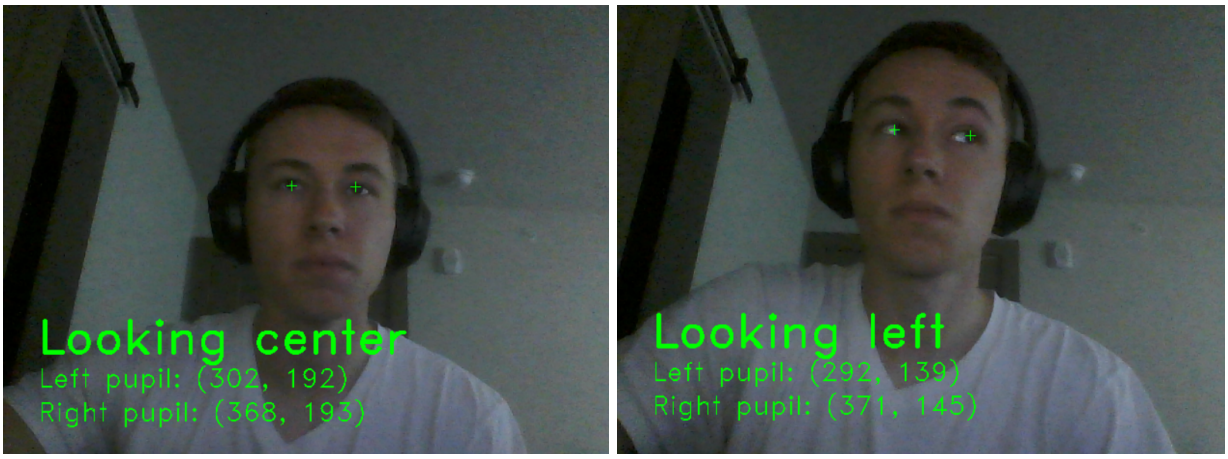
Gaze Tracking

Using Python, OpenCV, and Dlib

John Douglas
CPSC 4820-002
Clemson University
jdougl4@clemson.edu



I. Abstract

Gaze tracking is the process of measuring the point of gaze. Gaze tracking technology is used in research on the human visual system, in psychology, in psycholinguistics, marketing, product design, and countless other mediums. Thus, I've been interested in the topics of eye tracking and gaze tracking since this course began. When I learned we'd have a final project, my immediate goal was to attempt my own version of this as a culminating exercise and a demonstration of all I've learned in the course. The ultimate goal was a program that could detect the eyes of a user via the webcam, and track their direction in real time.

CCS Concepts: Computer Vision. OpenCV. Dlib. Python. Eye tracking. Gaze tracking.

II. Introduction

Dlib is a general-purpose cross-platform software library written principally for C++, but a number of its tools can be used in python applications. This software library is responsible for the development of thousands of machine learning models. Dlib's pre-trained model for facial detection and 68 facial landmarks is used in this project when defining a face object.

III. Pupil Object

The Pupil class's primary function is to detect the iris, and then estimate the position of the pupil. It uses two main methods to accomplish this. The first is an image processing method, which accepts two values: a frame containing the user's eye, and an integer threshold value which is used to binarize the eye frame. Put simply, it uses three OpenCV functions to perform some operations on the passed eye frame to isolate the iris (cv2.bilateralFilter, cv2.erode, and cv2.threshold). The second method is used to detect the iris of the user and estimate the

position of the pupil by calculating the centroid. The centroid is calculated as the arithmetic mean of all the points in a “blob”, where in this case, a “blob” (or the user’s pupil) is defined as a group of connected pixels in an image that shares some common property (e.g., a grayscale value):

$$\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

where $x_1 \ldots x_n$ are the distinct points of the shape.

The center of the blob is calculated using cv2.moments. Image Moment is a particular weighted average of image pixel intensities, with the help of which the centroid can be calculated. The centroid formula is given by:

$$C_x = \frac{M_{10}}{M_{00}}$$

$$C_y = \frac{M_{01}}{M_{00}}$$

where C_x is the x coordinate and C_y is the y coordinate of the centroid, and M denotes the moment.

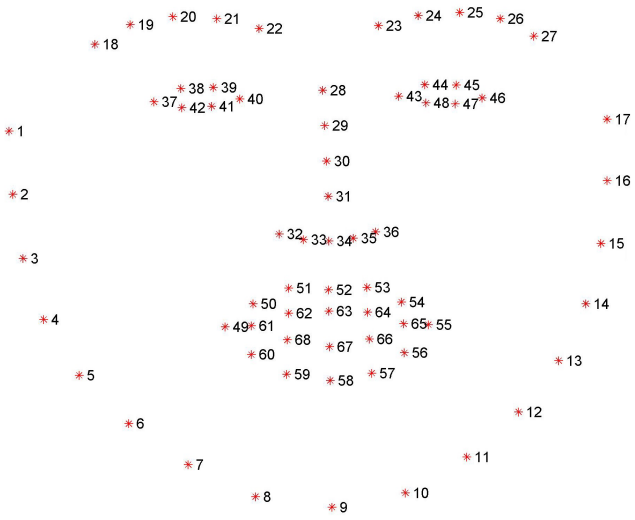
IV. Calibration

The calibration class attempts to calibrate the pupil detection algorithm by finding the best binarization threshold value for the person and the webcam. It has a method to return the percent of space that the iris takes up on the surface of the eye (utilizing the cv2.countNonZero function), a method to calculate the optimal threshold to binarize the frame for the given eye, and a method that attempts to further improve calibration by taking the given image into consideration.

The best threshold for the given eye is calculated by taking into account the size of the average iris, which is about 0.48 inches of 12mm.

V. Eye Object

The eye class’s primary objective is, given webcam input, to create a new frame of the user’s eye. It uses Dlib’s 68 facial landmarks and face detector.

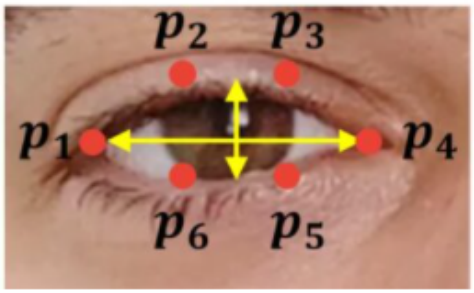


The object does this through the use of a few methods. First, a static method that simply finds the midpoint between two points. This is used for some of the calculations regarding blink detection. Second, an isolation method, to isolate an eye and generate a frame without other parts of the face. This method accepts an instance of self, a frame containing the user's face, landmarks for the face region (dlib.full_object_detection), and a list of points of the eye from the 68 Multi-PIE landmarks trained model. This method applies a mask to get only the eye, and then performs some number operations to crop around the eye.

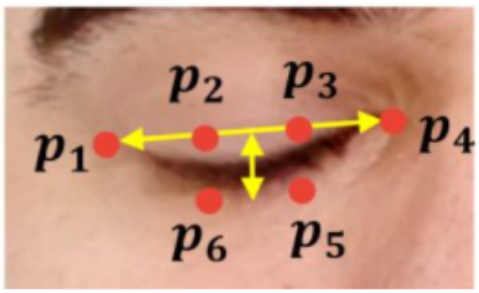
The third method calculates a ratio to indicate whether or not the eye is closed. This is done by taking the division of the width of the eye by its height. It accepts landmarks for the face region (again, of type dlib.full_object_detection) and a list of points of the eye from the 68 Multi-PIE landmarks trained model. To understand how this works, reference the image above of the 68 facial landmarks, specifically the eyes–notice that each eye is made up of 6 (x, y) coordinates, starting at the left corner of the eye, working clockwise around the remainder of the region. The relationship between these points is referred to as the *eye aspect ratio (EAR)*:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

where (for the left eye, for instance) p_2 and p_6 correspond with points 38 and 42 in the above diagram, p_1 and p_4 correspond with 37 and 40, and p_3 and p_5 correspond with 39 and 41. The greater the EAR, the more widely the eye is open. The is_blinking method in the gaze tracking class (later) decides a minimum EAR value as 3.8, which is used to decide whether the eye is closed or not.



Open eye will have more EAR



Closed eye will have less EAR

So, the blinking ratio method computes each eye's width and height, then determines the blinking ratio as the width divided by the height. The middle point method is used to generate a single line between the top and bottom of the eye (the middle of p_2 and p_3 connected to the middle of p_6 and p_5 as shown above).

Lastly, a method to detect and isolate the user's eye in a new frame. This data is sent to the calibration object and initializes a pupil object with values calculated by the other methods.

VI. Gaze Tracking

The gaze tracking class tracks the user's gaze with data from the previous objects. It provides the position of the eyes and pupils, and displays whether the eyes are open or closed.

It first uses the Dlib face detector to detect the user's face, and then Dlib's shape_predictor to get facial landmark points of a given user's face. It then checks that the pupils have been located.

The rest of the methods are mostly small, intuitive methods to calculate information about the position of the user's eyes. There are methods to return the (x, y) coordinates of both pupils, to calculate the horizontal ratio that indicates the horizontal direction of the user's gaze, boolean methods to determine whether the user is looking left, right, center, or is blinking, and then a method to return the main webcam frame with the user's pupils highlighted with a crosshair overlay.

To determine gaze direction, these methods return a number between 0.0 and 1.0 that indicate the horizontal direction of the gaze, where 0.0 is extreme right, 0.5 is center, and extreme left is 1.0. This value is calculated as the horizontal distance of the pupil from its projected center area (with the value from both eyes being averaged and returned). This method is used in the is_left and is_right methods.

References

1. Agarwal V. Real-time eye tracking using opencv and dlib [Internet]. Medium. Towards Data Science; 2021 [cited 2022Apr26]. Available from: https://towardsdatascience.com/real-time-eye-tracking-using-opencv-and-dlib-b504ca724ac6
2. Bapat K. Cv2.moments [Internet]. LearnOpenCV. 2018 [cited 2022Apr26]. Available from: https://learnopencv.com/tag/cv2-moments/
3. Blob detection using opencv (python, C++) | [Internet]. LearnOpenCV. 2021 [cited 2022Apr26]. Available from: https://learnopencv.com/blob-detection-using-opencv-python-c/
4. Boudreau M, Rosebrock A, Sobczak E, Rajendiran R, Haili, Chadha R, et al. Eye blink detection with opencv, python, and dlib [Internet]. PyImageSearch. 2021 [cited 2022Apr26]. Available from: https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/
5. Eroding and dilating [Internet]. OpenCV. [cited 2022Apr26]. Available from: https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html
6. Image thresholding [Internet]. OpenCV. [cited 2022Apr26]. Available from: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html
7. Pandey D. Eye aspect ratio(ear) and drowsiness detector using dlib [Internet]. Medium. Analytics Vidhya; 2021 [cited 2022Apr26]. Available from: https://medium.com/analytics-vidhya/eye-aspect-ratio-ear-and-drowsiness-detector-using-dlib-a0b2c292d706
8. Smoothing images [Internet]. OpenCV. [cited 2022Apr26]. Available from: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html
9. http://dlib.net/face_landmark_detection.py.html