openEyeTrack - A high speed multi-threaded eye tracker for head-fixed applications

Jorge (Paolo) Casas Chandramouli Chandrasekaran

11 July 2019

Department of Biomedical Engineering, Boston University, 02215

Department of Anatomy and Neurobiology, Boston University, 02118

Department of Psychological and Brain Sciences, Boston University, 02215

Statement of Need

When faced with a decision, an organism uses information gathered by their senses in order to determine the best course of action. Vision is one of the primary senses and tracking eye gaze can offer insight into the cues that affect decision making behavior. Thus, to study decision-making and other cognitive processes it is fundamentally necessary to accurately track eye position. However, they are often very expensive and incorporate their own proprietary software to detect the movement of the eye, which limits the researcher's ability to be fully informed regarding the ongoing processes within their experiment and incorporate modifications tailored to their needs. Here, we present our software solution, "*openEyeTrack*", a low cost, high speed, low latency, open-source video-based eye tracker (Casas and Chandrasekaran 2019).

Software and Hardware components

openEyeTrack takes advantage of OpenCV (Bradski 2000), a low cost high speed infrared camera and Gige-V APIs for Linux provided by Teledyne DALSA (Teledyne DALSA 2018), and the graphical user interface toolkit QT5 (The Qt Company Ltd. 2013), all of which can be downloaded for free. The only costs are from the hardware components such as the camera (Genie Nano M640 NIR, Teledyne DALSA, ~$450, ~730 frames per second) and infrared light source, an articulated arm to position the camera (Manfrotto: $130), a computer with one or more gigabit network interface cards, and a power over ethernet switch to power and receive data from the camera. By using the Gige-V Framework

to capture the frames from the DALSA camera and the OpenCV simple blob detector, "openEyeTrack" is able to accurately calculate the position and area of the pupil. Pupil size has been linked to arousal levels and can offer insight to the emotions of the subject. Video based eye trackers can perform nearly as well as classical scleral search coil based methods and can be used for most applications (Kimmel, Mammo, and Newsome 2012).

Multithreading provides improvements over existing open source solutions

openEyeTrack is based on other open-source eye trackers currently available such as "Oculomatic" (Zimmermann et al. 2016). However, most of these programs are single threaded: the frames are captured, processed, and displayed sequentially, only executing the next stage once the previous stage has been completed. Although single threaded methods have become more effective over the years, these stages are time consuming and can limit the overall performance. In order to increase performance "openEyeTrack" was developed as a multithreaded application. The capture, display, data transmission, and most importantly, processing components all happen within their own separate threads. By incorporating multiple threads, the processing speed of the frames is able to match the frame capture rate of the camera, allowing for lossless processing of data.

Algorithm

As depicted in **Figure 1** below, as frames transition between the captured, processed, and display processes, they are stored in queues which enable the different processes to run independently and allow for asynchronous capture, detection, and display. Once the camera grabs a new frame, it is very briefly stored in the Genicam memory buffers before being extracted and packaged by the "capture thread" into a struct and stores in a queue. This approach allows for the sequence of acquisition frames to be preserved and the frame acquisition process to take place without being slowed down by processing or displaying. The frames in the "capture queue" are popped off by the "n" (user specified) processing thread(s). Each processing thread takes the data from the "capture queue," converts it into an OpenCV Mat object, applies the OpenCV blob detection algorithm, notes the key features, and outputs the position of the blob as text on the frame and draws a circle around the blob. This process is very time consuming, which is why initializing multiple threads are recommended for higher performance. Once the final, processed images are ready, the processing threads store them into a display queue that the display thread will grab from to show the images. The processing threads also packages the frame and keypoints information into a struct object which is then stored in a "network queue". The "network thread" reads from this queue and sends out data over a UDP socket for downstream applications.

Performance

Under the conditions at the time of development, frame acquisition frame rates of up to 715 fps and display rates of up to 145 fps were available. Although more threads in theory should speed up the processing, four processing threads were sufficient to keep up with the camera. We found that performance improved when we used the gev_nettweak tool provided by Teledyne Dalsa, which adjusts various features for the network buffers allowing higher throughput transmission from the camera to the computer. Additionally, the environmental lighting significantly affects the speed at which the blob detection occurs. The opencv blob detector by default looks for black blobs and thus more light allows for easier detection by increasing the contrast between darker and lighter areas of the image.To facilitate the detection process, the images undergo a binary thresholding and the user can specify a region of interest for the blob detector to focus on. For eye tracking, it is necessary to have an infrared IR light source to increase the contrast between the pupil and the surrounding regions..

Limitations

Our eye tracking solution is not meant to solve all gaze tracking issues which may be more readily available in commercial solutions.

1. First, our eye tracker cannot be used if the head is freely moving. In our approach, which only detects the pupil, head motion is confounded with pupil motion. One future solution is to use both the corneal reflection and the pupil to allow for head-free eye tracking. This will be implemented in future versions of openEyeTrack.
2. Second, openEyeTrack does not output signals to analog channels which is a typical feature of commercial eye trackers. These analog signals were proxies for the analog signals from scleral search coils used for eye tracking.
3. Third, using openEyeTrack requires knowledge of Linux and some degree of comfort with the command line to compile and install various components and thus it is not as seamless and polished as commercial solutions. On the other hand it provides open source code for eye-tracking.

openEyeTrack is available on GitHub under https://github.com/mailchand/openEyeTrack and a more detailed description of usage can be found under the README.md file located in the repository. Currently, there are plans to incorporate *openEyeTrack* in research concerning the nueral dynamics of cognition, decision-making, and motor-control conducted at the Chand Lab at Boson University.

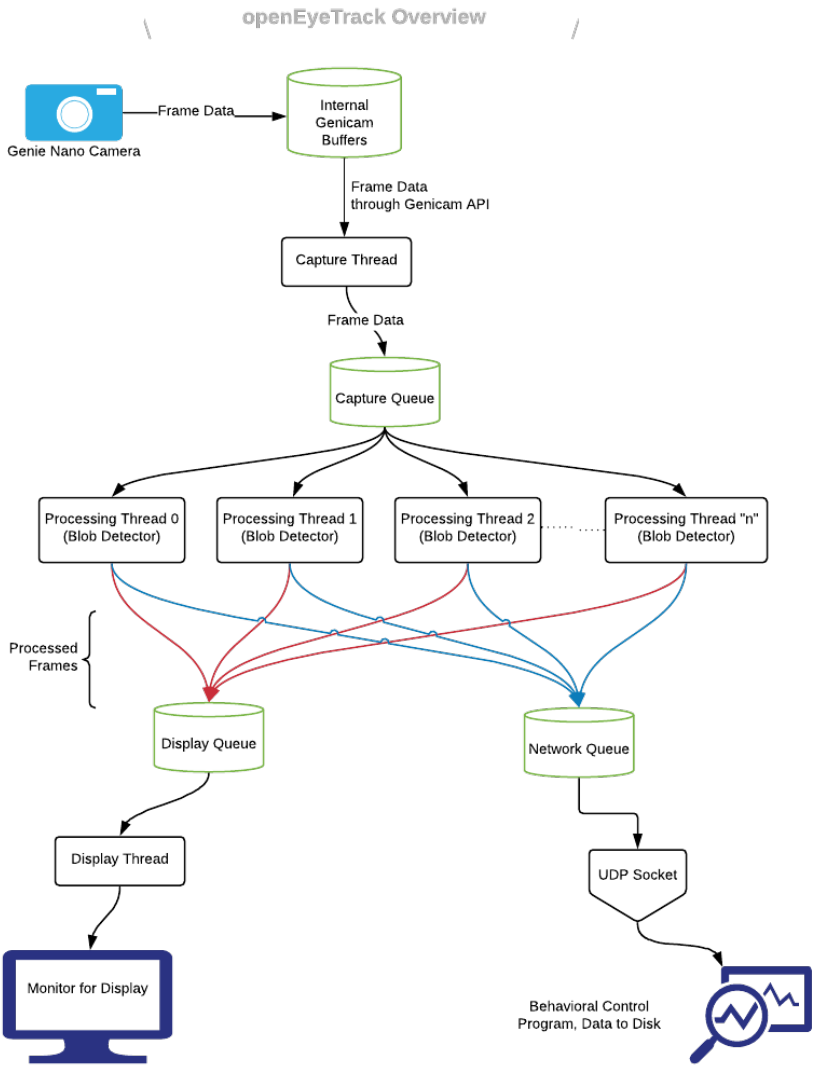

Figure 1: A visual depiction of the overall software and hardware architecture in openEyeTrack.

References

Bradski, G. 2000. *The OpenCV Library* (version 4.1.0). *Dr. Dobb's Journal of Software Tools.*

Casas, Jorge (Paolo), and Chandramouli Chandrasekaran. 2019. "OpenEyeTrack - a High Speed Multi-Threaded Eye Tracker for Head-Fixed Applications."

https://github.com/mailchand/openEyeTrack.

Kimmel, Daniel, Dagem Mammo, and William Newsome. 2012. "Tracking the Eye Non-Invasively: Simultaneous Comparison of the Scleral Search Coil and Optical Tracking Techniques in the Macaque Monkey." *Frontiers in Behavioral Neuroscience* 6: 49. doi:10.3389/fnbeh.2012.00049.

Teledyne DALSA. 2018. *Gige-V Framework for Linux* (version 2.10). https://www.teledynedalsa.com/en/support/downloads-center/software-development-kits/132/.

The Qt Company Ltd. 2013. *Qt* (version 5.5.1). https://www.qt.io/.

Zimmermann, Jan, Yuriria Vazquez, Paul W. Glimcher, Bijan Pesaran, and Kenway Louie. 2016. "Oculomatic: High Speed, Reliable, and Accurate Open-Source Eye Tracking for Humans and Non-Human Primates." *Journal of Neuroscience Methods* 270: 138–46. doi:https://doi.org/10.1016/j.jneumeth.2016.06.016.