Detecting and Tracking Eyes Through Dynamic Terrain Feature Matching

Jun Wang and Lijun Yin
Department of Computer Science
State University of New York at Binghamton
Binghamton, NY, 13902, USA

Abstract

Automatic eye detection and tracking is an important component in the advanced human-computer interface design. In this paper, we present a novel approach for detecting and tracking eyes through matching their terrain features. Regarded as a 3D terrain surface, eye region exhibits certain intrinsic traits when using a so-called topographic representation. With the topographic classification of terrain features, we generate a terrain map for each facial image and extract eye candidates from the terrain map. Our algorithm mainly consists of two parts. First, eye locations are estimated from the candidate positions using an appearance-based object recognition technique. Second, a mutual information based fitting function is defined to describe the similarity between two terrain surfaces. By optimizing the fitting function, eye locations are updated for each frame in a video sequence. The distinction of the proposed approach lies in that both eye detection and eye tracking are performed in a terrain map domain rather than an original intensity image domain. The robustness of the approach is demonstrated under various imaging conditions and with different facial appearances using a web camera.

1. Introduction

Research on eye detection and tracking has been intensified in recent years, driven by its important application in non-verbal human computer interaction [9, 2, 12]. As one of the most salient and stablest facial features, eye can be used for helping locate face, providing gaze information and even identifying facial behaviors (e.g., expressions).

During the past decade, great technical progress has been made for eye detection and eye tracking. Typically, the holistic method and the abstractive method have been developed for eye detection [7, 20]. The holistic method utilizes the global information to locate eyes, such as the Eigenspace based method [11]. The abstractive method applies standard pattern analysis algorithms to locate eyes with extracted local appearance features, such as deformable template [19]. Both Eigenspace-based and template-based methods evaluate the appearance features in the intensity image domain, which is sensitive to various imaging conditions, facial poses and expressions. Moreover, the framework for rapid object detection using a boosted cascade, proposed by Viola and Jones, can be applied for eye detection [15].

Due to the fact that the iris of human eye has large reflection to infrared light, the infrared (IR) illumination based technique is widely employed for locating eyes [21, 6]. This approach is relatively effective and robust. However, it requires the special hardware with IR lighting cameras. The result of detection and tracking still depends on the various eye appearances (e.g., orientation, size and eye blinking, etc.) In order to improve the performance of eye tracking, Kalman filter [6] or Mean Shift techniques [1, 4, 21] can be applied as the alternative remedies for real time implementation with sufficient accuracy.

Observing that the pupil center of human eye exhibits fuscous while the eye white appears bright, if a gray-scale image is treated as a 3D topographic surface with the height of each location being denoted by the intensity of the corresponding pixel, the eye region will show a certain terrain pattern. In particular, the center of eye exhibits the *"pit"* feature surrounded by *hillside* features. This gives us a hint that eyes can be detected by exploring their terrain features.

Motivated by the topographic analysis technique [14, 17], in this paper, we propose a new method using a terrain feature matching algorithm for eye detection and tracking. First, we derive a terrain map from a gray level image based on the topographic primal sketch theory. The terrain map is composed of topographic labels, where each pixel is labeled by one of the twelve different types of terrain features. Second, we extract the *pit* pixels in the terrain map as the candidates for pupil pair classification. Third, a probabilistic appearance model is learned to describe the distribution of terrain features and used as a classifier to choose the eye pair from the candidate points.

After determining the initial eye location, we can further proceed to track eyes through dynamically matching their surface patches between two adjacent frames. A mutual information (MI) based fitting function is constructed to estimate the similarity between two patches. Although the eye location can be tracked by optimizing the fitting function, it is computationally expensive if we exhaust all the possible matchings in the search area. Alternatively, here we apply an efficient strategy to find the optimal feature match. We take advantage of the *pit* terrain features to selectively compute the mutual information in the terrain map domain. Using such a strategy brings twofold benefits. First, the probability distribution function ($p.d.f.$) is much easier and more precise to be estimated in the terrain map domain than in the intensity domain because the terrain map has only twelve types of topographic features while intensity domain has 256 levels. Second, the optimal match can be usually found in the location of *pit* feature pixel. This implies that for most of cases, we do not need to traverse all the pixels in the search area. Experiments on eye tracking show that the optimal match can be achieved for more than 98% of frames by searching only several *pit* pixels in real video sequences.

The rest of the paper is organized as follows. In Section 2, the background of topographic representation and classification of gray scale image is introduced. In Section 3, the algorithm for estimating initial eye locations is described, followed by the description of the eye tracking algorithm in Section 4. Finally, the experimental result and concluding remarks are discussed in Section 5 and Section 6.

2. Topographic Feature Extraction

The gray scale image is typically represented as a 2-dimensional lattice of 1-dimensional intensity vectors. The space of the lattice is known as the *spatial* domain while the gray information is denoted as *range* domain [3]. Differing from other joint spatial-range domain based analysis, topographic analysis treats the gray level images as a continuous 3D terrain surface, instead of distribution of discrete points. The intensity $I(x, y)$ is represented by the height of the terrain at pixel (x, y). Figure 1 shows an example of a face image and its eye terrain surface.

As we know, the intensity variations on a 2D face image is caused by the face surface orientation and its reflectance. The resulted texture appearance provides an important visual cue to classify a variety of facial regions and features [18]. If viewed as a 3D terrain surface, a face image shows certain “waves” in the face region due to its surface reflectance property. For the eye region, it is generally composed of two parts: the black pupil part and the white part. The center of eye (or pupil) usually appears some *pit* features surrounded by some *hillside* features.

Mathematically, we can give a strict definition for a *pit* feature on the terrain surface. Assume that a continuous surface is represented by $z = f(x, y)$, the Hessian matrix can be obtained:

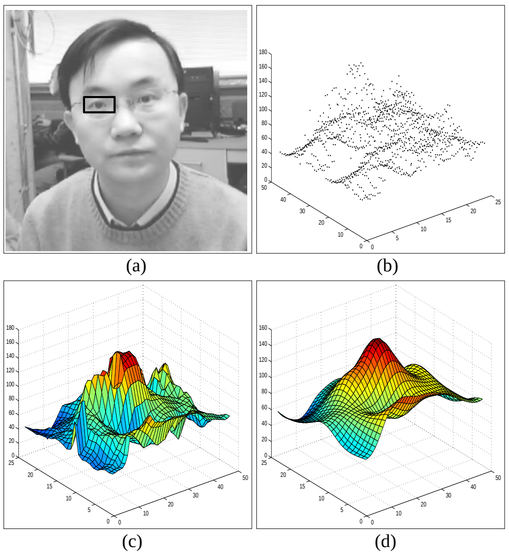

Figure 1: A face image and the corresponding 3D terrain surface of the eye region. The surface is reversed for better visualization, so the *peak* denotes the *pit* in real surface. (a) Original face image, marked out a eye patch with a size of 24×48 pixels; (b) Corresponding distribution in the joint spatial-range space with 1152 points; (c) Continuous terrain surface of the eye patch in the original image; (d) Smoothed terrain surface of the eye patch using a Gaussian filter with a kernel size of 15×15 and $\sigma = 2.5$.

$$\mathbf{H}(x, y) = \begin{bmatrix} \frac{\partial^2 f(x,y)}{\partial x^2} & \frac{\partial^2 f(x,y)}{\partial x \partial y} \\ \frac{\partial^2 f(x,y)}{\partial x \partial y} & \frac{\partial^2 f(x,y)}{\partial y^2} \end{bmatrix} \quad (1)$$

After applying eigenvalue decomposition to the Hessian matrix, we can get:

$$\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{U}^T = [\mathbf{u_1} \ \ \mathbf{u_2}] \cdot diag(\lambda_1, \lambda_2) \cdot [\mathbf{u_1} \ \ \mathbf{u_2}]^T \quad (2)$$

where λ_1 and λ_2 are the eigenvalues and $\mathbf{u}_1$,$\mathbf{u}_2$ are the orthogonal eigenvectors. A *pit* pixel can be detected when a local minimum gradient $\|\nabla f(x, y)\|$ is found in the local region. In other words, the following conditions must be satisfied, $\|\nabla f(x, y)\| = 0, \lambda_1 > 0, \lambda_2 > 0$ (i.e., the gradient is zero and the second directional derivative is positive in all directions.)

Similarly, there are also other terrain types being defined, such as *peak*, *ridge*, *saddle*, *hill*, *flat* and *ravine* [5]. *Hill* pixels can be further specified as one of the labels *convex hill*, *concave hill*, *saddle hill* or *slope hill*, and *saddle hills* can be further distinguished as *concave saddle hill* or *convex saddle hill*, saddle as *ridge saddle* or *ravine saddle* [14, 17]. Figure 2 shows the twelve types of terrain features.

Notice that the definition of terrain feature is restricted to the continuous 3D surface. In order to apply it to the digital image, the continuous surface function must be regressed from the discrete 3D points. We use a smoothed differentiation filter based on the Chebyshev polynomials to fit a small patch to the local surface. Then each surface point can be classified by its gradient value and principal curvatures (details can be found in [10, 17, 18].)

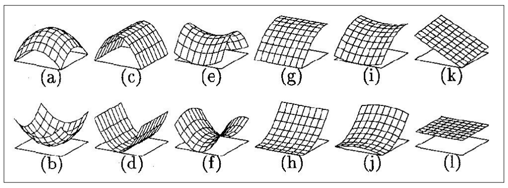

Figure 2: Topographic labels: The center pixel in each example carries the indicated label [14]. (a) *peak*; (b) *pit*; (c) *ridge*; (d) *ravine*; (e) *ridge saddle*; (f) *ravine saddle*; (g) *convex hill*; (h) *concave hill*; (i) *convex saddle hill*; (j) *concave saddle hill*; (k) *slope hill*; and (l) *flat*.

In general, in order to reduce the noise, it is necessary to apply a smoothing process before regressing the surface. Figure 1 (d) shows an example of the terrain surface of an eye region after it is smoothed by a Gaussian filter.

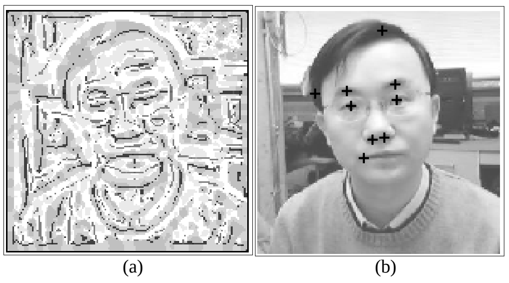

Figure 3: (a) Results of topographic classification of a facial image (e.g., twelve gray levels from black to white represent the labels from (a) to (l) respectively); (b) The *pit* features in the facial region are marked by the “cross” symbols.

By applying the topographic classification technique, each pixel in the original gray scale image is labeled by one of the twelve terrain features. Figure 3(a) shows one labeled face image, in which each label type is rendered by a distinct gray level. Figure 3(b) shows the detected *pit* features (denoted by “cross” symbols) in the face region. As we can see, the *pit* features are distributed very sparsely in the facial region. Although there are only a small number of *pit* features being detected, eye is the most reliable feature which shows the *pit* character in the face region. In order to extract the eye pair among the candidate *pit* pixels, we use a statistical verification approach to remove the fake feature pixels, which will be described in the next section.

3. Topographic Eye Location

Given the candidate set of *pit* features, a pair of eyes can be determined according to its local appearance in the terrain map. In order to classify the candidate *pit* pixels, we select a patch around each candidate pixel to analyze its terrain feature distribution. Assume that the centers of the left pupil and the right pupil are located at points α and β respectively, and the distance between them is measured as d. Two rectangular patches centered at α and β are then generated along the direction of line $\alpha\beta$. Each patch has a size of $0.6d \times 0.3d$. In order to determine the real eye location from the candidate *pit* features, we use a parametric probabilistic model to evaluate the possibility of the two points being an eye-pair. In the terrain map domain, each pixel is quantized to a range of $1, 2...M$, where M is the number of terrain type (i.e., $M = 12$). To analyze the topographic feature distribution, we generate a terrain feature vector for each candidate patch, which is defined as $\mathbf{t} = \{t_1, t_2, ...t_i, ..., t_N\}$, where $1 \leq t_i \leq M$ is the terrain type of each pixel and N is the number of pixels in the patch.

Here we employ a Gaussian Mixture Model (GMM) to describe the property of the terrain feature vector. If we treat each terrain feature vector of *pit* pixels as a sample, GMM presumes all the samples distribute in a high-dimensional space complying with several Gaussian distributions. Among all the samples for both eye candidates and non-eye candidates, we can further categorize them into three sub-spaces, i.e., a left eye space, a right eye space and an non-eye space. Each of them is described by a Gaussian distribution. Let's define a subspace $\mathcal{E}_l$ for the left eye, $\mathcal{E}_r$ for the right eye and $\mathcal{U}$ for the non-eye candidates, with the probability distribution being $\mathcal{N}_l(\mu_l, \mathbf{\Sigma}_l)$, $\mathcal{N}_r(\mu_r, \mathbf{\Sigma}_r)$ and $\mathcal{N}_u(\mu_u, \mathbf{\Sigma}_u)$, respectively. All these three subspaces constitute a sample space $\mathcal{O}$, whose parameterized form is defined as:

$$\mathcal{O} = \{\mu_l, \mathbf{\Sigma}_l, p_l; \mu_r, \mathbf{\Sigma}_r, p_r; \mu_u, \mathbf{\Sigma}_u, p_u\} \tag{3}$$

where p_l,p_r and p_n are the prior probabilities, μ and $\mathbf{\Sigma}$ denote the mean and covariance matrix of the Gaussian distributions. Given a pair of candidates, a and b, with the terrain feature vectors being $\mathbf{t}_a$ and $\mathbf{t}_b$, the posterior probability of the candidate pair belonging to the eye space $\mathcal{E} = \{\mathcal{E}_l, \mathcal{E}_r\}$ is calculated as follow:

$$\begin{aligned} p(\mathcal{E}|\mathbf{t}_a, \mathbf{t}_b) &= p(\mathcal{E}_l|\mathbf{t}_a) \cdot p(\mathcal{E}_r|\mathbf{t}_b) + p(\mathcal{E}_l|\mathbf{t}_b) \cdot p(\mathcal{E}_r|\mathbf{t}_a) \\ &= \frac{p(\mathbf{t}_a|\mathcal{E}_l) \cdot p_l}{p(\mathbf{t}_a|\mathcal{O})} \cdot \frac{p(\mathbf{t}_b|\mathcal{E}_r) \cdot p_r}{p(\mathbf{t}_b|\mathcal{O})} \\ &+ \frac{p(\mathbf{t}_a|\mathcal{E}_r) \cdot p_r}{p(\mathbf{t}_a|\mathcal{O})} \cdot \frac{p(\mathbf{t}_b|\mathcal{E}_l) \cdot p_l}{p(\mathbf{t}_b|\mathcal{O})} \end{aligned} \tag{4}$$

where the probability $p(\mathbf{t}_a|\mathcal{O})$ and $p(\mathbf{t}_b|\mathcal{O})$ are calculated as:

$$p(\mathbf{t}_a|\mathcal{O}) = p(\mathbf{t}_a|\mathcal{E}_l)\cdot p_l + p(\mathbf{t}_a|\mathcal{E}_r)\cdot p_r + p(\mathbf{t}_a|\mathcal{U})\cdot p_u \quad (5)$$

$$p(\mathbf{t}_b|\mathcal{O}) = p(\mathbf{t}_b|\mathcal{E}_l)\cdot p_l + p(\mathbf{t}_b|\mathcal{E}_r)\cdot p_r + p(\mathbf{t}_b|\mathcal{U})\cdot p_u \quad (6)$$

With the estimated parametric model, the real eye-pair can be extracted according to the maximum value of probability $p(\mathcal{E}|\mathbf{t}_a, \mathbf{t}_b)$.

We use a training set, including 293 face images, to learn the parameters of the probabilistic model. Among all the candidate pixels (*pit*-labeled pixels), we randomly select a pair of candidates, and generate a terrain patch for each candidate. After obtaining all the terrain patch vectors, we divide them into three sets, i.e., two positive sets including a group of left eyes and right eyes samples, respectively, and a negative set including a non-eye group. Figure 4 shows some samples of the positive set and the negative set used for training.

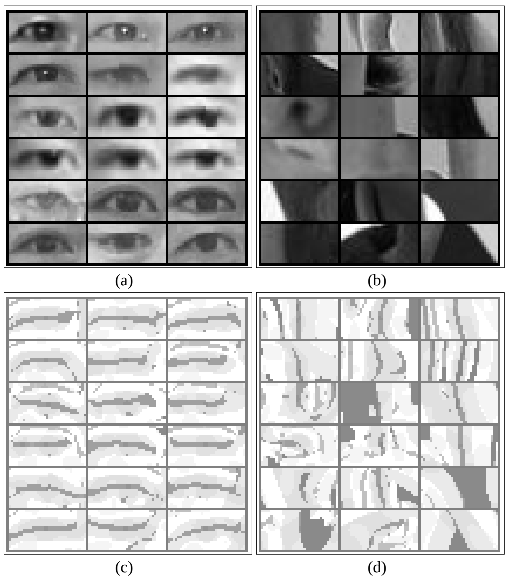

Figure 4: (a) The samples used as a positive training set, whose corresponding terrain patches are shown in (c); (b) The non-eye samples used as a negative training set, whose corresponding terrain patches are shown in (d).

From the training set, the parameters in formula 3 can be estimated. By using such a probability model, we are able to extract the eye-pair with a maximum probability value calculated by Equation 4. Note that during the classification of all candidate pixels, searching all possible pairs of candidates is a time-consuming process. In order to reduce the search space, we discard the pairs of candidates which have unreasonable distances between them (e.g., the distance of a pair of candidates is beyond the range of a normal eye-pair distance with respect to the given image size, or the orientation of the eye-pair is near the vertical direction.) As such, only a small number of pairs of candidates need be examined, and thus the computation load can be greatly reduced. Figure 5 shows four examples from the first frames of four videos, where the eyes of four subjects are detected correctly from a few of candidate pixels. The topographic eye location approach has some ceartain robustness to unconstrained background. Moreover it can be extended to solve multiple-face cases, which is verified by the experiments in our previous work [16].

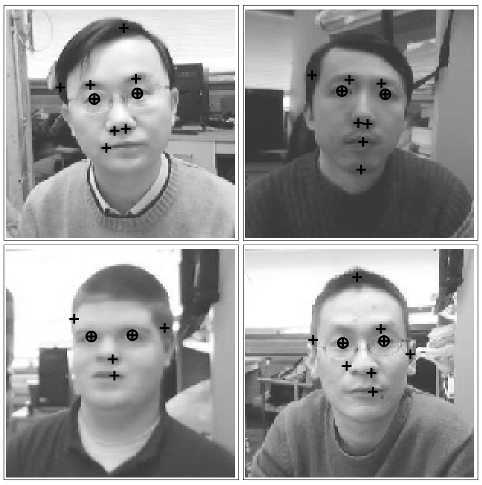

Figure 5: Examples of eye detection: "cross" symbols in the face regions mark the candidate locations; "circle + cross" symbols mark the detected eyes.

4. MI-Based Eye Tracking

After locating the eye position in the initial frame, the eye motion can be tracked in the subsequent frames. It seems that the GMM based model could be used to find the eye location in each frame. However, due to the training set only includes frontal facial images, it is not feasible to track eyes under various poses. We seek to explore the mutual information (MI) between neighbor frames to achieve a fast and robust tracking. Our experiments show that it is more reliable to use the eye patch detected in the previous frame as a dynamic template to estimate the eye location in the current frame.

Given the eye locations α and β of the ith frame, the eye positions α' and β' in the jth frame can be found through matching the terrain surface of patches. Figure 6(a-b) illustrates two sample frames, indexed as i and j, where the jth frame is several frames after the ith frame. Figure 6(c-d) shows two smoothed terrain surfaces of the left eye, which correspond to the image patches in (a) and (b), respectively. As shown in this figure, the two surfaces exhibit the similar terrain patterns, while the intensities of the corresponding image patches distribute in different ranges: one in $0-180$, the other in $0-200$.

The similarity of the two patterns can be measured in a 3D surface domain or a intensity image domain. However, in order to find a match efficiently, we use the mutual information to measure the similarity of two terrain patches in a terrain map space.

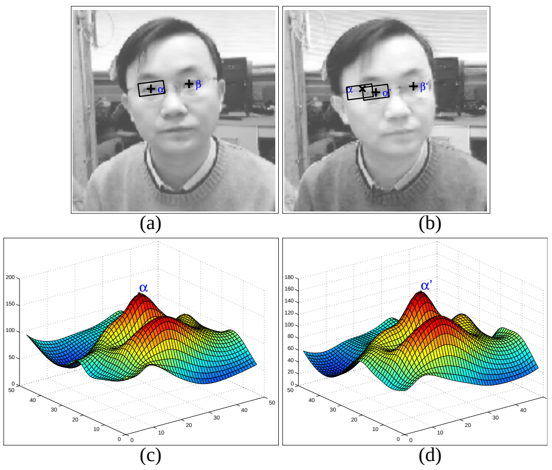

Figure 6: Frame i (a) and frame j (b) show the centers of pupils, denoted by (α, β) and (α', β'), respectively. (c-d): The terrain surfaces of the left eye patch in frame i and frame j.

Assume that two patches are centered at α and α' and the corresponding terrain feature vectors are $\mathbf{t}_\alpha$ and $\mathbf{t}_{\alpha'}$, which are represented by the random variables as X and Y, the mutual information between the two variables is calculated as:

$$I(X,Y) = \sum_X \sum_Y P_{XY}(\mathbf{t}_\alpha, \mathbf{t}_{\alpha'}) \log \frac{P_{XY}(\mathbf{t}_\alpha, \mathbf{t}_{\alpha'})}{P_X(\mathbf{t}_\alpha) \cdot P_Y(\mathbf{t}_{\alpha'})} \quad (7)$$

In order to calculate the mutual information, we must estimate the marginal and joint $p.d.f.$ of the random variables X and Y (where $1 \leq X, Y \leq 12$). Because there are only 12 kinds of terrain labels rather than 256 levels in the intensity domain, it is fairly easy and fast to estimate the discrete $p.d.f.$s P_X,P_Y and P_{XY}, which correspond to the normalized 1D and 2D histograms of the terrain map.

As described in Section 3, the appearance based terrain features can be represented by a terrain map. Figure 7(a-b) illustrates the terrain maps of two patches corresponding to the two surfaces in Figure 6(c-d). The terrain patch is marked out by a rectangle along the direction of the detected eye-pair in the ith frame. Let $p_i = \alpha$ denote the determined eye location in the ith frame and the variable $\hat{p} \in \mathcal{P}$ represent the current searching position in the $(i+1)$th frame, where $\mathcal{P}$ defines a search area. Then the mutual information can be defined as a function of the variable $\hat{p}$, as shown in the following formula:

$$I(X,Y) = g(\hat{p}) = g(\hat{p}_x, \hat{p}_y) \quad (8)$$

where $\hat{p} = (\hat{p}_x, \hat{p}_y)$ is a 2D coordinate of the patch center in the $i+1$ frame. The function $g(\hat{p})$ does not have the explicit form, but it can be computed with the sampled $\hat{p}$. Figure 7 (c) plots the MI values within a patch. From the theory of sparse structuring of statistical dependency [13], the variable of terrain features has strong statistical dependency with a small number of other variables and negligible dependency with the reaming ones. The exactly geometrical alignment of two patches demonstrates the property. As we know, the statistical dependency can be empirically evaluated by mutual information (MI). If we take the position of a terrain patch as a variable, the estimated MI value varies along with this variable. As illustrated in Figure 7, when the rectangular terrain map in (a) matches the region of (b) centered at α', the MI function outputs a maximum value I_{max}. The statistical dependency described by MI can be visualized in (c) and (d). The brightest spot shows the position with strongest statistical dependency between the two patches while the dark or shaded areas indicate different degrees of patch independency.

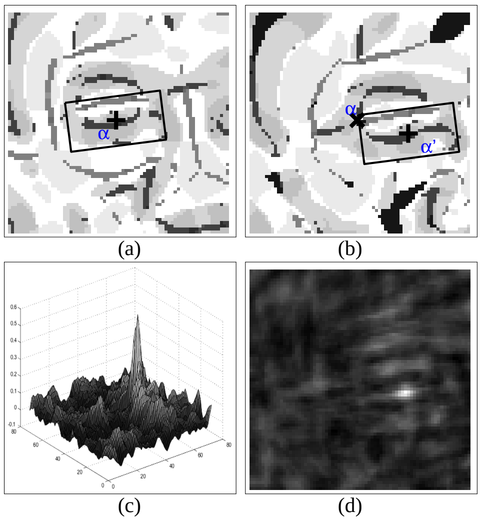

Figure 7: (a-b) The terrain maps of the frames i and j, where rectangles denotes the eye regions; (c) MI calculated on the jth frame as a function of the patch position. (d) The statistical dependency measured by the empirical mutual information of the jth frame. The white spot corresponds to the peak position of (c).

To this end, we formulate the fitting function for patch matching in the $(i+1)$th frame, given the eye location p_i in the ith frame:

$$f(p_i, \hat{p}) = g(\hat{p}) + \lambda \cdot e^{-\frac{\|p_i - \hat{p}\|}{\eta}} \tag{9}$$

The fitting function $f(\cdot)$ is composed of two parts: the mutual information $g(\cdot)$ for measuring the statistical independency and the distance penalty term for guaranteeing the continuity of tracking and preventing the terrain match from distracting by other similar regions (e.g., eyebrows). The parameters λ and η are used to balance the weights of the two parts. The updated eye locations of the $(i+1)$th frame is obtained by maximizing the fitting function:

$$p_{i+1} = \arg\max_{\hat{p} \in \mathcal{P}} f(p_i, \hat{p}) \tag{10}$$

It is conceivable that the computation cost is fairly expensive if the fitting function is optimized through traversing all the pixels in the search area. Fortunately, our target location (i.e, center of eye or pupil) shows, in most cases, the stable *pit* feature, which makes the maximum fitting value appear at such a *pit* feature location reliably. Therefore, the eye location can be estimated quickly by searching the best fitted patch in a very few *pit* locations.

When eyes are completely closed or the head rotation is in a large degree which makes pupils almost invisible, the fitting function outputs a small value, which signifies the loss of tracking. In this case, an approximate eye location must be estimated by finding the maximum fitting function value through all the pixels in the search area. Note that the distance penalty term can prevent the tracking point from jumping far into the other non-eye *pit* locations, such as eyebrows. This function maintains the smooth tracking of eyes. The whole automatic eye tracking algorithm is summarized as follows:

```
1. Derive the terrain map of the first frame using topographic classification technique and set the pit pixels as eye candidates;
2. Localize the eye-pair positions in the first frame through the GMM probability maximization as shown in Eq. 4;
3. Given eye locations of the $i$th frame as $p_i$ and the terrain feature $\mathbf{t}_i$, determine a search area $\mathcal{P}$ with size $K \times K$ and center $p_i$ in the $(i+1)$th frame for searching the current eye location $p_{i+1}$;
4. Calculate the terrain map of the selected search area $\mathcal{P}$ and detect the pit pixels. Compute the fitting function at each pit pixel location and get the maximum value.
5. If there is no pit pixel in the patch or the maximum fitting value computed in (4) is less than the predefined threshold $\theta$, maximize the fitting function by computing $f(p_i, \hat{p})$ for each pixel $\hat{p} \in \mathcal{P}$;
6. Update the current eye locations and eye terrain feature as: $i+1 \longrightarrow i, p_{i+1} \longrightarrow p_i, \mathbf{t}_{i+1} \longrightarrow \mathbf{t}_i$. Go to step 3 for the next frame tracking;
```

Note that unlike the computation in the eye detection stage, eye tracking stage only computes the terrain map in a small search area around the eye rather than the whole face region, it greatly reduces the computation time.

5. Experiments

The proposed eye detection and tracking algorithms are evaluated through the real time video sequences. We used normal webcam to capture the video (i.e., CREATIVE LABS webcam NX Ultra with frame resolution of 640×480.) The first frame is used for detecting eyes by our topographic-based eye detection algorithm. The whole procedure runs fully automatically.

The performance of the initial eye localization is affected by two factors: the candidate detection and the estimation of eye pairs from the candidates. The result of candidate detection relies on the parameter selection of smoothing filter and differentiation filter, especially the scale of the filters. In order to reduce noises as well as maintain facial image details, we set the consistent parameters for the following operations, both for the training images and for the testing videos.

- The Gaussian filter for smoothing has the size 15×15 and $\sigma = 2.5$;
- The discrete Chebyshev polynomial based differential filter has the kernel with a size of 5×5;
- The width and height of eye window is $0.6d$ and $0.3d$ (d is the distance of a pair of eyes).
- The search area for tracking is $(0.6d+15) \times (0.3d+15)$ pixels.
- For the fitting function, the coefficient λ is 0.4, and the value of η is set as the distance of the tracked eye pair in the previous frame. The threshold θ is set as 0.65.

Our eye detector is tested on a static image database (i.e., Japanese Female Facial Expression (JAFFE) database [8]). The JAFFE images of each subject show seven universal facial expressions. Figure 8 demonstrates some sample results from our eye detector. Among 213 facial images, 204 of them are correctly detected. The algorithm achieves 95.8% correct detection rate. As an initialization stage, we apply the eye detector to ten test videos, which are captured in our lab environment with a complex background. Experiments show that our eye detector localized eyes of the first frames of all the videos correctly. Figure 5 demonstrates some sample frames.

We test our tracking algorithm in two scenarios using both a fixed webcam and a movable webcam. In each scenario, the eye appearance of a subject is changed along

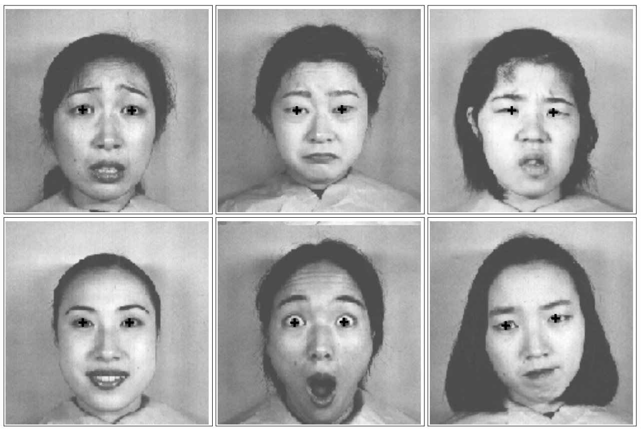

Figure 8: Examples of eye detection on the JAFFE database [8].

with the aspects of facial scales (e.g., moving forward and backward), facial poses (e.g., rotating head), gaze directions (e.g., rotating eye-balls), eye status (e.g., blinking, opening/closing eyelids by expressions), illuminations (e.g., changing lighting orientations and intensities) and partial face occlusions (e.g., wearing eye glasses or hiding non-eye areas). Figure 9 shows one example sequence which was captured by a fixed webcam. Figure 10 illustrates the sample frames from a video clip captured by an active webcam, which performed panning, tilting and rolling operations. The experiment shows that our eye detection and tracking algorithms perform well under various imaging conditions (The video clips can be found in author's website.)

Our eye tracking algorithm runs 18 frames per second on the PC with a single CPU (P4-3.4GHz.) We tested on 10 videos performed by 5 different subjects under varying imaging conditions (e.g., dynamic change of lighting, moving of camera, etc.) Each video has 400 - 600 frames. Experiments show that most of the time (98% of frames) the system outputs the correct tracking result (i.e., locating pupil positions precisely.) Our system fails to track eyes if the following cases occur: (1) the head rotation is beyond a certain range to make the eye invisible; (2) the eye is completely closed; (3) the subject is far from the camera, so the size of eye appeared in the image is too small. In the situation of missing track, we use the previous frame to find the eye location by re-initializing the system or wait until the normal case is restored.

As compared to the conventional eye tracking approaches, our approach is advantageous in that (1) no special hardware (e.g., IR devices) is employed; (2) no face tracker is required, which alleviates the potential instability of the system; (3) The use of the topographic terrain map makes the MI calculation very fast because the iteration is based only on the range of 12 in the terrain domain rather than the range of 256 in the intensity domain.

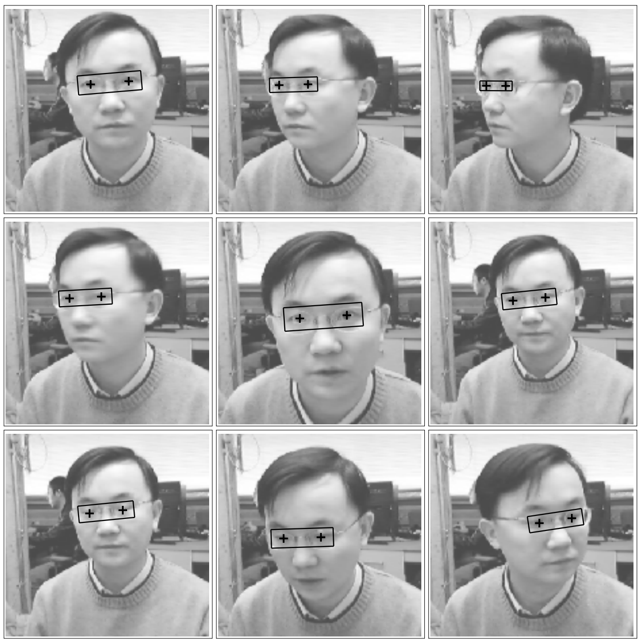

Figure 9: (A) Sample frames of detected and tracked eyes from a video sequence captured by a static web-cam. From top-left to bottom-right: frame 1, 31, 54, 68, 125, 192, 203, 292 and 404.

6. Conclusions and Discussions

In this paper, we proposed a system for eye detection and tracking through matching the terrain features. Our system works automatically using an ordinary webcam without special hardwares. The major contribution of this work lies in the proposed unique approach for topographic eye feature representation, by which both eye detection and eye tracking algorithms can employ the probabilistic measurement of eye appearance in the the terrain map domain rather than the intensity image domain. In addition, we defined a fitting function based on the mutual information to describe the similarity between terrain surfaces of two eye patches. With fairly small number of terrain types, the $p.d.f.$ of marginal and joint distributions can be easily estimated, and eventually, the eye location is determined by optimizing the fitting function efficiently and robustly.

Unlike some other approaches [9] which require to estimate the face region as a first pre-processing step, our algorithm can detect eyes directly from images with complex background. Since the maximum fitting value usually appears on the *pit* feature pixels, the matching process can be only performed on several candidate *pit* locations in the ter-

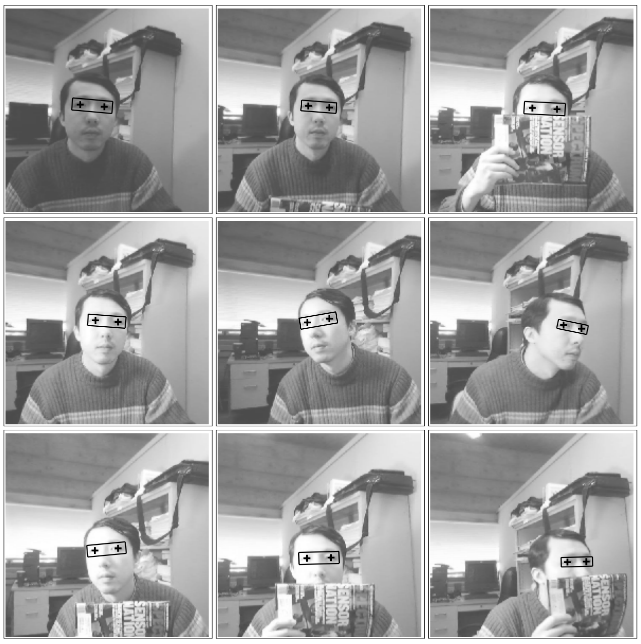

Figure 10: (B) Sample frames of detected and tracked eyes from a video sequence captured by an active web-cam. From top-left to bottom-right: frame 1, 73, 145, 250, 360, 425, 490, 515 and 540.

rain map. This saves us from ransacking all the pixels in the search area. The experiments show that our system can track eyes precisely in most of the time (98% of frames) using both static and active cameras under various imaging conditions and with different facial appearances. In the case of failure (e.g., large head rotation, eye invisible or eye close), we take the measure by re-initializing eye locations using the previous frame, or waiting until the normal case is restored by monitoring the fitting function values.

Note that our topographic-based appearance model can alleviate the influence of various imaging conditions. However, the image smoothing process and the surface patch fitting process are all dependent on the facial scale in the image. Our future work is to include a wide variety of training samples with multi-scale facial images and various lighting conditions to improve the robustness of eye detection and eye tracking. We will also extend the work to further analyze the facial pose information, as well as extend to detect precise eye information, such as gaze. Currently, our method achieves the tracking speed of 18 frames per second. Generating terrain map costs the most computation time, however, the nature of the parallel calculation of the terrain feature allows us to use dual CPUs to make multiples face tracking simultaneously in the future.

Acknowledgments

The authors would like to thank Micheal Lyons for providing JEFFE database. This work was supported by the National Science Foundation under grant No. IIS-0414029.

References

[1] G. Bradski. Real time face and object tracking as a coponent of a perceptual user interface. In *IEEE Workshop on Applications of Computer Vision*, Princeton, 1998.

[2] A. Amir C. Morimoto, D. Koons and M. Flickner. Real-time detection of eyes and faces. In *Workshop on Perceptural User Interfaces*, 1998.

[3] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *ICCV'99*, 1999.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE CVPR'2000*.

[5] R. Haralick, L. Watson, and T. Laffey. The topographic primal sketch. *Inter. J. of Robotics Research*, 2(2), 1983.

[6] A. Haro, M. Flickner, and I. Essa. Detecting and tracking eyes by using their physiological properties dynamics and appearcance. In *IEEE CVPR'00*, 2000.

[7] J. Huang and H. Wechsleris. Visual routines for eye location using learing and evolution. *IEEE Trans. on Evolutionary Computation*, 4(1):73–82, 2000.

[8] M. Lyons, J. Budynek, and S. Akamatsu. Automatic classification of single facial images. *IEEE Trans. on PAMI*, 21(12):1357–1362, 1999.

[9] J. Magee, M. Scott, B. Waber, and M. Betke. Eyekeys: A real-time vision interface based on gaze detection from a low-grade video camera. In *IEEE CVPR Workshop on Real-Time Vision for Human Computer Interaction*, 2004.

[10] P. Meer and I. Weiss. Smoothed differentiation filters for images. *Journal of Visual Communication and Image Representation*, 3(1), 1992.

[11] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *IEEE CVPR94*.

[12] R. Ruddarraju and et al. Perceptual user interfaces using vision-based eye tracking. In *5th International conference on multimodel interfaces*, 2002.

[13] H. Scheiderman. Learning a restricted bayesian network for object detection. In *IEEE CVPR'04*, 2004.

[14] O. Trier, T. Taxt, and A. Jain. Data capture from maps based on gray scale topographic analysis. In *Inter. Conf. on Document Analysis and Recognition (ICDAR'95)*, 1995.

[15] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE CVPR'01*, 2001.

[16] J. Wang and L. Yin. Eye detection under unconstrained background by the terrain feature. In *ICME2005*, Amsterdan, Netherland, July, 2005.

[17] L. Wang and T. Pavlidis. Direct gray-scale extraction of features for character recognition. *IEEE Trans. on PAMI*, 15(10):1053–1067, 1993.

[18] L. Yin and A. Basu. Generating realistic facial expressions with wrinkles for model based coding. *CVIU*, 84(2):201–240, Nov. 2001.

[19] A. Yuille, P. Hallinan, and D. Cohen. Feature extraction from faces using deformable templates. *IJCV*, 8(2), 1992.

[20] Z. Zhou and X. Geng. Projection functions for eye detection. *Pattern Recognition*, 37(5), 2004.

[21] Z. Zhu, K. Fujimura, and Q. Ji. Real-time eye detection and tracking under various light conditions. In *ACM SIGCHI Symp. on Eye Tracking Res. and App.*, 2002.