Head-mounted gaze tracker

Cornell University
School of Electrical and Computer Engineering

Anil Ram Viswanathan, Zelan Xiao

Advisor: Dr. Bruce Land

Introduction

An eye-tracker identifies the focus of a person's gaze within his/her field of vision. Eye tracking has been used in research to identify the visual behavior of individuals performing a variety of tasks, and in answering a number of questions -

- What do shoppers look for in a shopping aisle?
- What does a soccer goal-keeper look at while defending a penalty kick?
- What do infants focus on, and how does this affect their learning?
- Where do you put content on a website so that it catches the eye?

Eye trackers come in all flavors of shapes and sizes–

- There are rigidly controlled laboratory set-ups that require the subject to be constrained using chin rests
- Some use remote video cameras to track eyeball movement
- Some are mobile, head-mounted units.

Most of these units are intended for specialized applications, and are consequently, quite expensive. The goal of this project was to build a ***light, portable***, and most importantly, ***inexpensive*** system that can provide reasonably accurate eye tracking.

Head-mounted unit

The Head-mounted unit essentially consists of a couple of (wireless) cameras -

- One focused on the eye
- One looking out at the scene.

The cameras simply transmit the video to the base station. The base station has radio receivers connected to USB video input devices, allowing the PCs to treat them simply as USB cameras.

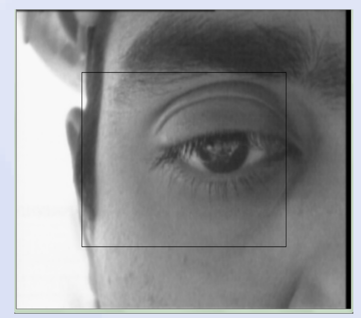
Bill of materials

One of the primary goals of the project was to develop an inexpensive system. Our costs are:

Wireless cameras	: 2 x $ 75 = $ 150
Hauppauge composite-to-USB converters	: 2 x $ 50 = $ 100
Hard hat	: $ 25
Total expenses:	*$275!*

Finding the eye position

Finding the position of the eye involves a series of operations on the eye image. The software pipeline used is based on OpenCV, a powerful open-source library of image-processing functions. This implementation combines two techniques – we first detect an approximate eye position, and then refine it using a circle Hough transform.



Capture eye video

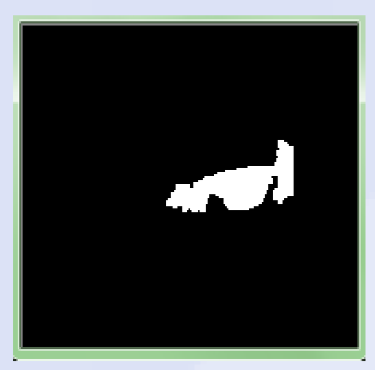
Convert to gray scale
Run a Gaussian filter to remove high-frequency noise.

Detect eye-analogues

Input: Expected eye region size (width, height)

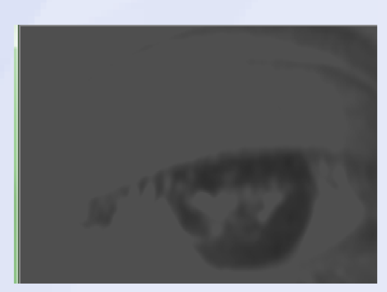
Eye pixels are *darker* than surrounding regions. This is the basic premise. If a pixel is darker than 6 of its neighboring regions, it is considered to be an *eye-analogue* pixel.

This operation may generate several possible eye-analogue regions!



Reduce to a single region

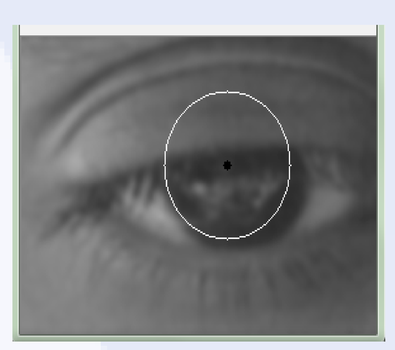
- Identify *blobs* – perform connected-component labeling.
- Remove blobs that are too wide or too tall – compare the height/width of the blobs with the expected eye window size.
- Remove blobs that are too small – again, compare the height / width of the blobs.
- Remove blobs that do not have the expected aspect ratio.



Circle Hough Transform

A Hough Transform identifies specific shapes in an image. Generally used in conjunction with edge detectors, it operates by having each point on an edge *vote* for the location of a specific feature.

A Circle Hough Transform has each point on the edge vote for the center of a circle (assuming the radius is known). The circle center lies perpendicular to the gradient of the image; if the direction of the edge is known, this reduces to a vote at a single point within the object. The votes may be spread out to compensate for errors.



A Circle Hough Transform is used to identify the center of the iris. With a well-defined boundary, the iris is nearly always easily identifiable. Moreover, the Hough Transform works even if the complete shape is not present; this is the most common case in processing eye images, as the eyelids occlude the iris.

To make sure that excessive noise doesn't affect the transform, we apply a threshold to the eye region. Values greater than the threshold are set to the threshold; those lesser than the threshold are left as is. This again builds upon the assumption that in general, the iris is darker than the rest of the eye!

Finding the gaze point

Finding the eye location is half (albeit the difficult half!) of the picture – we need to map this eye position into a corresponding position in the subject's field of view. This is implemented as a two step process -

- Have the subject look at two (or more) 'known' points on the screen, s & s'; note the corresponding eye points, e and e', and calculate the scale:

  $$scale = (s' - s)/(e' - e)$$

  In this implementation, a bright object (the brightest in the frame) is introduced into the scene, and the subject is asked to look at it. Finding the brightest point in the scene frame is a trivial operation. This is repeated multiple times.
- Apply linear interpolation to convert any eye point to the corresponding scene point.

  $$s_{new} = (e_{new} - e) . scale + s$$

Observations and results

The eye tracker works well enough, if a set of parameters are tuned for each user -

- The *threshold* value, applied before the Hough transform – this is largely dependent on the lighting conditions, and the color characteristics of the subject's eye. A darker eye can afford a lower value of the threshold; a lighter iris requires the threshold to be higher.
- The *accumulator threshold* for the Hough transform – this determines the minimum number of votes a point must gather before it is considered to be a valid circle center;
- The expected radius of the eye – this is fed as another input to the Hough transform.

Even when these parameters are set, we observed that the eye detector would keep 'skipping' around – largely because of noise in the image. To reduce this effect, we introduced a 'low-pass filter' -

$$eyePos_{new} = \alpha . eyePos_{old} + (1 - \alpha) . eyePos_{estimated}$$

This ensures that the estimated position converges. The parameter α is also tunable.

Literature referenced

- The OpenEyes project, at http://thirtysixthspan.com/openEyes/.
- *Efficient face candidates selector for face detection,* Jianxin Wu, Zhi-Hua Zhou, National Laboratory for Novel Software Technology, Nanjing University, China

Acknowledgments

We would like to thank our guide, Dr. Bruce Land, for his advice and support in completing this effort. We would also like to thank Dr. Ron Hoy, Dept. of Neurobiology and Behavior, Cornell University, for his sponsorship of the project.