Human Vision Tracking

Uchinta Kumar Boddapati
CS 6375 Machine Learning
Spring – 13

Abstract

The goal of the project is to develop software capable enough to track the vision of human eye on the computer screen. The software makes use of inbuilt camera of the laptop to track the human face and the eye ball movement relative to the head. Using Machine Learning Algorithm KNN, I am able to approximate which part of the screen/monitor the user is watching. Training the algorithm will give more accurate results.

In this report, my approach and technologies used to achieve the requirements are described initially. And a detailed description is provided how to use the software and what to expect from the software. Finally the report is concluded with limitations, future improvements and scope of this software in real world scenarios.

1. Approach

Whenever a user looks at laptop screen, a background application opens the inbuilt camera and starts collecting frames. Using OpenCV library and its ApI, the application can recognize if a user face is present or not in the frames obtained. Once a face is recognized, some mathematic approximation is applied to identify the size and region of the eyes respective to the face. After calculating the region of interest, once again OpenCv functionalities and eye ball color gradient variation helps to pin point the center of the eye ball. This is equal to a (x,y) point on the frame. For each frame, all possible such eye mid-points (x,y) are collected. To map these data points to actual location on the screen, I made use of Machine Learning Algorithm 'K Nearest Neighbour' to do this mapping. As we know any Machine Learning algorithm will have a training phase and a testing phase. Similarly, here the user is requested to go through a training phase to get better results in the testing phase. As a part of the training phase, user is asked to look at different regions of the monitor for a particular period of time 't'. During that period, eyeball center points are calculated and a mapping is associated internally between the obtained eyeball center point and the actual region on the monitor. This process repeats for a comfortable amount of time until we have sufficient number of training data to estimate different regions of the screen. Once the training data collection and mapping is done, user is provided an option to switch to testing mode. Now in this phase, user is free to look at whichever region he wants on the screen. Whenever a user looks at a particular region on the monitor, the application captures the respective frame from camera, process it and obtains the eye ball center point (x,y). Using K Nearest Neighbour algorithm, the obtained point is compared with existing training data to know which region it belongs to. Each region has a specific training sample. So, the obtained test sample distance from each of these training samples is calculated, say the Euclidean distance. And finally the training sample closest to this test sample is selected. Now we know the associated or mapped region to this training sample. Once the region is identified, the application highlights that particular region. Collecting good number of training samples will increase the accuracy in identifying the region of gaze during testing phase.

2. Technologies used

The main task here is to obtain the center of the eye ball from the captured camera frame. This point (x, y) on the image is used as training or testing sample. To achieve this, the captured frame is processed through series of phases to obtain the desired result. As a first step, the captured frame is fed to OpenCV algorithm to identify human face. OpenCV existing ApI tracks the face and updates our application with the bounding rectangle. Once the bounding rectangle of the face is obtained, it is fed to another phase called 'Identifying the eyeball'. During this phase, an approximation is calculated to identify a bounding rectangle for both the eye balls. Eye balls will be in the upper half of the face rectangle is one such approximation to mention. And then based on an approximation of distance between the eyes, a bounding rectangle for each eye is calculated. Once it is obtained, the image is processed to obtain eye center. A general assumption that eye color will be in black color is made here. So the bounding rectangle for the eyeball is processed for color change gradience. Gradient vectors are calculated in horizontal and vertical directions and a point of intersection for these vectors is estimated as eyeball center. Thus, a series of phases helps to achieve the actual eye ball center from the captured frame which is nothing but our training/testing sample data point. These data points are given as input to the KNN algorithm. Once the application enters the testing phase, the trained KNN estimates region of gaze based on the input test sample. An association between region of gaze and training sample data point is maintained internally. This association helps to highlight the region of gaze on the monitor when needed.

3. Requirements

In order to execute the program, eclipseIDE, OpenCV library and windows/Linux OS are required. The application project settings should be configured to include 'inc' and 'libs' related to OpenCV. Once the setup is properly done, executable should be generated with out any issue.

4. Execution

Once the executable is generated, running the program is straight forward. Once the application is executed, it previews captured camera frames in a window with horizontal and vertical white lines which bisect the entire preview in to MxN blocks. If user is facing the camera, he can see a bounding rectangle surrounding his face. During this phase, the application is neither in testing phase nor training phase. User explicitly needs to key-in the character 't' to enter 'Training phase' or character 's' to enter testing phase. If he is in training phase, a decrementing counter from '5' to '0' in red color is displayed in a random block. User MUST gaze at that counter without any interruptions like moving out of camera view or closing eyes. Once the counter reaches '0', the countdown will start in another randomly selected block. This process is repeated for all the MxN blocks. Once the application collects training data for all the blocks, it displays a message "Training completed". Now user is ready to enter testing phase. Once he enters testing phase, he can gaze at any region/block of his own wish. The application highlights that particular block in color to mention that user is watching there. If a particular user trains the application once, he no need to train for succeeding execution of the program. He can directly enter testing phase. Training is needed only if a user wants to test the application for the first time.

5. Limitations

Considering time frame of the project implementation, certain limitations are enforced or the results are compromised to certain extent. **1.** User should make sure that the bounding rectangle identifying his face should be in the center of the captured frame. And this bounding rectangle or the user face should not tilt or move during testing or training phase. Else the results will be not correct. **2.** Actual point of gaze is compromised to actual block of gaze to simplify the implementation of the program.

6. Future Improvements

As there are certain limitations in the current implementation, the program can be extended to handle them appropriately. The restriction estimating block can be removed and actual point of gaze on the screen can be calculated with lot more training data.

7. Scope

The scope of this application is used by many technology companies. Using this software, online advertising companies are gathering user details like where a particular user is watching on the screen frequently. Depending on this data, customized advertisements are placed at those locations to catch attention of the user.

8. References:

1. *Fabian Timm and Erhardt Barth - Accurate eye centre localisation by means of gradients*