# Detecting and Tracking Eyes By Using Their Physiological Properties, Dynamics, and Appearance

Antonio Haro† Myron Flickner‡ Irfan Essa†

†GVU Center / College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

‡Computer Vision Enhanced User Interfaces
IBM Almaden Research Center
San Jose, CA 95120

## Abstract

*Reliable detection and tracking of eyes is an important requirement for attentive user interfaces. In this paper, we present a methodology for detecting eyes robustly in indoor environments in real-time. We exploit the physiological properties and appearance of eyes as well as head/eye motion dynamics. Infrared lighting is used to capture the physiological properties of eyes, Kalman trackers are used to model eye/head dynamics, and a probabilistic based appearance model is used to represent eye appearance. By combining three separate modalities, with specific enhancements within each modality, our approach allows eyes to be treated as robust features that can be used for other higher-level processing.*

## 1. Introduction

In this paper, we present methods for tracking and detecting eyes in indoor environments. Robust and reliable eye detection is an important first step towards the development of user interfaces capable of gaze tracking and detecting eye contact. To support concurrent higher-level processing, algorithms for eye detection should be cheap both in cost and computational complexity. For this reason, we have developed an algorithm that runs in real-time on a consumer-end processor using an inexpensive (under $50) black and white camera with infrared lighting. Our algorithm does not require any camera calibration and works for all people. We also do not require users to do any preregistration prior to having their eyes detected.

Our main goal is to detect eyes reliably and in real-time. In our method, we use the physical properties of pupils along with their dynamics and appearance to extract regions with eyes. The detector gives us a probabilistic measure of eye detection for each region. The probability for each region is automatically weighted with components coming from the appearance and dynamics along with temporal information.

We use Kalman trackers to deal with the dynamics of head/eye movements. A probabilistic appearance based model of the eyes is used to compute statistics of the texture for different regions to aid in our classification. All three processes, measuring eye physiology, dynamics, and appearance, are merged to achieve robust detection and tracking.

Once we know which regions are likely to be eyes, we undertake higher-level processing on these regions. We observe pairs of regions and probabilistically determine which regions are most likely to be faces when paired together.

**Prior Work** There is much prior work [2] on finding faces and facial features in complex scenes. Some approaches, like Kothari, *et al.* [4], find eyes in images without finding faces. They observed that image gradient intersection points were good candidate locations for eyes. Most false positives were removed by ensuring similar numbers of gradient intersections for pairs, by using a priori inter-eye distances, and using temporal cues. The physical properties of the eyes are not taken into account nor are dynamics modeled as we propose in our method. In addition, the technique functions on the pixel level, so there is no model.

Other approaches like those of [10, 8] find faces first and then process the facial regions to find facial features. Faces are usually chosen to be located first because they occupy more of the image than their features. Our method differs from these approaches in that we use eyes to reliably locate faces.

Scassellati [10] finds faces in cluttered scenes in real-time. Once the faces are located, their system foveates on them and then zooms in on the left eye. However, their system uses ratio templates to find faces and, as such, is not rotationally invariant. Also, specialized DSP chips are required to achieve real-time performance. Our algorithm does not require any specialized chips to perform tracking in real-time.

Oliver, *et al.* [8] utilize blobs and Kalman tracking to find and track faces as well as facial features in real-time. They initially use blobs and color information to create a mixture model to find the faces and then assign trackers to track each face. The system runs in real-time and achieves

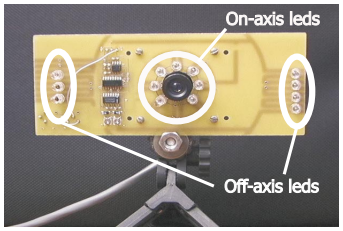


**Figure 1**. *Our infrared lighting camera [7]*

good results, but due to the fact that color and anthropometric statistics (to find facial features) drive their system, faces might not always be detected if they are of a size and shape that the system was not trained on. This situation could arise if a face is occluded by an object, or if someone is not completely within the field of view of the camera.

Our work has a lot in common with the work of Rasmussen, *et al.* [9]. In both works, trackers are given multiple sources of information to update their estimates. The main difference is that our algorithm runs in real-time as our model is simpler, without any degradation in performance. Also, like in their work, we utilize a probabilistic framework for our tracker's components. This allows us to combine the information from the modalities and is what makes our algorithm robust. These modalities can be interchanged with stronger components for possibly better performance as discussed in section 5.

## 2. Pupil Thresholding

Our system utilizes a black and white camera with infrared lighting [7] as a first step in pupil detection. The camera has two concentric rings of IR LEDs (Figure 1), one along the camera's axis, and one off-axis, to exploit the *red eye effect*. As the two rings are switched on and off, they generate two interlaced images (Figure 2) for a single frame. The image where the inner ring is on, which we refer to as the *bright image*, has white pupils. The image where the outer ring is on, the *dark image*, has dark pupils. The difference image that results by subtracting these two similar images contains interlacing artifacts and is quite noisy because regions of high specularity will have large differences. We use the difference image despite its noisiness as a preprocessing step since it contains valuable information.

There are several things we can do to get rid of the non-eye pixels in the difference image, such as thresholding the image. However, there is no guarantee that a particular threshold will be good for different environments. An adaptive thresholding algorithm could be used, but is very computationally expensive, especially when done on a consumer-end CPU as it must be done on every frame.

We use an adaptive thresholding algorithm that is computationally fast. This algorithm is intended to get rid of a conservative amount of specular reflection noise and interlacing artifacts. We would rather have excess candidates at this stage than miss a pupil region. The algorithm computes the histogram for the current frame, back-integrates it, and keeps a certain amount of the brightest pixels (about 1/1000 of the total number of pixels in the frame). The rest of the pixels are then set to black. This algorithm is extremely fast, which is desired as this is the lowest level step in our overall approach and the later steps are more computationally expensive.

Adaptively thresholded pixels are grouped into *candidate regions*. Candidate regions are those groups of pixels that are likely to be pupils. 16x16 (pixel) windows are centered on the brightest pixel in each candidate region. Pixels that are not at least three connected are not added to candidate regions and are set to black. We also ensure that the regions do not overlap as it is impossible for two pupils to overlap.

## 3. Candidate Region Tracking

Higher level information can now be gathered on these candidate regions. Tracking the candidate regions yields temporal information which is also used to handle temporary occlusions, such as blinks. Kalman filters are used because they update with a small number of highly optimizable matrix operations yet perform well for our model of the dynamics. Kalman trackers [13] are assigned dynamically as new regions to track are detected, and removed as these leave the field of view.

The trackers have four dimensional state vectors: $x$-position, $y$-position, velocity in the $x$ direction, and velocity in the $y$ direction. Acceleration information was not incorporated as it did not improve tracking performance. The dynamics are modeled as a one pixel variance between frames for the $x$ and $y$ components of the state vector, and a two pixel variance for each of the velocity components of the vector. This model works because we assume that people do not perform sudden and jerky movements.

Constant velocity is assumed for regions that do not have new measurements before removing their trackers, in case there was a temporary occlusion. We also ensure that trackers never track the same regions.

We use the Kalman filter's covariance matrix to give a measure of similarity between a particular region's motion compared to a pupil's motion. The covariance matrix update equation for the Kalman filter is given by:

$$P_k = (I - K_k H)P_k^{'}, \tag{1}$$

where K is the Kalman gain matrix, H is the connection between the state vector and measurement vector, $P_k^{'}$ is the prior estimate of $P_k$, and $P_k$ is the covariance matrix at time $k$. To measure the probabilistic accuracy of each tracker we compute:

$$P(\hat{x}_{k+1}) = N(\hat{x}_k, P_k). \tag{2}$$

This equation yields a measure of how well the current state estimate and the previous state estimate correlate to

**Figure 2**. *Left: the bright image, Center: the dark image, Right: Difference Image (contrast enhanced)*

the covariance of the model. If this result is near 0, it means that the tracker is not tracking well because the state changed significantly compared to the covariance between the two frames. If this result is near 1, it means that the tracker is tracking well. As a result, we can compute:

$$M = 1 - P(\hat{x}_{k+1}). \tag{3}$$

$M$ gives us a sense as to whether the region is moving as a pupil that is attached to a head should move. If it is near 0, it implies that the region is stationary because its state vector is varying very slowly. If it is near 1, it implies that the region is moving like an eye because its state vector is varying rapidly.

The strength of the tracker does not affect the above equation because the state estimates in the previous and current frames are what are being compared. Even if we were using a perfect tracker, this equation would still hold because the state must either change from frame to frame or remain stationary. In both cases, the formulation for $M$ is correct.

# 4. Appearance based models for candidate regions

Texture information must be taken into account because a region can resemble a pupil in the difference image and can move like a pupil, but could in reality be a moving reflective surface. For example, in Figure 3 we see a set of regions in one frame that are being tracked. Those regions are being tracked because they passed our adaptive thresholding pre-processing and comply with our motion model. However, the majority of the regions that are being tracked are specular reflections from the glass of water, which we are not interested in.

We perform appearance based matching using principal component analysis (PCA). We create two vector spaces as done in [12]: one for eyes, with 85 training images of pupils from the dark image, and one for non-eyes, with 103 training images of patches that are not eyes from the dark image. We take each candidate region, form a vector from its texture, and project it into both of the spaces. The vector is then classified as belonging to the space it is closest to. This class information is then added to the information being kept for the region.

The problem with PCA is that the distances to the spaces are not in any particular scale. This makes it hard to combine the resultant texture analysis with the tracking information we are already maintaining for each region. Combining these helps in deciding whether a particular region is an eye or not. Also, if we just use the distances, we have no measure of confidence that something belongs to the eye or non-eye vector space. To alleviate both of these problems, we use probabilistic principal component analysis (PPCA) [11, 6].

## 4.1 Probabilistic PCA

Probabilistic principal component analysis (PPCA) frames the distances in a PCA vector space probabilistically. PPCA treats all of the training data as points in a probability density with Gaussian distribution. We use PPCA because we want a means to get a probability for a particular region; that is, we want to know the probability that the texture for a particular candidate region belongs to the eye vector space or to the non-eye vector space. If $t$ is the region we are interested in classifying, we want to calculate $P(t)$ for both the eye vector space and for the non-eye vector space. The maximum of these probabilities yields the density that the region is most likely a part of. Bishop, *et al.*provide the full derivation [11] for $P(t)$ by using the fact that PCA is a special case of factor analysis. Here we present a condensed version of their derivation.

Factor analysis is similar to PCA in that $N$ dimensional data is reduced to $D$ dimensional data with $D \ll N$. When dimensionality is reduced with factor analysis, we get:

$$t = Wx + \mu + \varepsilon. \tag{4}$$

This means for an $N$ dimensional $t$, we can reduce it to a $D$ dimensional vector $x$, which represents the latent variables. $W$ are the factor loadings, $\mu$ is the mean of the training set, and $\varepsilon$ is the error, modeled as $N(0, \Psi)$, where $\Psi$ is diagonal. The model for $t$ is also assumed to be normal $N(\mu, C)$ where $C = \Psi + WW^T$. In factor analysis, if $\mu$ and C are calculated, then the probability of $t$ belonging to a particular density is given by $P(t) = N(\mu, C)$.

If the factor loadings were indeed the principal components, one would be able to see the similarities to PCA. However, the loadings are not guaranteed to be the principal components, so we must calculate them. It is pos-

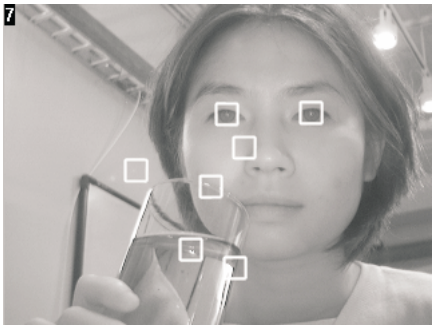

**Figure 3**. *Tracking all candidates*

**Figure 4**. *Classified results*

sible to calculate them because in PCA the data is assumed to have a systematic component, an error term for each variable, and common variance $\sigma^2$. If $\Psi = \sigma^2 I$ and $S = WW^T + \sigma^2 I$, where S is the covariance of the training data, then PCA can be treated as factor analysis. Therefore, $W = E\sqrt{(\Sigma - \sigma^2 I)}R$, where $E$ are the eigenvectors of the training data, $\Sigma$ is a diagonal matrix with the eigenvalues, and R is an arbitrary rotation matrix (since $WW^T$ eliminates the $R$). Because we are concerned with maximum likelihood, $\sigma^2$ ends up being the average of the lost variance or the average of the eigenvalues of the unused eigenvectors:

$$\sigma^2 = \frac{1}{N-D}\sum_{i=D+1}^{N}\lambda_i. \quad (5)$$

All of these equations allow us to calculate $P(t)$ given PCA information for a set of training data. For a given candidate region, the probability that it is an eye or not an eye can now be determined. Since these probability densities are in high dimensional space, the probabilities end up being very small, so they are normalized to be in the interval between 0 and 1.0.

## 5. Classifying Candidate Regions

PPCA gives us probabilities of candidate regions being eyes or non-eyes. This is not enough as the probabilities are temporally unstable and depend on the regions being centered closely around the pupils. Since the region positions are being driven by the results from the Kalman trackers, they are not necessarily exactly over eyes at a given time. Moreover, if classification is done just for one frame, no temporal information is used. This is undesirable because the tracker may have drifted off slightly and would be misclassified as a result. Clearly, we would also like to incorporate the movement information that we are calculating for each frame coming from the Kalman tracker's covariance matrix.

For a particular candidate region $t$, we combine all of the modalities in the following equation which is a weighted probability of all of the statistics we have at time $i$:

$$P_i(t) = \alpha P_{eye}(t) + \beta P_{noteye}(t) + \gamma M + P_{i-1}(t), \quad (6)$$

where $\alpha$, $\beta$, and $\gamma$ vary on the confidence of the results from the PPCA components and the Kalman tracker respectively, $M$ (described in section 3) is a measure of whether the region is moving like an eye, and $P_{i-1}(t)$ is the previous weighted probability of the particular region and has an exponential dropoff. All regions have all of these statistics initialized to 0 in the first frame. The motivation of this equation is that if the trackers are somewhat off-center and the probabilities that a region is an eye or non-eye are close, $\gamma$ will be increased and $\alpha$ and $\beta$ decreased by the system. Conversely, if the movement information is not very helpful, the system relies more on the PPCA component. Temporal information is also included so that single instances of misclassification from the PPCA components do not bias the final classification. Once $P_i(t)$ is calculated, if it is $> 0.5$, we classify the region as an eye for this frame; if it is $\leq 0.5$ we do not. Probabilities were computed for our different modalities so that they could be combined easily and automatically at this stage without heuristics.

Figure 4 shows a frame from live video of a user doing some movements in front of the camera. Notice that only eyes are detected; even the specular reflections from the glass do not fool the system. While the appearance of the glass is indeed similar to a pupil due to specular reflections, it is not classified as such because the dynamics for the region do not match the motion model and the appearance is not consistent over time with that of an eye.

## 6. Results of Eye Tracking

Our method yields a robust pupil detector which detects pupils very easily, minimizing false positives. False positives occur very rarely, but as a result of incorporating temporal information into our classifier, they always decay out quickly (within five frames or less). False positives occur in the system when something reflects light back like an eye, moves slowly like an eye, and actually looks like an

eye. For false positives to occur, all three of these conditions must occur and hold over a large number of frames, which is rare.

Our implementation runs very fast. On a single processor Pentium II 200mhz, the system runs at about 25 frames per second at an interlaced resolution of 640x480 (320x240 each for the bright image and the dark image). On a dual processor Pentium II 200mhz, the system runs at about 29 frames per second at the same resolution. This is a result of having our system largely parallelized to take advantage of multiple processors and to fully exploit the processing pipeline.

Several experiments were performed to test the reliability of the pupil regions that are detected and tracked. For detailed evaluation, we recorded two sequences of 30 sec. of video at 30fps. The first sequence had slow head movements with small out of plane rotations. The second had fast head movements with large out of plane rotations. For each frame in each sequence, the left and right eye positions were manually determined for comparison. Both sequences consisted of one user sitting in front of the camera. The results of these experiments can be seen in tables 1 to 4.

In table 1, the RMS error is only slightly better with our method. The reason for this is that for small head movements and out of plane rotations, the trackers can track fairly well. However, in the faster sequence, shown in table 2, the RMS error is significantly smaller with our method when compared to Kalman tracking with adaptive thresholding. Our method is 0.5 worse than just Kalman tracking in the case of the right eye's x value, but this is most likely due to unsteady hands when locating the eyes for ground truth.

Also of note are tables 3 and 4, which show that when using Kalman tracking with adaptive thresholding there are about double the amount of detected regions compared to the expected number of eyes (2) in the image. Our method

| **RMS error** (900 frames) | **Kalman tracking** | **Weighted prob. (Our method)** |
|---|---|---|
| Left eye x | 3.06888 | 2.13689 |
| Left eye y | 1.51138 | 1.49021 |
| Right eye x | 2.52898 | 2.02244 |
| Right eye y | 1.46873 | 1.42833 |

**Table 1.** *Seq. 1: Slow and small head movements*

| **RMS error** (900 frames) | **Kalman tracking** | **Weighted prob. (Our method)** |
|---|---|---|
| Left eye x | 17.56796 | 13.53245 |
| Left eye y | 10.27857 | 6.10000 |
| Right eye x | 17.31363 | 17.89213 |
| Right eye y | 12.78411 | 5.75922 |

**Table 2.** *Seq. 2: Fast and large head movements*

| **Detected eyes** (900 frames) | **Kalman tracking** | **Weighted prob. (Our method)** |
|---|---|---|
| None detected in | 0 frames | 0 frames |
| Average detected | 3.95540 | 2.10813 |

**Table 3.** *Seq. 1: Slow and small head movements*

| **Detected eyes** (900 frames) | **Kalman tracking** | **Weighted prob. (Our method)** |
|---|---|---|
| None detected in | 0 frames | 81 frames |
| Average detected | 4.89608 | 1.80335 |

**Table 4.** *Seq. 2: Fast and large head movements*

on average detected close to the correct number of eyes. It did not detect eyes in the fast test case for a small portion of the frames (81/900) due to its inability to exploit temporal information since the head movements are very jerky and extremely out of plane.

In looking at all of the tables together, we see that our method tracks with a higher accuracy than Kalman tracking with adaptive thresholding and also yields more meaningful regions. Our method consistently finds the right number of pupils in the scene, and as such, is a better foundation for higher level processing.

## 7. Pairing Candidate Regions

Pairing regions with high probabilities of being pupils into faces is valuable since face finding is the first step in any other higher level processing. Since our pupil list is reliable, we can pair pupils very reliably as well.

We use a facial appearance model to classify faces. PPCA is used by creating a vector space of a set of the upper quarter of 165 faces at a low resolution (20x11). Each face training sample consists of the bounding rectangle around the left and right eyes with a small amount of space above and below the eyes as shown in Figure 5. For every pair of pupils in our list of candidates at each frame we figure out all possible pairings such that a priori information on interocular distances is satisfied. We then find the affine warp to normalize the face candidate's texture region so that the left and right pupil positions line up with the left and right pupil positions of our training data.

Once this is done, we can project the candidate face's texture into the vector space to calculate a probability that the pairing constitutes a face. We find the maximum a posteriori (MAP) set of pairings for our set of pupil candidate regions for each frame. The pairings that result are very reliable even for users with their faces close together or at different head orientations. Mispairings do not occur if only one eye is visible; single eyes and non-eyes are left unpaired. Figure 6 shows an example of the results of our pairing algorithm. As a result of this pairing step, if users

**Figure 5**. *Facial pairing training data*

are facing the camera, we can very reliably find an arbitrary number of faces in the scene. Otherwise, we try to pair unpaired regions again in the next frame.

## 8. Discussion & Applications

There are a large number of interesting applications that can use the methods we have outlined here. We can count the number of people in a scene, identify facial expressions, perform facial recognition, or estimate head pose for multiple people, among many other possible applications. We are particularly interested in the last application as we expect to combine our robust face finding method with texture based head tracking to do real-time multiple person pose detection.

Once eyes are paired off, we can do processing to tell if someone is falling asleep by looking at how the rate of their blinking changes as has been shown possible in [1, 3, 5]. Blinks could be detected and statistics computed to do this by measuring the delays between blinks or whether blinks have stopped and the eyes are in fact closed.

We have shown that for our system utilizing multiple modalities yielded increased robustness. We are interested in exploring whether multiple simple modalities are always better that single "strong" components. Another interesting avenue of future investigation is how the performance of our system would be affected by using better appearance based models, such as utilizing support vector machines instead of PPCA.

## 9. Summary

In this paper we presented a real-time pupil detector and tracker. The system is multi-modal, which adds to its robustness. Since there are plenty of leftover cycles, we can do interesting higher level processing with them. With eyes as robust features, we can find faces, which in turn gives us even more information to use in tracking and classifying pupil regions.

The system has been tested on a number of users each with different eye shapes and skin tones. It was able to consistently locate eyes and faces for single and multiple subjects and was able to reliably track these as well.

Being able to find and track eyes reliably in complex scenes has allowed us to do higher level processing, such as maintaining eye contact and finding faces reliably. We foresee many other applications employing this method.

**Acknowledgments** The authors wish to thank Chakra Chennubhotla and Alex Cozzi for their discussions. Drew Steedly and Gabriel Brostow provided comments on early drafts. Arno Schödl and Wasinee Rungsarityotin provided comments on the final draft.

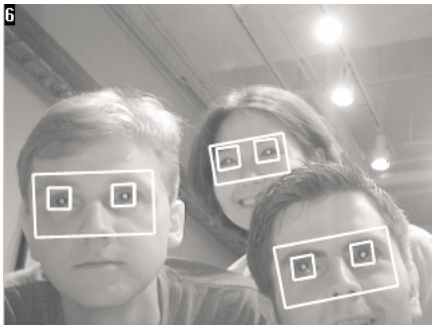


**Figure 6**. *Pairing found pupils from multiple people*

## References

[1] M. Eriksson and N. Papanikotopoulos. Eye tracking for detection of driver fatigue. In *IEEE Conference on Intelligent Transportation Systems*, pages 314–319, 1997.

[2] I. Essa. Computers seeing people. *AI Magazine*, 20(1):69–82, Summer 1999.

[3] M. Funada, S. Ninomija, S. Suzuki, I. Idogawa, Y. Yazu, and H. Ide. On an image processing of eye blinking to monitor awakening levels of human beings. In *18th Annual International Conference of the IEEE Engineering in Medicine and Biology*, volume 3, pages 966–967, 1996.

[4] R. Kothari and J. Mitchell. Detection of eye locations in unconstrained visual images. In *ICIP96*, page 19A8, 1996.

[5] S. Kumakura. Apparatus for estimating the drowsiness level of a vehicle driver. U.S. patent no. 5786765.

[6] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *International Conference on Computer Vision*, pages 786–793, 1995.

[7] C.H. Morimoto, D. Koons, A. Amir, and M. Flickner. Pupil detection and tracking using multiple light sources. Technical Report RJ-10117, IBM Almaden Research Center, 1998. http://domino.watson.ibm.com/library/cyberdig.nsf/Home.

[8] N. Oliver, A.P. Pentland, and F. Berard. Lafter: Lips and face real time tracker. In *CVPR97*, pages 123–129, 1997.

[9] C. Rasmussen and G. Hager. Joint probabilistic techniques for tracking multi-part objects. In *CVPR98*, pages 16–21, 1998.

[10] B. Scassellati. Eye finding via face detection for a foveated, active vision system. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[11] M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[12] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuro Science*, 3(1):71–86, 1991.

[13] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report 95-041, University of North Carolina, Department of Computer Science, 1995.