.0036$ | $0.7712 \pm 0.0044$ | $0.8295 \pm 0.0052$ |
| | sqrt | $0.6977 \pm 0.0047$ | $0.7937 \pm 0.0034$ | $\mathbf{0.8557 \pm 0.0052}$ |
| GABOR | original | $0.672 \pm 0.0053$ | $0.7558 \pm 0.0052$ | $0.8238 \pm 0.0021$ |
| | sqrt | $0.6942 \pm 0.0072$ | $0.7698 \pm 0.0056$ | $0.8358 \pm 0.0058$ |
| Feature Combination | | | | $\mathbf{0.88 \pm 0.0037}$ |

**Table 2.** The improvements of CSML over cosine similarity and WPCA

As shown in table 2, CSML improves about 5% over WPCA and about $10-15\%$ over cosine similarity. LBP seems to perform better than Intensity and Gabor Wavelets. Using square root of the feature vector improves the accuracy about $2-3\%$ in most cases. The highest accuracy we can get from a single type of feature is $0.8557 \pm 0.0052$ using CSML with the square root of the LBP feature. The accuracy we can get by combining 6 scores corresponding to 6 different features (in the rightmost column in table 2) is $0.88 \pm 0.0038$. This is better than the current state of the art result reported in [14]. For comparison purpose, the ROC curves of our method and others are depicted in Figure 3. Complete benchmark results can be found on the LFW website [22].

## 4 Conclusion

We have introduced a novel method for learning a distance metric based on cosine similarity. The use of cosine similarity allows us to form a simple but effective objective function, which leads to a fast gradient-based optimization algorithm. Another important property of our method is that in theory the learned matrix cannot perform worse than the regularized matrix. In practice, it performs considerably better in most cases. We tested our method on the LFW dataset and achieved highest accuracy in the literature. Although initially CSML was designed for face verification, it has a wide range of applications, which we plan to explore in future work.

## References

1. Phillips, P., Wechsler, H., Huang, J., Rauss, P.: The FERET database and evaluation procedure for face-recognition algorithms. Image and Vision Computing **16** (1998) 295–306

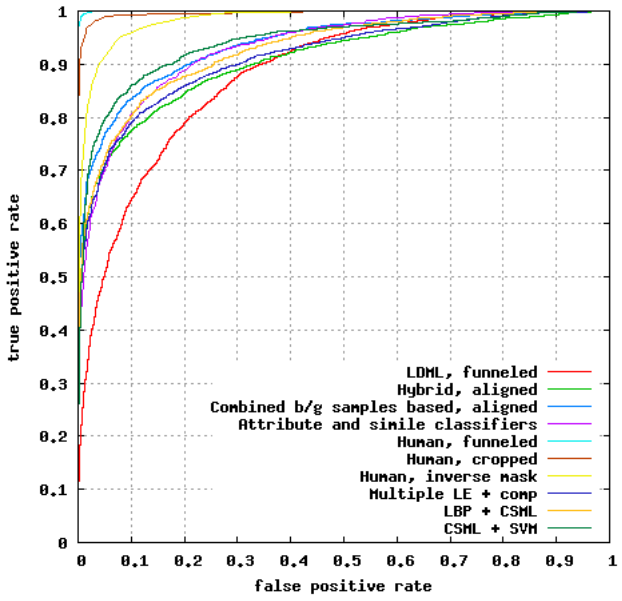


**Fig. 3.** ROC curves averaged over 10 folds of View 2


2. Shan, S., Zhang, W., Su, Y., Chen, X., Gao, W., FRJDL, I., CAS, B.: Ensemble of Piecewise FDA Based on Spatial Histograms of Local (Gabor) Binary Patterns for Face Recognition. In: Proceedings of the 18th international conference on pattern recognition. (2006) 606–609
3. Hieu, N., Bai, L., Shen, L.: Local gabor binary pattern whitened pca: A novel approach for face recognition from single image per person. In: The 3rd IAPR/IEEE International Conference on Biometrics, 2009. Proceedings. (2009)
4. Shen, L., Bai, L.: MutualBoost learning for selecting Gabor features for face recognition. Pattern Recognition Letters **27** (2006) 1758–1767
5. Shen, L., Bai, L., Fairhurst, M.: Gabor wavelets and general discriminant analysis for face identification and verification. Image and Vision Computing **27** (2006) 1758–1767
6. Nguyen, H.V., Bai, L.: Compact binary patterns (cbp) with multiple patch classifiers for fast and accurate face recognition. In: CompIMAGE. (2010) 187–198
7. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst (2007)
8. Guillaumin, M., Verbeek, J., Schmid, C.: Is that you? metric learning approaches for face identification. In: International Conference on Computer Vision. (2009) 498–505
9. Taigman, Y., Wolf, L., Hassner, T.: Multiple one-shots for utilizing class label information. In: The British Machine Vision Conference (BMVC). (2009)
10. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighborhood component analysis. (In: NIPS)

11. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems 15. Volume 15. (2003) 505–512
12. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. Advances in Neural Information Processing Systems **18** (2006) 1473–1480
13. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML '07: Proceedings of the 24th international conference on Machine learning, New York, NY, USA, ACM (2007) 209–216
14. Wolf, L., Hassner, T., Taigman, Y.: Similarity scores based on background samples. In: ACCV (2). (2009) 88–97
15. Ahonen, T., Hadid, A., Pietikainen, M.: Face Recognition with Local Binary Patterns. LECTURE NOTES IN COMPUTER SCIENCE (2004) 469–481
16. Daugman, J.: Complete Discrete 2D Gabor Transforms by Neural Networks for Image Analysis and Compression. IEEE Trans. Acoust.Speech Signal Process **36** (1988)
17. Shan, S., Gao, W., Chang, Y., Cao, B., Yang, P.: Review the strength of Gabor features for face recognition from the angle of its robustness to mis-alignment. In: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Volume 1. (2004)
18. Tan, X., Triggs, B., Vision, M.: Fusing Gabor and LBP Feature Sets for Kernel-Based Face Recognition. LECTURE NOTES IN COMPUTER SCIENCE **4778** (2007) 235
19. Zhang, W., Shan, S., Gao, W., Chen, X., Zhang, H.: Local Gabor Binary Pattern Histogram Sequence (LGBPHS): A Novel Non-Statistical Model for Face Representation and Recognition. In: Proc. ICCV. (2005) 786–791
20. Wolf, L., Hassner, T., Taigman, Y.: Descriptor based methods in the wild. In: Real-Life Images workshop at the European Conference on Computer Vision (ECCV). (2008)
21. Deng, W., Hu, J., Guo, J.: Gabor-Eigen-Whiten-Cosine: A Robust Scheme for Face Recognition. LECTURE NOTES IN COMPUTER SCIENCE **3723** (2005) 336
22. http://vis-www.cs.umass.edu/lfw/results.html: (LFW benchmark results)