The EyeSaver

Errol Caoleng, Emily Han Guang Liu, Matthew Larkins

UCSD IEEE QP WI21 Team 5

GitHub repository: https://github.com/KhanKhuu/EyeSaver-Prototype

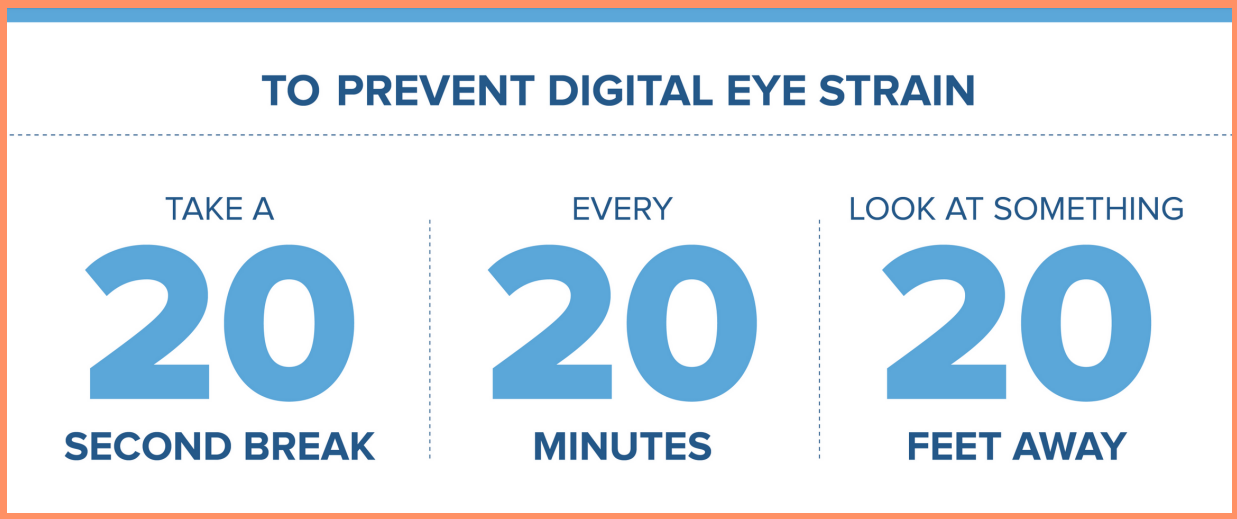
Introduction

RSI in the eyes is becoming increasingly common in today's digital age

Generally speaking, almost 90% of computer users experience visual and ocular discomforts including headaches, dry eyes, eye train, blurry vision, and more. These symptoms stem from extended technological interactions (i.e. computer, tablet, or phone), giving rise to digital eye strain—known formally as Computer vision syndrome (CVS)—affecting almost 60 million people today. Given that CVS arises from repeating a motion, it is classified as a form of Repetitive Strain Injury (RSI). With technology rapidly popularizing over the past few decades, public health experts are becoming increasingly concerned about CVS downstream effects. In 2013, Smita et.al found that about 62% of individuals who use computers for more than 6 hours per day have CVS symptoms: internal, external, and/or visual. Not only does CVS damage visual ability, it also contributes to a negative work environment by reducing productivity, accuracy, and job satisfaction. In 2018, optometrist Dr. Jeffrey Anshel proposed the 20/20/20 model: Every 20 minutes of screen time, take a 20 second break by looking at objects 20 feet away to prevent eye strain. Two years later, Waleed et.al designed and conducted a controlled educational interventional experiment to assess participants' ocular outcomes after learning about the 20/20/20 rule. Ultimately, based on questionnaire results, the researchers concluded that the intervention was a success. The reduction in eye dryness among patients instructed on the 20/20/20 rule compared to their non-educated counterparts was statistically significant.

Purpose & Motivation

Our project targets the external ocular symptom of eye dryness, which is a frequent issue associated with prolonged computer use common today. When focusing on the screen, users often exhibit reduced blinking rates, which can ultimately cause eye dryness and strain. We are interested in introducing the empirical evidence-based 20/20/20 model to all computer users through our program, which will remind them to take regular breaks from their screen. In accordance with the American Optometric Association suggestions to implement the 20/20/20 rule everyday to reduce CVS symptoms, we hope our program will be a convenient tool to use in improving one's ocular health. Given that the prevalent use of digital devices will only continue to rise across all aspects of life, it is increasingly imperative that users practice healthy screen time habits.



Timeline

Week 2/3 Developed concept of the EyeSaver program. Chose hardware we will use and split the project into two parts: 1) Get labelled gaze data and train a classification model on it to determine when user is looking at the screen or not, and 2) Implement a timing controller that alerts the user when to start and finish breaks based on the input from the gaze classifier.

Week 4 Developed pseudocode for timing controller and obtained a foundation of code that we used to get gaze data from the camera.

Week 6 Established a communications system (using an MQTT server/client setup) in python code to communicate between two programs. This is the protocol we used to send classifier predictions from the camera module to the timing controller.

Week 7 Implemented the pseudocode from week 4 into our timing controller prototype that uses MQTT to communicate with another program. Designed a test program to send simulated signals similar to what we expected to get from our classifier. Used the test program to debug and refine our controller to its near-finished state.

Week 8 Created programs to get labelled data from the camera module's gaze detector output. These are what the user will use to calibrate the camera to work with their office setup. Implemented a linear classifier (sklearn's SGDClassifier) and automated training it on the labelled data collected from the calibration programs. Saved the data processing pipeline and model parameters for use within the camera module's driver program in order to clean up the gaze detector outputs and make predictions on the real-time data. Simplified the timing controller program and connected it with the gaze classifier model output using the MQTT server. Optimized both programs for speed and functionality.

How-to-use

First, use the calibration programs in order to get data that the computer will work with when training its machine learning models. After executing both programs and following their directions (e.g. look at the screen for the calibrate_eyes_ON.py -c and look away for the calibrate_eyes_OFF.py -c), execute the training program (e.g. train_model.py) in order to train the machine learning model with the data received from the earlier programs.

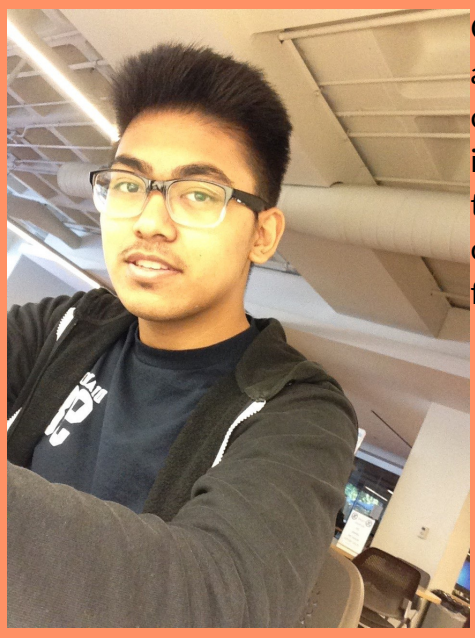
To ensure best performance, try to look at the screen (or away from the screen, depending on which data you are producing) from many different angles when calibrating, e.g. by slowly sweeping the eyes from side-to-side and top-to-bottom across the screen. It is possible to adjust the both calibration programs to change the amount of data the model has to work with; in general, the more data the model has to work with, the more accurate gaze classifications will be.

Then execute the alert program and eye monitor program (e.g. eye_strain_alerter.py and eye_monitor.py -c) for the computer to monitor your eye movements and keep track of whether you are looking at the screen or not. The program will provide you with reminders of when you have been looking at the screen for too long and need to take a break and when you can get back to work after you have rested your eyes. The eye monitor program can be adjusted so that you can change the amount of time you work and the amount of time you take a break for (e.g. by changing the variables called MAX_SCREEN_TIME and LOOK_AWAY_TIME, respectively).

A recommended balance is to look at something 20 feet away for 20 seconds after 20 minutes of looking at the screen. The values of the variables are the specified time in seconds.

Teammate contributions

Errol brought his electrical engineering expertise to lead the brainstorming of the initial pseudocode. This code laid the foundation for writing the code post-research and setting up of all hardware components.



Guang assisted Matthew with the brainstorming, testing, and debugging of the machine learning model and time controller programs to find ways of simplifying and improving the code to make it run more efficiently. He also took an important suggestion for decreasing runtime of certain parts of the camera module communication program from his roommate John.

Having zero software/hardware experience, Emily came in with a strong desire to see for herself the utility of having computer science and engineering skills. As a pre-med, she began by researching RSI, coming across a catchy 20/20/20 rule that laid the foundation for the project. Given she works at a business school, it is no surprise she helped coordinate workflow and piped up every time she had an idea for the code. Her requests help facilitate the user's experience to keep minimize stress and maximize productivity and well-being. In addition, she had a major role in the design of the documentation.

Matthew introduced the initial idea of an eye-strain preventer and had the OpenCV AIKit module to run the neural nets required efficiently. He also found an efficient gaze detection code for use with the module and implemented the gaze classifier model, from obtaining data to cleaning it up and training the linear model to applying the model on real-time data. Worked with Guang on the MQTT communications from the camera program to the timing controller.

Real-world applications

Getting into the habit of taking frequent breaks is important to preventing RSIs. However, it is easy to lose track of time and forget to take breaks, so the utility of this device is sending reminders to users. This device is mainly used for actively tracking whether or not the user is looking at the screen in order to remind them if they have been looking at the screen for too long. As long as the user follows through with the device's reminders, then our project will help mitigate RSI in the eyes. As a result, CVS symptoms such as eye twitching, soreness, burning, itching, and/or dryness will be prevented. Ultimately, mitigating eyestrain will prevent more severe outcomes like blurry vision, double vision, and increased sensitivity to light.

However, given some more additions to the code and additional computer learning models, it may very well be possible to adapt our project to also keep track of user posture as well to further promote better work habits.

Difficulties & Challenges

The first challenge that we encountered working on this project was that, while it was a mostly software-based project, all of the team members—except Matthew—lacked the experience with programming needed for this project, so most of us had to learn along the way. As a result, we needed to work on the project synchronously to learn how to do the project. Emily was the only one who had zero software/hardware experience, so she learned and made sense of the code from watching the others.

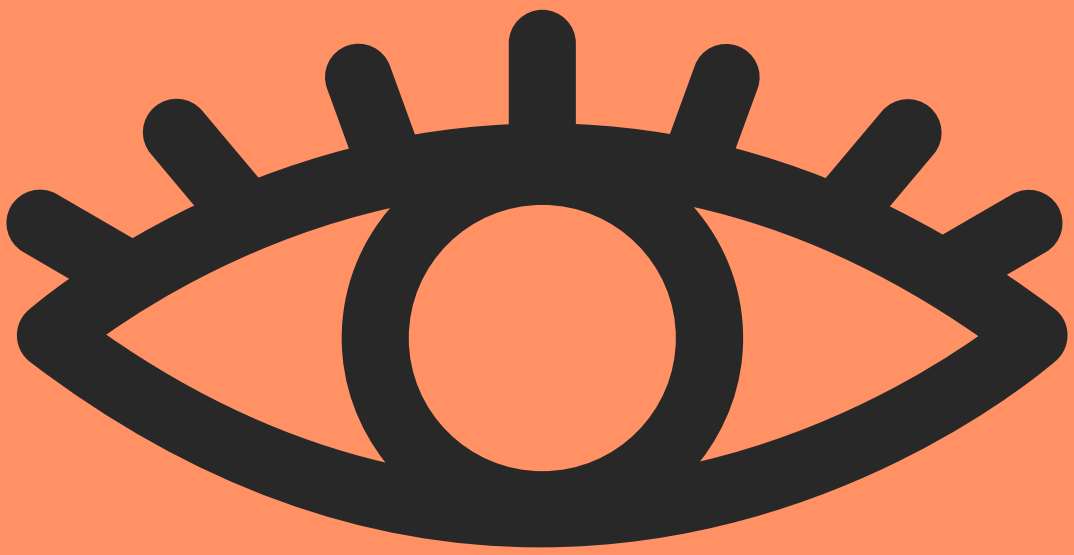
Another difficulty that we encountered was finding time and coordinating when to have meetings and work on the project, as everything was remote and we needed to find a time that worked for everybody to work together and learn from working on the project. We decided to work on individual components as our schedules allowed on all days except for Friday, during which most of the collaborative progress was made.

Improvements/ Enhancements

If we had two more weeks, we would most likely have made our project more user friendly, as in we may have made an interface for the user to interact with instead of a terminal command. For example, instead of a simple command remainder for the user to look away from the screen, we may have added a pop up remainder which would be more effective at reminding the user to look away from the screen.

Conclusions

Given how important computers are in our everyday lives, it is worthwhile to develop healthy user habits. Our device would be a valuable addition in developing ergonomic computer environments given it is convenient to set up and hassle-free once the user begins running it. We hope our device will not only facilitate productivity, but also well-being. Long-term eye damage—largely irreversible—is therefore effectively prevented by regular use of our device until natural habits develop.



References

Alghamdi, Waleed, & Saif H Alrasheed. "Impact of an educational intervention using the 20/20/20 rule on Computer Vision Syndrome." African Vision and Eye Health [Online], 79.1 (2020): 6 pages. Web. 14 Feb. 2021

Smita A, Goel D, Sharma A. Evaluation of the factors which contribute to the ocular complaints in computer workers. J Clin Diagn Res. 2013;7(2):331–335.