Fast and Robust Probabilistic Inference of Iris Mask

Yung-hui Li and Marios Savvides
Carnegie Mellon University, Pittsburgh, PA, U.S.A.

ABSTRACT

Iris masks are essential in iris recognition. The purpose of having a good iris mask is to indicate which part of iris texture map is useful and which part is occluded or contains noisy artifacts such as eyelashes, eyelids and specular reflections. The accuracy of the iris mask is extremely important. The performance of the iris recognition system will decrease dramatically when iris mask is inaccurate, even when the best recognition algorithm is used. Traditionally, people used naive rule-based algorithms to estimate iris masks from the iris texture map. But the accuracy of the iris mask generated in this way is questionable. In this paper, we propose a probabilistic and learning-based method to automatically estimate iris mask from iris texture map. The features used in this method are very simple, yet the resulting estimated iris mask is significantly more accurate than the rule-based methods. We also demonstrate the effectiveness of the algorithm by performing iris recognition based on masks estimated by different algorithms. Experimental results show the masks estimated by the proposed algorithm help to increase the iris recognition rate on NIST Iris Challenge Evaluation (ICE) database.

Keywords: Iris recognition, iris mask estimation, probabilistic inference, Figueiredo–Jain GMM, Gaussian Mixture Models, machine learning, pattern recognition, biometrics

1. INTRODUCTION

Most iris recognition systems consist of four stages: image acquisition, iris segmentation, iris normalization and recognition. In most cases, in the iris normalization stage, iris images are transformed from the Cartesian coordinate system to the polar coordinate system as suggested by Daugman.[1] There are two main advantages of this coordinate transformation. First, it normalizes the nonlinear texture variation caused by changes of environmental illumination that leads to pupil dilation and contraction. Without doing this transformation, it may be hard to compare two irises when the size of the pupil is different. The second advantage of coordinate transformation is that it translates the rotational shift in the Cartesian coordinate to a pure translational shift in horizontal direction in the polar domain. It also simplifies the problem in the matching stage because it is much more difficult and error-prone to perform pattern matching with rotational variation than translational variation. Because of these two advantages, most iris recognition systems adopt this type of iris texture normalization in their implementation.

In most cases, after transforming the iris texture from the Cartesian coordinate to the polar coordinate, one has to create a mask for the iris map in the polar coordinate. The goal of this mask is to indicate which part in the iris map is truly iris texture, and which part is noise. The sources of the occlusions/artifacts of the iris map may come from eyelids, eyelashes, or specular reflections. Fig. 1 shows example images of an iris texture map in polar domain and its accurate mask (created by manual labor). Note that in the iris texture, the noisy regions consist of texture created by eyelids, eyelashes, and specular reflections. All of these noises have to be indicated in the mask in order for the high performance of iris recognition.

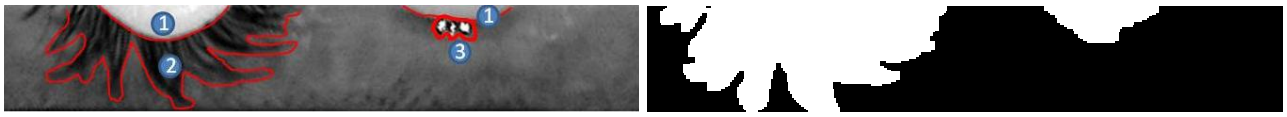


Figure 1. Normalized iris texture map (left picture) and its accurate mask (right picture), with white color indicates occluded area. In this iris map, there are noises caused by (1) eyelids, (2) eyelashes, and (3) specular reflections, as indicated in the picture. All of these artifacts have to be indicated in the mask in order to achieve high recognition performance.

The accuracy of the iris mask has a great impact to the recognition accuracy of iris recognition system. Traditionally, the main focus of research around iris recognition addresses the power of matching algorithm and feature extraction. Most researchers emphasize the novelty and effectiveness of their iris feature extraction and matching algorithm. While these remain an important issue and a major role in iris recognition, we have found that the accuracy of the iris mask contributes much more than what researchers thought in the past. Accurate iris masks, combined with good features and effective recognition schemes, make the iris recognition system more successful. However, if the iris mask is inaccurate, no matter how good the feature extraction and recognition algorithm is, the overall recognition rate will decrease dramatically due to mis-matching insignificant occluded iris regions. This has been proved in our experiments, presented in later section of this paper.

In this paper, we propose a probabilistic, learning-based algorithm to estimate iris mask from the original normalized iris texture map. It does not require any other information from eye images in the original Cartesian coordinate. Another advantage of our proposed algorithm is its efficiency. Since we are modeling probability distribution with mixtures of Gaussian, and the formula of Gaussian distribution is easy to evaluate, our proposed algorithm can automatically generate an iris mask very efficiently in a short time. These advantages make our proposed algorithm not only theoretically appealing but also practically efficient.

The rest of the paper is organized as follows. We will review previous work about automatic iris generation in Section 2 and explain our proposed algorithm in Section 3. Our experimental results will be shown in Section 4. In Section 5, we will present a discussion about the results we get. Finally, we conclude the contribution of our proposed algorithm and also give a suggestion about our future work in Section 6.

2. PREVIOUS WORK

Daugman's research is one of the earliest in iris recognition.[1] He described details about the normalization scheme for iris texture, and called it "Doubly Dimensionless Projected Polar Coordinate System". However, when considering which part of the iris texture in the polar coordinate system is the authentic iris, he did not address this issue significantly and proposed a simple method which assumed the top part and the $45°$ notch at the 6 o'clock position is occluded, for every iris image.

Ma, et al. proposed a full framework for the iris recognition system as well.[2] Although it addressed many issues in iris recognition, it does not mention any specific solution for occlusion detection. The only thing related to occlusion is that they discard the lower part of the normalized iris texture and focus only on the more discriminative regions. This scheme is equivalent to automatically assuming the lower part of iris texture is occluded. A similar assumption was proposed in the work of Tisse, et al.[3]

Daugman, in his later work, proposed a more sophisticated algorithm for finding the occluded region in an iris.[4] He proposed to locate the boundary of eyelids at the segmentation stage. By replacing the circular integration operator (for finding the iris and pupil boundary) with spline parameters, one can approximately locate the eyelid boundaries.

Kong and Zhang proposed a model for detecting the eyelashes and specular reflections.[5] They proposed to use Gabor filters to detect separable eyelashes and use variance of intensity in the local window to locate clusters of multiple eyelashes. After that, the connective criterion is enforced to enhance the robustness of the algorithm. For strong specular reflections, they just used a hard threshold on pixel intensity to identify it. For weak specular reflection, they used the mean and standard deviation of the local window as a adaptive threshold for pixel intensity to classify.

Zou et al. proposed a procedure for iris occlusion detection.[6] It consists of four stages. First, it detects horizontal edges, since eyelids are more likely to be horizontal. Second, it performs the morphological operation on those edges to enhance them. Third, it uses the segmentation result to locate valid edges. Finally, it uses connectivity information to refine the mask. Although this method looks simple and easy to implement, it may not work well for eyelashes because they are mostly vertical. Also, the edge detection based algorithm may incorrectly classify some part of iris texture into noisy region when the contrast within iris texture is large.

3. PROPOSED METHOD

All of the previous works have some limitations. The methods described by Daugman,[1] Ma, et al.,[2] or Tisse, et al.[3] appear naive and cannot handle iris images taken in real-time. The method described by Kong and Zhang[5] is basically a rule-based approach. The rule-based approach may work well for a specific settings or particular environment but may not work when the environment is changed. Parameter tuning for the rule-based approach is also important. These issues make the rule-based approach flimsy and not flexible.

To overcome the shortcomings of all the methods mentioned above, we would like to propose a novel method for automatic iris mask generation. We would like this method to be flexible in terms of that it will not need information from previous stage, which means without knowing anything from the eye image in Cartesian coordinate, we would still be able to get accurate iris masks. As we have mentioned, currently the most sophisticated algorithm require information about eyelid boundary from images of Cartesian coordinate in order to generate masks for images in polar domain.[4] We would like propose a better algorithm that does not need to do that. On the contrary, our proposed algorithm can estimate the occluded region just based on the image in polar domain. In this way, our method would facilitate the whole iris recognition system to be more modular and each stage would be more independent to each other.

We can treat the problem of iris occlusion estimation as a pixel-wise, two-class classification problem. Given an iris image I in polar domain, we would like to get a mask image M such that $M(x, y) = 0$ if and only if $I(x, y)$ is true iris texture, otherwise $M(x, y) = 1$. Therefore, the problem of iris occlusion estimation is equal to perform a two-class classification problem (iris texture vs. occlusion) on each pixel. Assuming the size of iris image I is 30x180, then for one input iris image, we have to do such the pixel-wise classification 5400 times in order to generate the full mask for it.

There are many different machine learning algorithms that can solve two-class classification problems. We propose to use the Gaussian Mixture Modeling (GMM) to model the posterior probability distribution of both iris texture and occlusion classes, and then the classification decision can be made by comparing the posterior probability of these two classes. GMMs have been widely used in all kinds of problems in machine learning and pattern recognition, including speech processing,[7] human skin detection,[8] real-time tracking,[9] hazardous chemical agents detection[10] and bearing damage detection for induction motor.[11] The advantage of GMM is its modeling ability. As long as the number of Gaussian distributions is large enough, GMM can virtually model any shape of distribution. Another advantage of GMM is its mathematical equation is easy to evaluate. Therefore, the classification speed is high during testing stage. This is an important consideration when dealing with problems like automatic mask generation because we have to perform pixel-wise classification for every iris image. If the classification speed is not high enough, it will not have the practical value in a real-time iris recognition system.

3.1 Probability density function for GMM

Let us review the basic mathematical foundations for GMM. The Gaussian distribution of a D dimensional random variable X which has a value x is represented by (1)

$$X \sim \mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]} \tag{1}$$

where μ is the mean vector and Σ is the covariance matrix of the Gaussian distributed random variable X.

The probability density function of GMM can be defined as a weighted sum of multiple Gaussian distributions, as shown in (2)

$$p(x;\ \theta) = \sum_{c=1}^{C} \alpha_c \mathcal{N}(x;\ \mu_c; \Sigma_c) \tag{2}$$

where α_c is the priori probability that the random variable x is generated by the cth Gaussian mixture, and they should satisfy

$$0 \leq \alpha_c \leq 1 \quad and \quad \sum_{c=1}^{C} \alpha_c = 1 \tag{3}$$

Based on (1)-(3), the probability density function for a Gaussian mixture model can be completely defined by a parameter list as shown in (4)

$$\theta = \{\alpha_1,\ \mu_1,\ \Sigma_1, ...,\ \alpha_C,\ \mu_C,\ \Sigma_C, \} \tag{4}$$

We can calculate the number of free parameters for a GMM, given the random variable X is of dimension D. For each μ_i, there are D free parameters; for each Σ_i, since it should be symmetric in nature, the number of free parameters is $\left(D^2 - D\right)/2 + D = \frac{1}{2}(D^2 + D)$. For each α_i, we have one free parameter. Combining these calculation and (3), the total number of free parameters is $C(\frac{1}{2}D^2 + \frac{3}{2}D) + C - 1$.

3.2 MLE and MAP estimation for model parameters

The process of training GMM is to estimate the parameter list θ, given the observation X. Suppose we have a set of independently, identically distributed samples X=$\{x_1,\ x_2,\ ...,\ x_N\}$ drawn from the same distribution described by GMM probability density function $p(x;\ \theta)$. The likelihood function can be defined as (5)

$$\mathcal{L}(\mathrm{X};\ \theta) = \prod_{n=1}^{N} p(x;\ \theta) \tag{5}$$

It tells the probability that the series of observation X is generated by distribution governed by θ. The goal of parameter estimation is to find the optimal parameter $\hat{\theta}$ that maximize the probability:

$$\hat{\theta} = arg\ \max_{\theta} \mathcal{L}(\mathrm{X};\ \theta) \tag{6}$$

$\mathcal{L}(\mathrm{X};\ \theta)$ in (5) and (6) contains many multiplication operations. We can speed-up the operation by taking the logarithm function on both side. The log-likelihood function can be described as (7):

$$L(\mathrm{X};\ \theta) = \ln \mathcal{L}(\mathrm{X};\ \theta) = \sum_{n=1}^{N} \ln p(x_n;\ \theta) \tag{7}$$

Since the logarithm function is monotonically increasing, we can substitute $\mathcal{L}(\mathrm{X};\ \theta)$with L(X; θ) in (6) and get the same answer.

Parameter estimation by (6) is called Maximum-Likelihood Estimation (MLE). Sometimes maximum a posteriori (MAP) estimation is used instead of MLE:

$$\hat{\theta}_{MAP} = arg\ \max \{\ln \mathcal{L}(\mathrm{X};\ \theta) + \ln \mathcal{L}(\theta)\} \tag{8}$$

3.3 EM algorithm

In literature, Expectation-Maximization (EM) algorithm is used to find the parameter $\hat{\theta}_{MLE}$ or $\hat{\theta}_{MAP}$. EM is an iterative procedure to estimate parameters when part of data is missing. In the case of GMM training problem, we need to estimate multiple parameters at the same time, including the mean and covariance matrix for each Gaussian mixture and which training sample belongs to which Gaussian mixture. EM can optimize all of these unknown parameters and converges to a local maximum of $\hat{\theta}$. There are two main steps in EM algorithm, described below.

3.3.1 E-step

The E-step in EM is to estimate the expectation of the likelihood of the observed data, assuming that we know the optimal model parameters. In other words, it is to evaluate the expectation function $Q(\theta;\, \theta^i)$ in (9)

$$Q(\theta;\, \theta^i) = E_{\mathrm{Y}}\left[\ln \mathcal{L}(\mathrm{X},\, \mathrm{Y};\, \theta)|\mathrm{X};\, \theta^i\right] \tag{9}$$

where Y is the unknown feature, in our case, the labels that indicate which Gaussian mixture each sample belongs to.

3.3.2 M-step

The M-step in EM is to search the parameter space to find the optimal parameter $\hat{\theta}$ that can maximize the likelihood function defined in (9):

$$\theta^{i+1} = arg\ \max_{\theta} Q(\theta;\, \theta^i) \tag{10}$$

The two steps will be executed repeatedly until θ converges to a local maxima. Usually we can define a convergence criteria as a threshold on the difference of the likelihood between two iterations:

$$Q(\theta^{i+1};\, \theta^i) - Q(\theta^i;\, \theta^{i-1}) \leq T \tag{11}$$

Alternatively, convergence criteria can be defined as a threshold on the distance in parameter space:

$$||\theta^{i+1} - \theta^i|| \leq \epsilon \tag{12}$$

3.4 Figueiredo–Jain's extension for GMM training

EM-based parameter estimation for GMM training has a few drawbacks. First, the number of Gaussian mixtures has to be manually chosen. Therefore, a wrong estimation for the number of Gaussian mixtures will definitely hurt the accuracy of the trained GMM. Second, the initialization of the mean of each Gaussian mixture is also crucial. Similar to K-means algorithm, if the initial position of the means of Gaussian is placed inadequately, EM algorithm may be not able to converge to the true location of the mean of each Gaussian.

Because of these two major drawbacks in EM training algorithm, we would like to use alternative training method. Figueiredo and Jain proposed an unsupervised learning method for GMM.[12] This method can estimate the number of Gaussian mixtures without human intervention, and can avoid the boundary of the parameter space during the converging stage. The basic idea of Figueiredo–Jain's extension for GMM training (FJ algorithm) is that it dynamically adjusts the number of Gaussian distributions by eliminating Gaussians that are not supported by the observation. Also, during the mixture elimination process, it chooses to eliminate the Gaussians that are becoming singular. By doing this, it can avoid converging to small Gaussians on the boundary of parameter space.

FJ algorithm uses the idea of Minimum Descriptive Length (MDL) and applies it to mixture model training. It is equivalent to using the objective function in (13)

$$\begin{aligned}\Lambda(\theta,\, \mathrm{X}) \quad = \quad & \frac{V}{2}\sum_{\alpha_c>0} \ln\left(\frac{N\alpha_c}{12}\right) + \frac{C_{nz}}{2}\ln\frac{N}{12} \\ & + \frac{C_{nz}\,(V+1)}{2} - \ln \mathcal{L}(\mathrm{X},\, \theta)\end{aligned} \tag{13}$$

where N is the number of training points, V is the number of free parameters of the GMM, C_{nz} is the number of Gaussian mixtures that have a nonzero weight ($\alpha_c > 0$), θ is defined as in (4), and the last term is defined in (7).

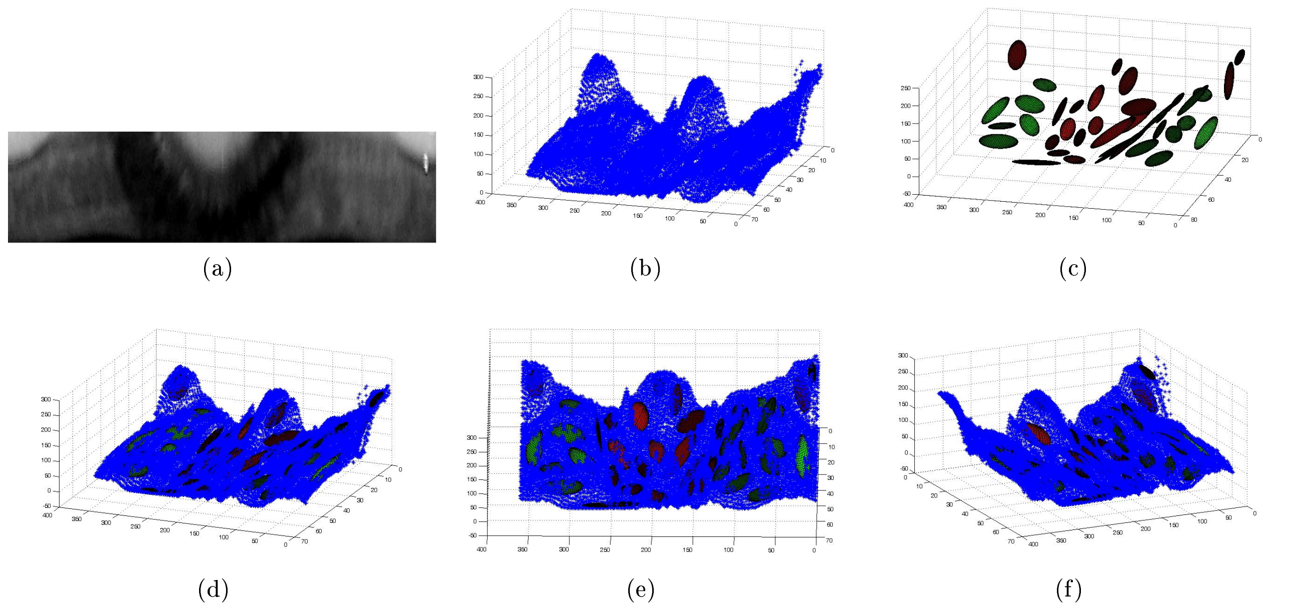


Figure 2. Visualization of GMM trained on single iris texture. (a) Example image for training; (b) Example iris texture viewed in 3D, where z coordinate is the pixel intensity value; (c) 3D GMMs trained with FJ algorithm. Mixtures with red color represent GMMs for occlusion; mixtures with green color represent GMMs for authentic iris texture; (d)-(f) Plotting original iris texture data together with trained GMM, in 3D view, viewed from 3 different angles. From (d)-(f), We can see that trained GMMs fit the training data very well.

By using (13) as new objective function, the formula for estimating the prior distribution of the Gaussian mixture in FJ algorithm becomes

$$\alpha_c^{i+1} = \frac{\max\left\{0, \left(\sum_{n=1}^{N} w_{n,c}\right) - \frac{V}{2}\right\}}{\sum_{j=1}^{C} \max\left\{0, \left(\sum_{n=1}^{N} w_{n,c}\right) - \frac{V}{2}\right\}} \tag{14}$$

where $w_{n,c}$ is the probability that the n^{th} observation is generated from the c^{th} Gaussian mixture, defined as

$$w_{n,c} = \frac{\alpha_c^i p\left(x_n|c;\, \theta^i\right)}{\sum_{j=1}^{C} \alpha_c^i p\left(x_n|c;\, \theta^i\right)} \tag{15}$$

The formula for estimating parameter μ_c and Σ_c is the same as in the traditional EM algorithm:

$$\mu_c^{i+1} = \frac{\sum_{n=1}^{N} x_n w_{n,c}}{\sum_{n=1}^{N} w_{n,c}} \tag{16}$$

$$\Sigma_c^{i+1} = \frac{\sum_{n=1}^{N} w_{n,c}\left(x_n - \mu_c^{i+1}\right)\left(x_n - \mu_c^{i+1}\right)^T}{\sum_{n=1}^{N} w_{n,c}} \tag{17}$$

We will not repeat too many details about FJ algorithm. Interested readers should refer to Figueiredo and Jain's work.[12]

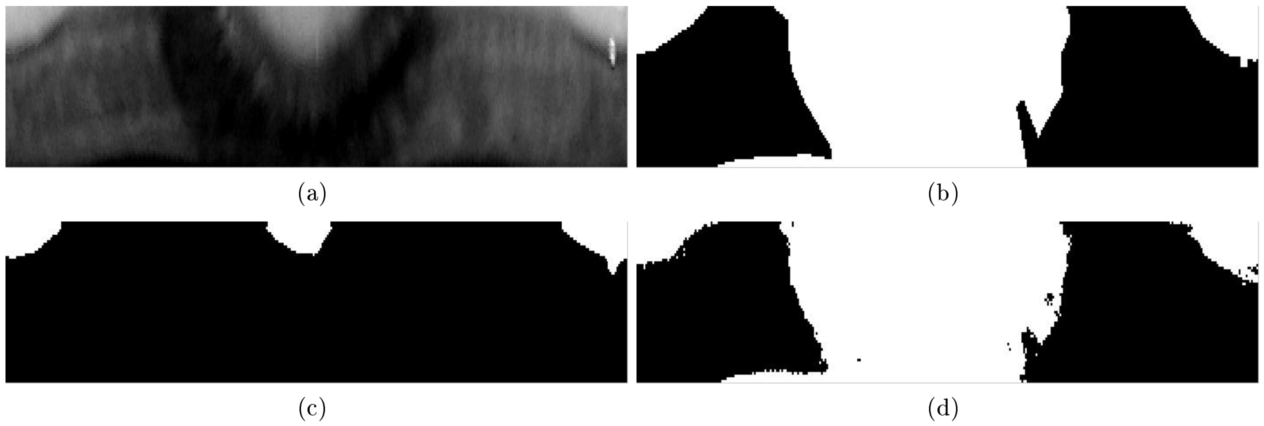


Figure 3. Comparison of the iris mask estimated by different algorithms. (a) the input iris texture image in polar coordinate; (b) the "perfect" mask, generated by manual labor; (c) the mask generated by a rule-based method; (d) the mask generated by FJ-GMM algorithm, with IMS feature set.

4. EXPERIMENTS AND RESULTS

4.1 GMM trained on single image

First, we use only one iris image to train GMM models. The training image we used is image 245241.tiff from ICE2 database. After manually segmenting the iris and performing iris normalization, we get Fig. 2(a). Now, if we think the intensity value of every pixel as the Z coordinate of that pixel in three-dimensional space, we can plot the iris texture map in Fig. 2(a) in 3D perspective, as shown in Fig. 2(b). We also manually created a mask for Fig. 2(a) to indicate which part is the authentic iris texture and which part is the occlusion, as shown in Fig. 3(b). By using 3D coordinates of each pixel as the feature vector for that point, and using a manually created mask as the class label set for each point on the image, we trained two GMM models; the first one is to model the distribution of iris texture, and the second is the occlusion regions. We plot the trained GMMs in 3D space in Fig. 2(c). In Fig. 2(c), GMMs with the red color represent the GMMs trained for occlusion and the GMMs with the green color are models for iris texture. The brighter the color, the higher the prior probability it has among all GMMs.

From Fig. 2(d)-(f), we can see that the trained GMMs fit very well with the original points in three-dimensional space. The red GMMs fits to the noisy part while the green GMMs fit the authentic iris texture. It shows the proposed algorithm works for this single image case.

If we use the Fig. 2(a) as our test image and use the trained GMM to perform classification for every pixel on this image, and plot the results of the classification back into a two dimensional matrix, with 0 indicating the authentic iris and 1 indicating occlusion region, we can visualize the mask generated by the GMM. We show the comparison of the iris mask estimated by different algorithms in Fig. 3. The input iris texture image in polar coordinate is shown in Fig. 3(a), while the ground truth of the mask shown in Fig. 3(b), which is manually labeled. The mask generated by proposed method is shown in Fig. 3(d), compared with another mask generated by a rule-based method, shown in Fig. 3(c). As we can see, the result generated by proposed algorithm is much better than what we can get by the rule-based method.

4.2 Automatic iris mask generation on ICE database

We also performed a large-scale experiment on automatic mask generation using the proposed method, and compared the results with masks generated by the other methods. We used the following mask to compare our method: masks that are created by manual labor, masks that are estimated by a rule-based method, and masks that are estimated by FLDA method. The rule-based method we used here is the one we used in both the previous and next sections. It is similar to the method described in Kong and Zhang's work.[5] Basically it detects whether there is a strong variance of pixel intensity on a local window and uses it as a feature to perform classification. It can be illustrated in four steps:

1. Normalize the image so that the energy of pixel intensity sums up to one.
2. Compute the mean value of the local 5x5 window centered at each pixel. It is the mean image.
3. Compute the global mean and standard deviation of all pixel intensity of the mean image.
4. For every pixel on the mean image, if the difference between its intensity value and global mean is less than twice the global standard deviation, classify it as iris texture. Otherwise, classify it as an occluded region.

The Fisher-Linear Discriminant Analysis (FLDA)-based occlusion detection method we used is described in Jason Thornton's PhD Thesis.[13]

For the proposed algorithm, we would like to try to experiment with all kinds of different features and see which feature is really discriminative for the goal of the occlusion estimation. The feature we would like to try includes:

- X, Y coordinate of the location of the pixel and pixel intensity
- Mean and standard deviation in a local 3x3 neighborhood
- Response intensity after the image is filtered by Sobel edge filter
- Response intensity after the image is filtered by Gaussian
- Response intensity after the image is filtered by Laplacian of Gaussian (LOG)
- Response intensity after the image is filtered by Gabor filter
- Response intensity after the image is filtered by first-order and second-order Haar wavelet

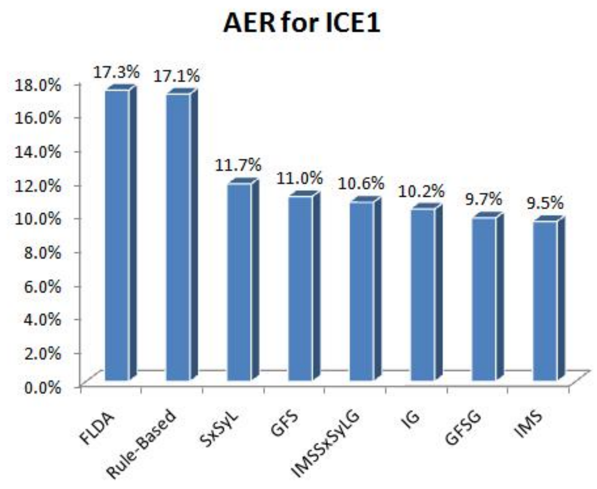
We give a code name to each experiment in order to better distinguish what we tried in each experiment. The code name and the features are listed below.

1. IMS: Intensity of the pixel, Mean and Standard deviation within 3x3 neighborhood.
2. SxSyL: Response intensity after the image is filtered by Sobel edge filter (both horizontally and vertically), and by Laplacian of Gaussian
3. IG: Intensity of the pixel and response intensity after the image is filtered by Gabor filter
4. IMSSxSyLG: Combination of all of the three above
5. GFS: Response intensity after the image is filtered by Gaussian, and the first-order and second-order Haar wavelet
6. GFSG: Similar to GFS, plus the response intensity after the image is filtered by Gabor filter

Note that all of the above experiments included features of the (X,Y) coordinate of the pixel location.

We performed our experiments on NIST ICE database.[14] In ICE database, there are two subsets: ICE1 and ICE2. We run experiments on both of them. We performed segmentation and generated masks for every image in ICE manually to get the ground truth for every mask. For each iris class in ICE, we picked one image as training data and left all the other images as test data. For each testing image, we computed the average error rate (AER) of the masks estimated by different algorithms. AER can be computed as:

$$AER = \frac{\sum_{i=1}^{N} e_i}{N} \tag{18}$$



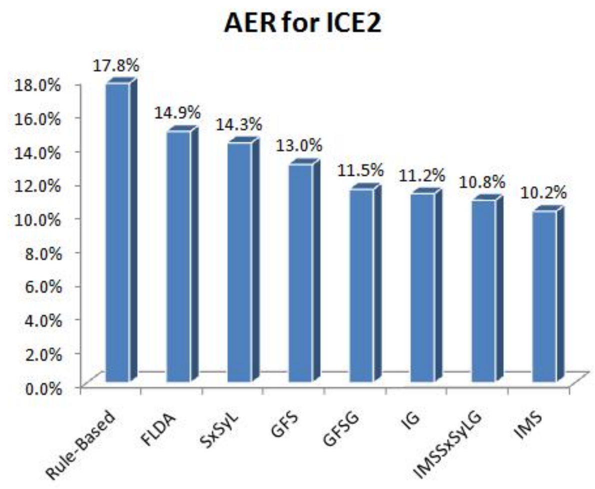


Figure 4. The Average Error Rate for the mask generated by different algorithms.

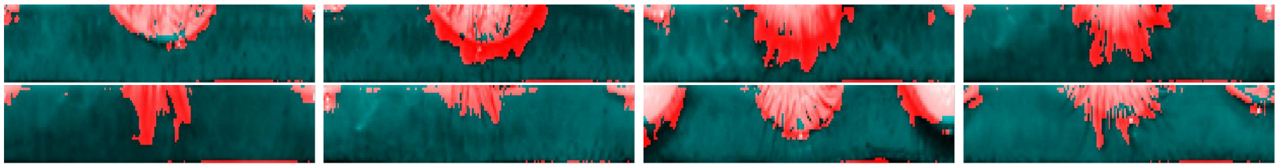


Figure 5. A few example mask created by proposed method, with GFSG feature set. Red color denotes the detected occlusion, dark green region denotes the detected iris region.

$$e_i = \frac{NP(mask_{alg} \otimes mask_{gt})}{W \times H} \tag{19}$$

where (W, H) is the size of the mask, $mask_{alg}$ and $mask_{gt}$ is the mask generated by specified algorithm and human labor, respectively; N is the total number of testing images; $\otimes$ is XOR operator that can be used to compute the difference between two images; $NP()$ is the function that counts the number of pixels in the images which are not zeros. The results are shown in Fig. 4. A few examples of iris mask images created by GFSG method are shown in Fig. 5.

As stated in Section 3, the efficiency of the proposed algorithm is one of its advantages. Therefore, we also measured how much time it takes for each algorithm to create one iris mask. We test all of our proposed algorithms as well as the baseline algorithms on the same platform, which has dual core AMD opteron processor. The speed of CPU is 2.6GHz. All the methods are implemented with Matlab codes, running on Linux server. The results are plotted in Fig 6.

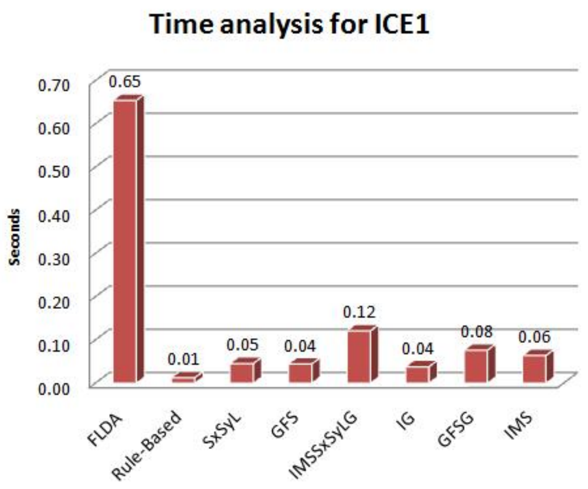
4.3 Iris recognition performance based on different masks

We also measured the iris recognition performance based on different masks. The iris feature extraction and matching algorithm we used in this experiment was Libor Masek's Matlab implementation of Daugman's algorithm, which is publicly available.[15] We performed iris recognition experiment on the ICE database, with manually created masks, and masks estimated by the two baseline algorithms, as well as masks estimated by proposed algorithms. The results are plotted in the format of ROC curves and shown in Fig. 7.

5. DISCUSSION

5.1 Average error rate for occlusion estimation

First, in terms of the accuracy of generated iris mask, Fig. 4 shows that in both ICE1 and ICE2 database, our proposed method (no matter which feature sets we use) is better than rule-based method and FLDA-based



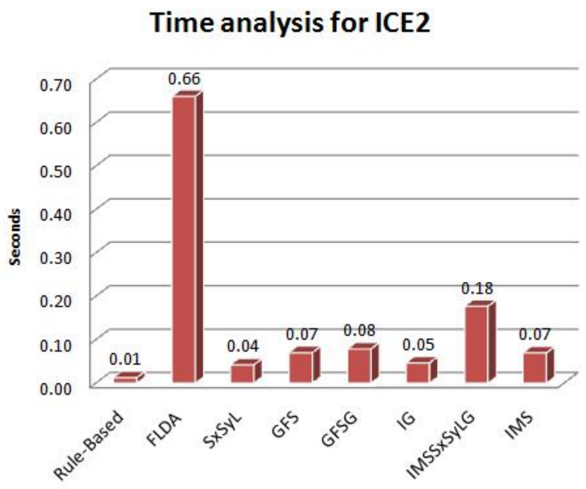
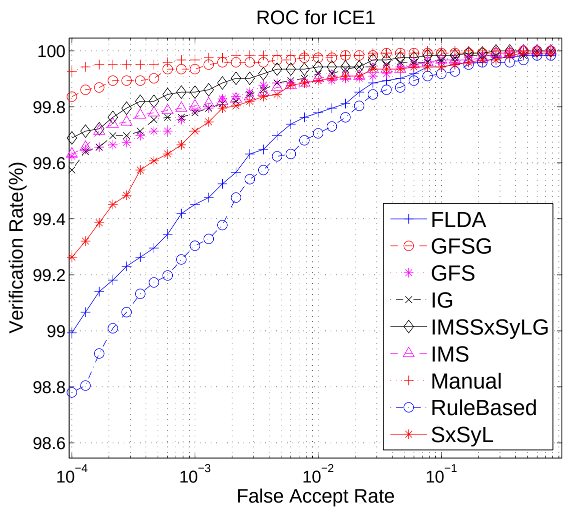
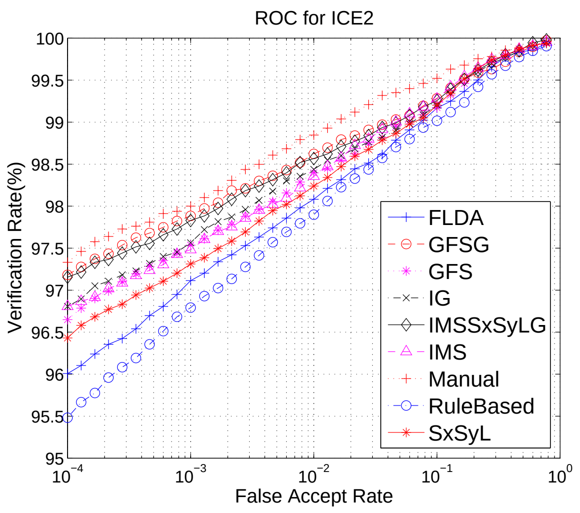


Figure 6. Time analysis: how much time it takes for each algorithm to create one iris mask



(a)



(b)

Figure 7. ROC curves of recognition performance, based on masks generated from experiment configurations. (a) ROC for ICE1 (right eyes) (b) ROC for ICE2 (left eyes)

method. This result proves our proposed method is useful and is able to create iris masks which are much more similar to manually created ones.

Considering the question of which feature set is more suitable for the goal of creating accurate iris masks, we can see that in both ICE1 and ICE2 experiments, IMS gives the best result. This means we can simply use pixel intensity, mean and standard deviation to be our features to train GMM and create a very accurate iris mask learning machine. Other features are also good, though not as good as IMS, in learning the characteristic of the occlusion region. One thing that is opposite to our intuition is that we may think the performance of IMSSxSyLG should be at least as good as IMS, if not better. But experimental result tells us this intuition is wrong. It tells us that in the problem of learning the texture representation, it is not true that the more features the better result. Only discriminative feature can give us best results. Redundant feature will not improve but only hurt our classifier.

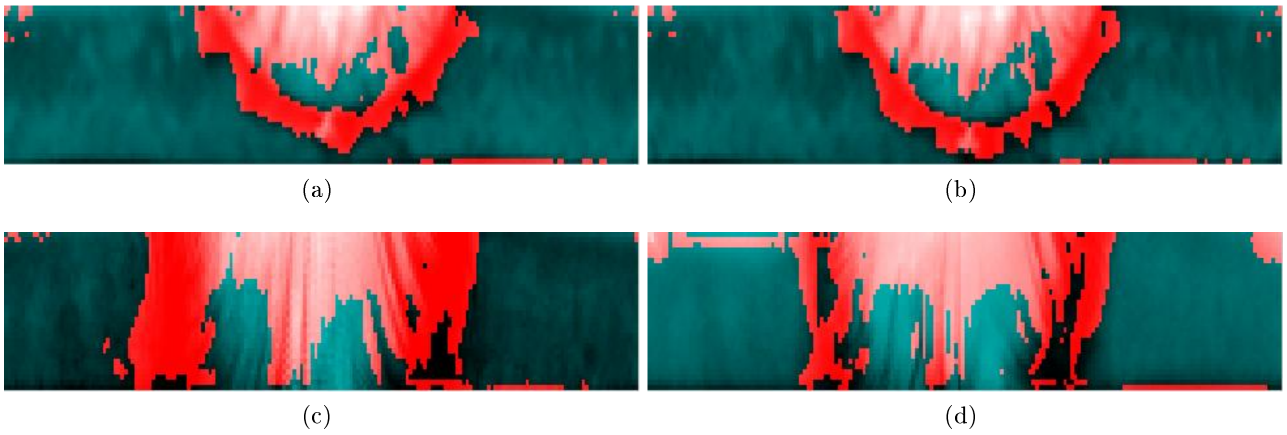


Figure 8. Examples of incorrect iris masks created by GFSG feature set. (a) and (b) are images from class 4 in ICE2. (c) and (d) are images from class 6 in ICE2.

5.2 Time analysis

As for the efficiency perspective, from Fig. 6, we can see that FLDA-based algorithm takes much longer time than all of the others, making it a very bad choice for the purpose of occlusion estimation, both in terms of accuracy and efficiency. Second, the rule-based method is most efficient algorithm. However, the fact that the accuracy of the created iris mask is bad makes it a sub-optimal algorithm.

All of our proposed algorithm falls within the bounds within FLDA and rule-based algorithm, in terms of consuming time. We can see that there is a trend that the more features an algorithm uses, the longer the time it takes to estimate iris mask. It is pretty reasonable and coincides with our intuition.

5.3 ROC curve analysis

The ultimate goal for us to have a good iris occlusion detector is to enhance the iris recognition performance with better iris mask. Therefore, besides measuring the similarity between the manual masks and masks estimated by each algorithm, we also perform the whole iris recognition process using the created iris masks. From Fig. 7, we can clearly see the rank of performance of each algorithm. For both ICE1 and ICE2, the performance rank is similar. Manually created masks achieved the highest score, which make sense to us. FLDA and rule-based algorithms are two of the worst, which makes sense too because their masks deviated farthest from the perfect masks.

One thing that counters our intuition is that for both ICE1 and ICE2, the best performance given by proposed algorithm is GFSG. For ICE1 database, it is still reasonable because from Fig. 4, the AER of GFSG is second best. But for ICE2 database, AER of GFSG is in the middle rank. Therefore, the result shown in Fig. 7(b) is surprising to us.

The possible explanation is that although the mask created by GFSG is not so accurate as some other feature set, and some eyelid/eyelash occlusions are mistakenly being recognized as the iris texture, but those eyelid/eyelash regions may carry discriminative information for iris recognition. This discriminative information, plus original iris texture information, makes the final iris recognition system performs well.

This explanation can be illustrated by Fig. 8. In Fig. 8, two pairs of example images are shown. Figure 8(a) and (b) are two images from the same iris class, and its corresponding masks created by GFSG feature set. These two images are example iris masks that have higher error rate compared to the rest of the iris masks for the same class. Figure 8(c) and (d) show another pair of iris and their masks, from another iris class. From these two pairs of examples, we can see that for the case that GFSG gives an erroneous iris mask, usually the regions that are mistakenly recognized as iris texture still have some discriminative information. In examples shown in Fig. 8, the incorrectly recognized region has spacial eyelash patterns that belong to that class. Therefore it is possible to have higher iris recognition performance.

6. CONCLUSIONS AND FUTURE WORK

Extracting robust iris masks is one of the key stages to improving iris recognition. This is something that many researchers have neglected. How to estimate iris occlusion and create accurate iris mask robustly and efficiently is key to high performance iris recognition. In this work, we have demonstrated that our approach for the iris occlusion estimation is efficient, and effective. It can detect the iris occluded region and the detection result (iris mask) can be very similar to manually created result. It is effective because we only use one training image for each class to get such a satisfactory result. Also, it only requires less than 0.1 second for one image (for GFSG case, in Matlab implementation). Experiments have also shown that the iris masks created by proposed method can help improve iris recognition performance to be very close to the manually created masks.

On the other hand, there is still room for improvement. The best AER we get is 9.5%, which means we can still try to improve the results. Future works include using more sophisticated filters to extract features for FJ-GMM training. Also, we can apply some other method like morphological operation to refine the estimated iris mask so that it can be more similar to manually created ones. In addition, we can also try other machine learning algorithms like SVM or KNN to see if they are better than FJ-GMM method.

REFERENCES

1. J. Daugman, "High confidence visual recognition of persons by a test of statistical independence," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(11), pp. 1148–1161, 1993.
2. L. Ma, T. Tan, Y. Wang, and D. Zhang, "Personal identification based on iris texture analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**(12), pp. 1519–1533, 2003.
3. C. loic Tisse, L. Martin, L. Torres, and M. Robert, "Person identification technique using human iris recognition," *Proc. of Vision Interface* , pp. 294–299, 2002.
4. J. Daugman, "How iris recognition works," *Circuits and Systems for Video Technology, IEEE Transactions on* **14**, pp. 21–30, Jan. 2004.
5. W. Kong and D. Zhang, "Accurate iris segmentation based on novel reflection and eyelash detection model," *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on* , pp. 263–266, 2001.
6. J. Zuo, N. Kalka, and N. Schmid, "A robust iris segmentation procedure for unconstrained subject presentation," *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the* , pp. 1–6, 19 2006-Aug. 21 2006.
7. L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
8. M.-H. Yang and N. Ahuja, "Gaussian mixture model for human skin color and its applications in image and video databases," *Storage and Retrieval for Image and Video Databases VII* **3656**(1), pp. 458–466, 1998.
9. C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* **2**, pp. –252 Vol. 2, 1999.
10. J. Ilonen, J.-K. Kamarainen, H. Kalviainen, and O. Anttalainen, "Automatic detection and recognition of hazardous chemical agents," *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on* **2**, pp. 1345–1348 vol.2, 2002.
11. T. Lindh, J. Ahola, J. Kamarainen, V. Kyrki, and J. Partanen, "Bearing damage detection based on statistical discrimination of stator current," *Diagnostics for Electric Machines, Power Electronics and Drives, 2003. SDEMPED 2003. 4th IEEE International Symposium on* , pp. 177–181, Aug. 2003.
12. M. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(3), pp. 381–396, 2002.
13. J. Thornton, "Matching deformed and occluded iris patterns: a probabilistic model based on discriminative cues," *PhD thesis, Carnegie Mellon University* , 2007.
14. "Iris challenge evaluation," *National Institute of Standards and Technology, http://iris.nist.gov/ICE/* , 2006.
15. L. Masek and P. Kovesi, "Matlab source code for a biometric identification system based on iris patterns," *The School of Computer Science and Software Engineering, The University of Western Australia* , 2003.