Eye Detection and Gaze Estimation

Max Praglin
Department of Electrical Engineering
Stanford University
Stanford, CA
mpraglin@stanford.edu

Bryant Tan
Department of Electrical Engineering
Stanford University
Stanford, CA
arctanb@stanford.edu

***Abstract*—A technique for eye detection and gaze estimation was developed in MATLAB for the ubiquitous cases in which a user sits in front of a camera and screen. Such a technique could enable computer interfaces that determine the user's input from low-effort eye movements that naturally express user intent. Furthermore, applications such as driver fatigue assessment, surveillance, or advertising would benefit from such technology. Eye location in images is a previously solved problem; however, "black box" eye detection techniques do not allow us flexibility in and understanding of the core functionality required for a gaze-driven interface.**

To assess the effectiveness of this technique, two experiments were performed. First, eye detection (localizing the centroid of the iris and pupil) was performed on images from the training set with 50% success in finding both eyes with zero false positives, 94% success in finding at least one eye of the subject, and 17% success in finding both eyes of subjects not wearing glasses. Second, gaze estimation was performed on subjects viewing known points in a room.

I. Introduction

Substantial value can be gained from identification of where on the screen a user is looking. Our motivation for developing a technique of eye detection and gaze estimation is primarily derived from interest in a novel interface for laptop computers. This interface would allow a user to control the computer through simply looking at features of interest, and perhaps blinking to express action.

Applications such as driver fatigue assessment, surveillance, advertising, sociological studies, etc. would benefit from this technology. For example: a car could safely come to a stop should the driver experience a microsleep. Law enforcement personnel could direct their attention to subjects prolongedly looking at an unexpected object. An advertising company could target ads based on what catches the user's attention. A map of implied social interaction could emerge from inferring which people in a room look at each other, but do not physically make contact.

The crux of a novel gaze-driven interface is, not surprisingly, eye localization and gaze estimation. The technique explained in this paper estimates a subject's gaze direction after calibrating with an image of the subject looking straight ahead. The subject need not be a member of the set of images used to train the eye detection module.

II. Methodology

We made the following assumptions about the problem setup:

- The subject is directly facing the camera, and is near the center of the image
- We have a 'calibration' image of the subject looking straight ahead

We denote the centroids of the eyes in the calibration image as the 'calibration eyes', and the centroids of the eyes in the test image as the 'test eyes'.

The core of the gaze estimation algorithm is eye detection. Once the pixel locations of the test eyes have been identified, it is straightforward to compute the vectors between the calibration eyes and the test eyes, thus determining a vector proportional to $\begin{bmatrix}\sin\theta_x & \sin\theta_y\end{bmatrix}^T$.

A. Eye Detection

Our overall strategy was to use MAP detection based on pixel color to recognize irises. Color methods have been successfully employed in the past to find irises [1]. We made a modification to MAP, however, in which we normalize every pixel's RGB vector to have unit length, making the detector invariant to changes in lighting conditions.

We constructed a training set based on 50 professional portrait photos by manually indicating the eye areas using MS Paint for each training image.

Using this training set, we trained a MAP detector with 16 buckets in each of the three color dimensions: our algorithm simply counts the number of positive and negative pixels that fall into each bucket, and sums these counts over all the training images.

Since eyes represent such a small proportion of the pixels, this data set is skewed. Further, since each training image represents a different eye color, the threshold for a positive detection needs to be very low. We produced the ROC curve (see Fig. 1) and plotted precision against threshold in order to determine a reasonable threshold to use.

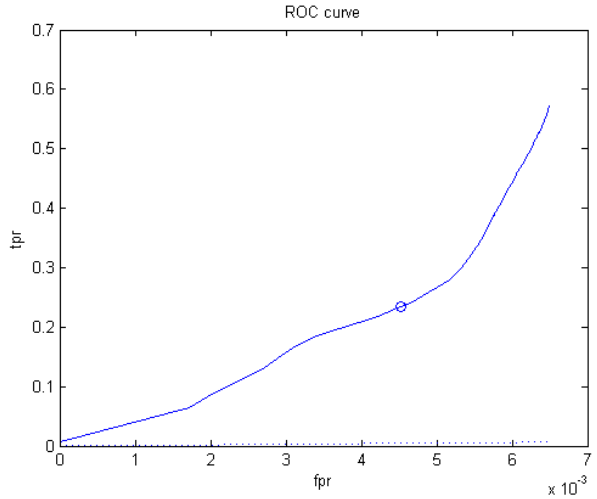

Fig. 1. ROC curve for threshold selection.

We note that the ROC curve's axes are not equal, and the dotted line near the bottom of the plot represents the line of unit slope. See Fig. 2.

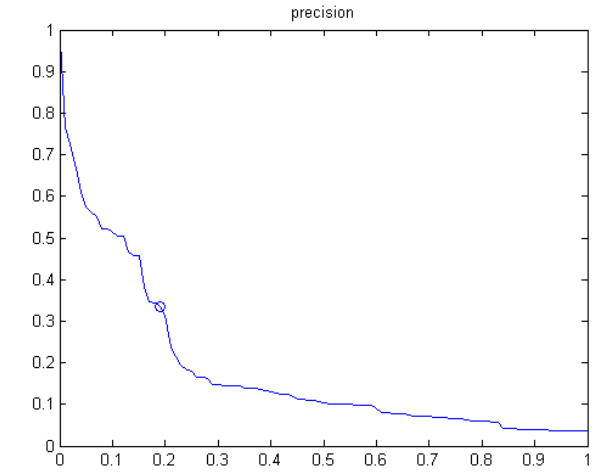

Fig. 2. Precision against threshold. We chose a threshold of 0.19.

We found while testing using various thresholds (both above and below the final chosen threshold) that false positives were more difficult to deal with, and more prevalent, than false negatives. We chose a threshold corresponding to a precision of around $\frac{1}{3}$. Increasing the threshold slightly beyond our chosen threshold corresponds to a precipitous drop in precision, so this represents the largest threshold that seemed feasible.

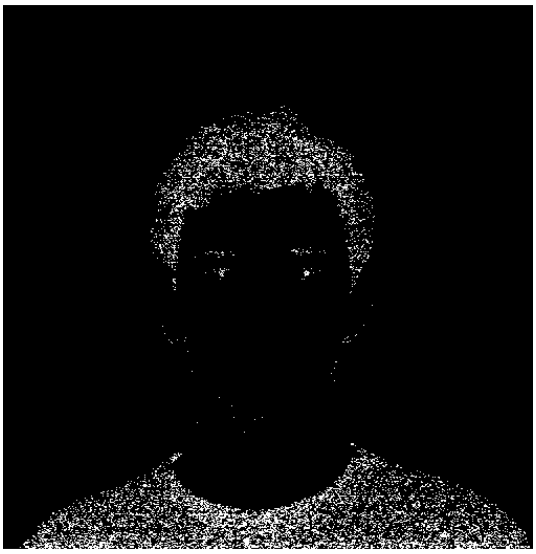

Fig. 3. MAP detection. There are many false positives, most of which lie outside of the facial region.

Even using this threshold, we found MAP produced too many false positives.

We found, however that most of the false positives were associated with hair, clothes, and other regions outside the face, as shown in Fig. 3. This observation motivated the next step in our algorithm, in which we exclude from consideration MAP detected points outside a region detected as the face (see 'Face Detection' below). The result of excluding points outside the face region is shown in Fig. 4.

Fig. 4. MAP detected points that lie within the facial region.

The final step is to eliminate all other points that do not represent eyes. We assume the detected pixels that correspond to eyes represent the largest two connected regions. We repeatedly remove connected regions smaller than a threshold, for increasing values of the threshold, until there are only two connected regions left. Fig. 5 shows a plot of the number of connected regions as a function of the threshold for connected

region size. Fig. 6 shows the result of this thresholding process.

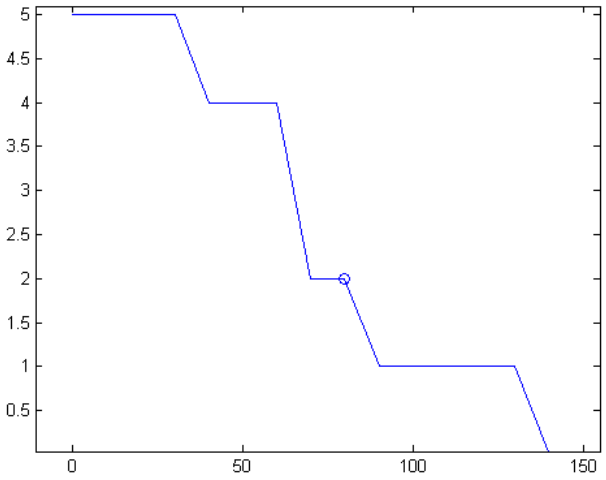

Fig. 5. Number of connected regions as a function of threshold.

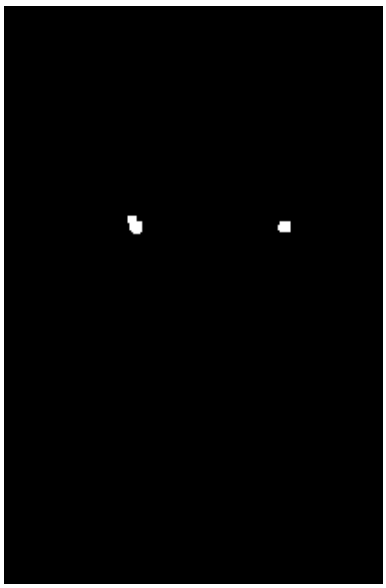

Fig. 6. The two largest connected regions, deemed to be eyes by our algorithm.

Finally we compute the centroids of those two regions. These are the detected eyes.

In summary:

1) MAP training.
2) MAP detection.
3) Mask out non-face regions.
4) Remove small connected regions until only two remain.
5) Compute the centroids of these connected regions. These are the two eyes.

B. Face Detection

We make the additional assumption that the face is reasonably centered in the image, and occupies the majority of the center square of a 3×3 grid overlaid on the image. Our algorithm is related to the methods presented in [2], and as in that paper, our primary objective was speed since we do not require extremely high precision or object tracking.

We make the observation that faces are relatively homogeneously colored. Therefore the face can be detected as the set of pixels that are close in color-space to the mean of the center square color. Our algorithm for face detection, therefore, is as follows:

1) Crop the image to only the center $\frac{1}{3}$ in both dimensions.
2) Compute the mean color
3) Mark all pixels that are within a certain distance in color-space of the mean color in the image.
4) Erode the image with a small square to remove spurious points.
5) Compute the convex hull of the remaining points. This encloses the face.

Fig. 7. Set of pixels sufficiently close to the mean face color.

Fig. 8. Convex hull of detected points plotted over the face.

An advantage of this method is that it ignores features outside of the facial region as defined by color features. This means hair and clothes will not appear in the considered region.

C. Gaze Estimation

Since the test image may show a face that has been moved, rotated, or scaled with respect to the calibration image, we need to compute a mapping from the test image to the calibration image. To this end, we chose SIFT combined with RANSAC and a homography model. Since some webcams have wide-angle viewing capability (and thus introduce distortion), and a face may move around considerably within the viewing range, we must account for resultant skew or apparent perspective changes that may result from that distortion, hence our choice of a homography model. Further, as noted in [3], many parts of the face (other than the nose) can be considered to lie in a plane. A homography is therefore a reasonable model to apply to eyes under facial rotations.

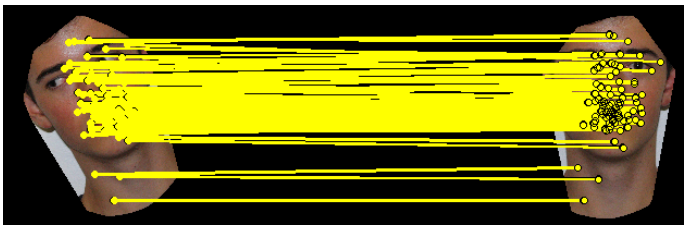

Fig. 9. Inliers computed with SIFT features, and RANSAC with a homography model.

SIFT computes correspondences between key points on the face, which we post-process to extract a homography. Using this homography, we can map eye locations in the test image to the corresponding eye locations in the calibration image. Eye detection on the calibration image yields calibration eyes, and we can now compute the vectors between calibration eyes and test eyes.

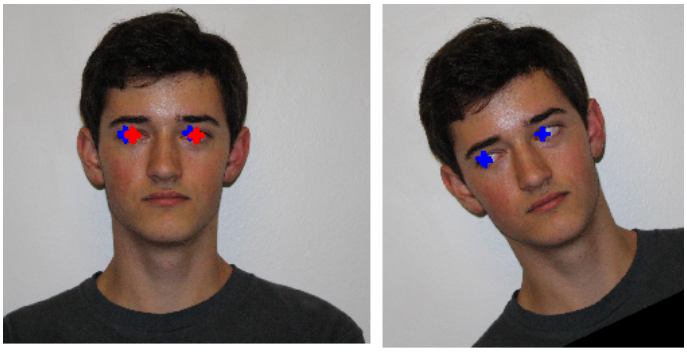

Fig. 10. Mapping detected eyes from a (deliberately) rotated test image back onto the calibration image. Test eyes (right). Calibration eyes (left, red) with test eyes (left, blue) mapped onto the calibration image axes, and overlaid.

In summary:

1) Locate test eyes and calibration eyes.
2) Use SIFT + RANSAC + homography model to compute a homography from the test image to the calibration image
3) Use this homography to map test eyes onto the calibration image.
4) For each calibration eye, compute the vector to the closest test eye (reject if there is no sufficiently close test eye). This is closely related to the gaze angle.

If $\begin{bmatrix}\Delta x & \Delta y\end{bmatrix}^T$ is the average vector from calibration eyes to test eyes, then $\Delta x \propto \sin\theta_x$ and $\Delta y \propto \sin\theta_y$.

III. Results

We assessed the accuracy of gaze estimation by photographing a subject instructed to look at specific points in a room, both at relatively small and large angles from the line between the camera and subject's face. These points were measured with respect to the camera - this enabled a comparison between the actual gaze direction and the gaze angle estimated by our algorithm. The technique's success is shown in Fig. 11.

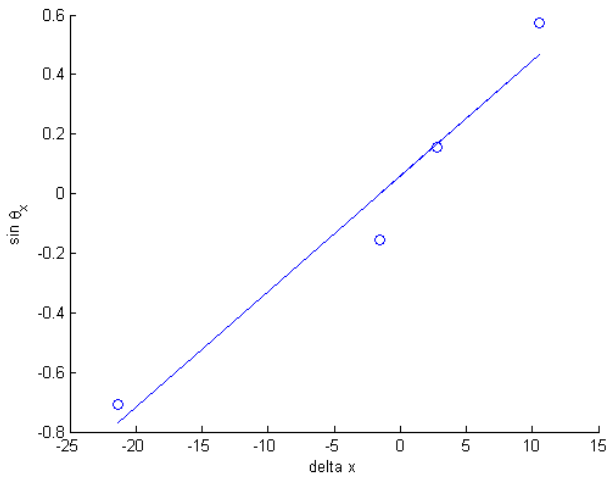

Fig. 11. Plot of Δx (x-component of test-eye to calibration-eye vector) against $\sin\theta_x$ (actual gaze angle). The algorithm correctly detects whether the gaze angle is to the right or left, even for small $|\theta_x|$.

The MATLAB profiler was used to identify the time spent in each section of the eye detection and gaze estimation workflow:

23.8%	SIFT (for computing homographies)
21.7%	eye detection (in test image)
20.7%	eye detection (in calibration image)
16.5%	housekeeping (i.e. image reads)
Other	gaze calculation, glue code

To correctly detect both eyes in a 1944x1296 image of a subject, the aforementioned eye detection technique required approximately 6 seconds on a single core of an i7 processor. It is worth noting that gaze estimation accounts for a trivial fraction of instructions after eyes have been detected.

We were unable to test our algorithm on a large data set that was not the training set, since we did not have the time to compile a large collection consisting of many faces, each with several different gaze angles, in the lighting conditions and quality under which our algorithm was designed.

Since eye detection sits at the core of the algorithm, we conducted tests to determine its accuracy. To this end, we ran our eye detection algorithm on each of the 50 training images and visually inspected the results, tabulated as follows:

	total #	✓	✗	success rate
Entire data set*	50	47	3	94%
Entire data set†	50	25	25	50%
Non-glasses†	44	24	20	55%
Glasses†	6	1	5	17%

* = only one eye detected
† = both eyes detected

Our algorithm only considers test eyes near the calibration eyes, and takes the average of the vectors if both eyes are found. Therefore, for our algorithm to work correctly, we only need to find one of the two eyes (though it must be the same eye in both calibration and test images).

IV. Conclusion & Remarks

A few immediate improvements could be implemented with relatively little additional development time:

- Use homography from calibration image to test image to only look for eyes in that region of the face. This would improve the algorithm's speed and potentially vastly improve accuracy.
- Experiment with eroding the face mask by various amounts to eliminate false positives arising from an excessively large detected face region.
- Compute the centerline of the face so a calibration image is not required.

The technique developed in this final project was motivated by the tools taught in EE368. We believe that the following improvements related to techniques not covered in the class, which were suggested by attendees of the EE368 Poster Session, would be worthwhile next steps for bettering our technique:

- Local Binary Patterns for facial descriptors [4].
- Descriptor Quantization to include color information in keypoint descriptors [5].

We observed that in general, gaze estimation is more successful in the horizontal direction than the vertical direction. We suspect that the predominantly horizontal shape of the eye contributes to this effect.

References

[1] M. K. Monaco and M. K. Monaco, "Color space analysis for iris recognition by," 2007.

[2] H. Yin, P. Fu, and S. Meng, "An efficient face detection method in color images," in *Knowledge-Based Intelligent Information and Engineering Systems*, ser. Lecture Notes in Computer Science, R. Khosla, R. Howlett, and L. Jain, Eds. Springer Berlin Heidelberg, 2005, vol. 3684, pp. 875–880. [Online]. Available: http://dx.doi.org/10.1007/11554028_122

[3] O. Boumbarov, S. Panev, I. Paliy, P. Petrov, and L. Dimitrov, "Homography-based face orientation determination from a fixed monocular camera," in *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*, vol. 1, Sept 2011, pp. 399–403.

[4] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 12, pp. 2037–2041, Dec 2006.

[5] M. Pedone and J. Heikkila, "Local phase quantization descriptors for blur robust and illumination invariant recognition of color textures," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, Nov 2012, pp. 2476–2479.

Appendix

We completed the vast majority of work for this project in group meetings, and as such, are both equally responsible for all aspects of the project. Algorithm brainstorming was a joint effort, we pair-programmed, and each created half of the training and test image sets.