A Vision-Based System for Non-Intrusive Posture Correction Notifications

by

Nathan D. Beals

Undergraduate Honours Thesis
Faculty of Science (Computer Science)
Ontario Tech University
Oshawa, Ontario, Canada

Supervisor(s): Dr. Christopher Collins
Dr. Faisal Qureshi

Abstract

Sedentary lifestyle and the prolonged sitting associated with it may have negative health consequences and poor sitting posture only exacerbates that. In this thesis I propose a software application for monitoring a computer user's posture using a webcam, and notifying them in an unintrusive way when poor posture is detected.

Contents

Chapter 1

Introduction

1.1 Problem Statement and Motivation

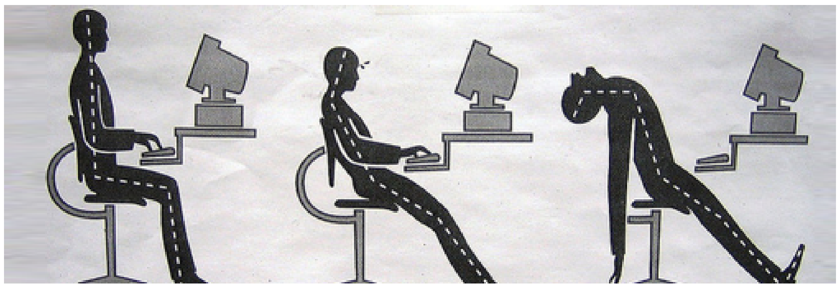


Figure 1.1: Illustrating good, poor, and very poor posture.

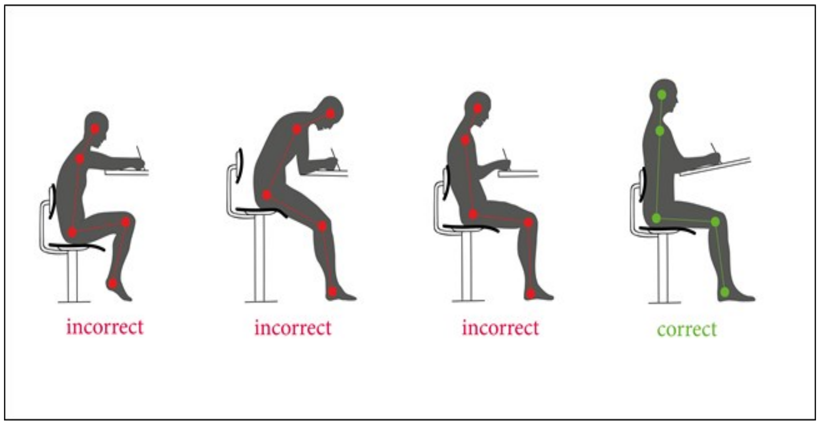
The prevalence of sedentary life has increased across my lifetime, and only continues to grow [10, 4]. Prolonged periods of being sedentary negatively affects health [2]. We have known for over 300 years that poor posture can cause disease in "chair-workers" [7]. To add to that, recent studies have shown that exercise does not entirely undo these negative effects [9]. Many people do not have a choice on whether they have to sit for their work; I am one of them so this project also fills a personal desire.

For the purposes of this thesis, I have defined good posture as a seated position with your back straight, with an angle between torso and legs within 90 to 110 degrees, a chair height such that feet are resting flat on the ground and the head is upright. [8]

The negative effects of poor posture can be somewhat mitigated by forcing myself to sit correctly as much as reasonably possible; good posture some of the time is better than never having good posture. In this research I propose a two-part system for detecting a computer users posture and for notifying the user of their poor posture so they may correct it.

1.2 Overview

(a) A typical desk set up. Could be a laptop. [13]



(b) Various types of correct and poor posture, not all are detected, more in Chapter 3.[11]

This project presents a system for monitoring posture with webcams available to consumers and then providing notifications that correct poor posture if it is detected. This is intended to help build good posture habits and mitigate the aforementioned negative health effects.

The posture monitoring framework is a Python-based OpenCV system for connecting to USB cameras. Using PoseNet [5], the positions of body parts and facial features (eyes, ears, etc.) are retrieved from the camera input. The important values are extracted from the PoseNet output and loaded into shared memory for the rest of the system to access. This allows for decoupling of the vision and PoseNet model from the rest of the system.

Poor posture notifications, determining when to send an intervention and building a model (calibration, long-term habits) of the user are all done in the notification system. PyQt5 [17] was used to create the user interface which enabled the notification elements to appear on top of currently focused applications while also not interfering with the users keyboard and mouse input to that application.

There are two pre-requisites for users; having a webcam and placing it more or less

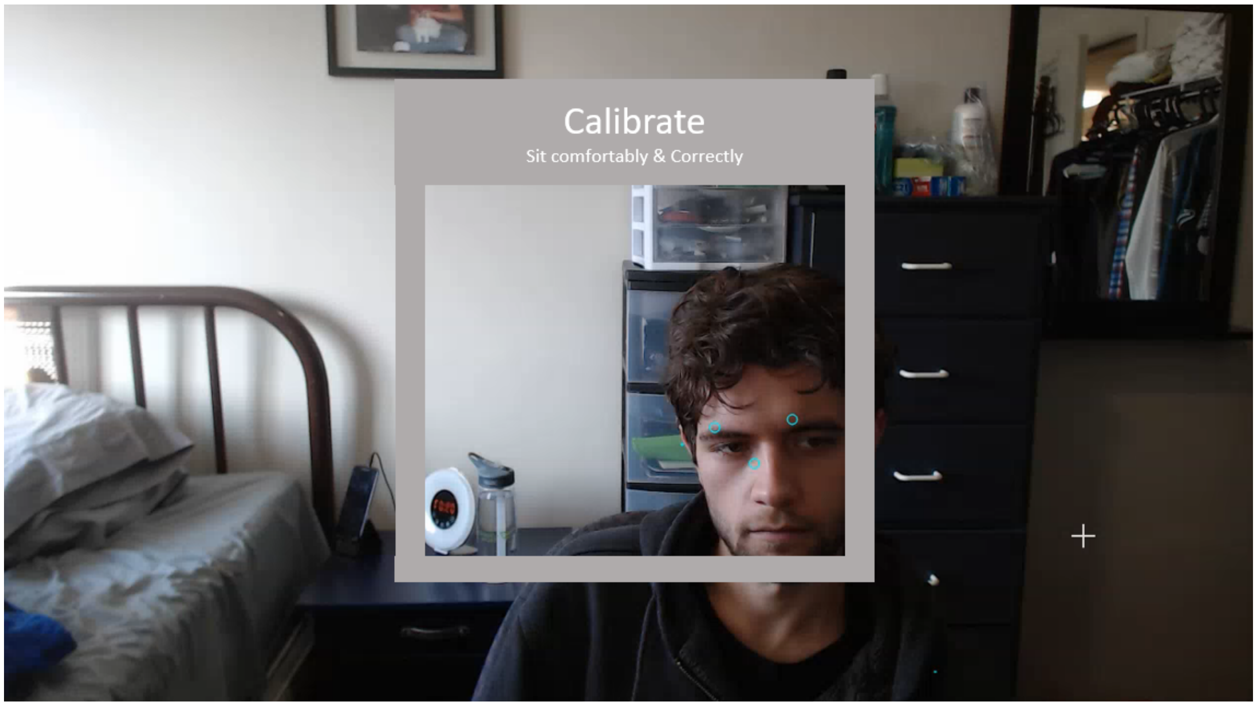


Figure 1.3: A bounding box the user resizes and moves over their face once after they are seating with good posture, calibrating where the user's face should be.

centered on top of their monitor. After that, users will start the application and be greeted with a 10 second calibration routine in which they will sit upright and with a correct posture, and then center the bounding box over their head. This is shown in Figure 1.3.

User interaction data is gathered as notifications are delivered to the user and they optionally provide feedback stating a positive or negative reaction to the notification. The intent is that over time the notification delivery system may better adapt to a users habits so as to not disrupt their work. Better posture some of the time is preferable to the software being disabled.

Chapter 2

Related Work

There has been extensive prior work in Human-Computer Interaction regarding sitting posture [4] and posture correction [1, 12] as well as in Computer Vision for detecting poses from images [5], [6]. However, there is little work focusing on correcting posture with a webcam without a need for special hardware.

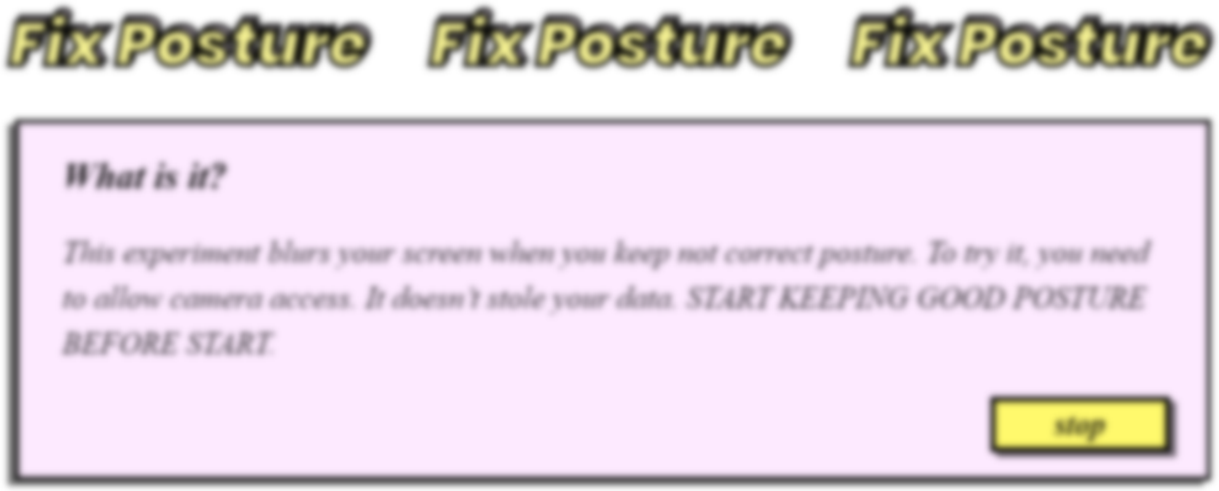
2.1 Workrave

Workrave [18] was the original inspiration for this project and while similar, the primary focus of Workrave is to get the user to take breaks from their computer. Using a combination of different configurable timers, the user can be prompted to take short, medium or long breaks. Also provided are illustrations of different stretches one could do while taking a break.

2.2 Hardware Devices

Various hardware solutions also attempt to solve the same problem. SpiMO [1] presents a pillow with embedded sensors that monitors posture and delivers feedback to the user through the hardware device. Lumo Lift [12] is a product that monitors posture through a bluetooth sensor attached to the users upper torso, posture feedback is delivered to the user through their phone.

2.3 Fix-posture.glitch.me



What is it?

This experiment blurs your screen when you keep not correct posture. To try it, you need to allow camera access. It doesn't stole your data. START KEEPING GOOD POSTURE BEFORE START.

Figure 2.1: Fix-posture blurring the screen.

Fix-posture [14] is a proof of concept showing that the concepts of the system I proposed is at least technically possible. Using PoseNet on the browser, the y position of the left eye is calibrated when the user presses start and the web page is blurred if the left eye moves too far up or down. Where I found Fix-posture to be lacking was; (1) it only runs in the web browswer, (2) the intervention scheme was obtrusive (it blurred your screen) and (3) the poor posture detection is too simple and error-prone.

Chapter 3

System

In this section, I will explain the technical and physical constraints this system must operate under. Then I explain my approach for identifying poor posture and for notifying the user. Finally, I discuss implementation details and visual design.

3.1 Constraints

The physical constraints are that the user is seated at a desk with a monitor in front of them that they must place a camera upon. The camera should be centered on the monitor. The technical constraints come from needing to gather posture data using 2D input from a consumer webcam, and the processing power needed to do so. User interaction constraints stem from needing to design non-disrupting notifications.

3.2 Design Approach

The system is split into three major components; (1) A posture detection system, (2) a notification system to alert the user of their poor posture, and (3) A process and shared memory manager to orchestrate communication between each component. This system is meant to run in the background while the user is doing other tasks and therefore needs as few resources as possible to not affect the user's work, which itself may be resource intensive.

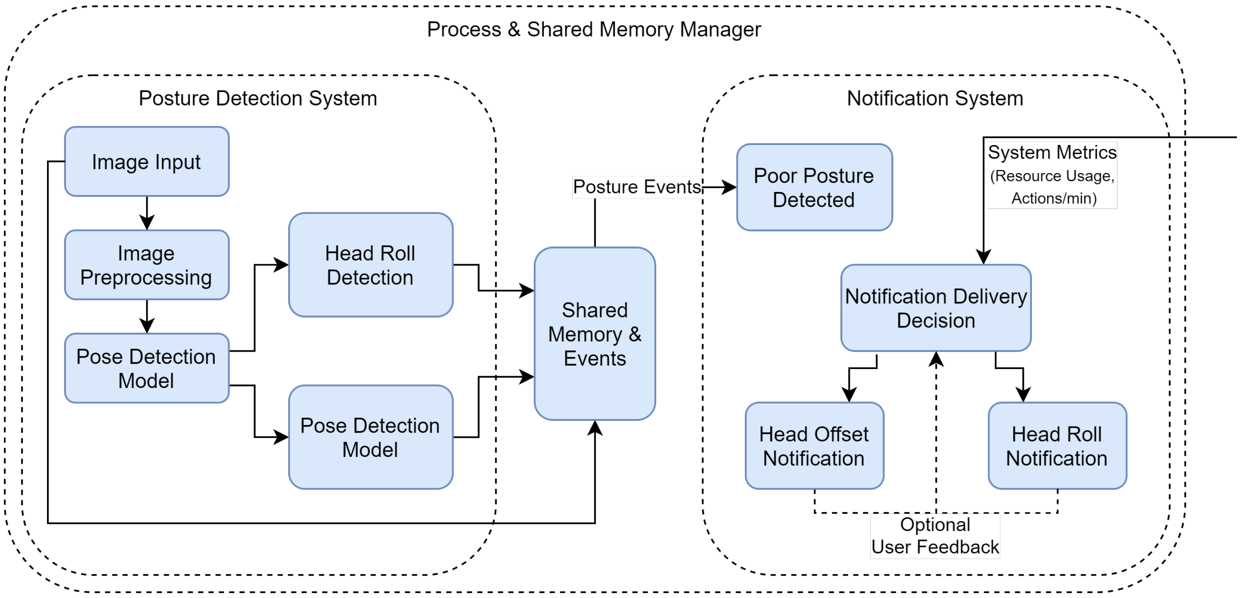


Figure 3.1: A diagram outlining the different systems and how they interact

3.2.1 Posture Detection

Detecting poor posture is done using only the feature positions from the user's head. From that, three classes of poor posture detection are possible. When the physical constraints mentioned earlier are met the system can determine the roll angle of their head. If the user's head is not upright, that is identified as poor posture. Detecting the other two classes requires calibration, which will be discussed below. After calibration, if the size of the user's face bounding box is different from the calibrated bounding box size then that means the user is sitting too far back or too close and identified as poor posture. Similarly, poor posture is identified if the user's eyes are too far outside of the calibrated bounding box, this posture means the user is leaning too much.

3.2.2 Notifications

What drove me away from using Workrave [18] was how intrusive the notifications were, some of them even blocking mouse and keyboard input until you dismissed the notification. Not wanting to repeat that, I took inspiration from video game [3] user interfaces. This means that the posture notifications discussed below are designed to contrast as little as possible, exist for as short as possible, not require interaction, and generally be unobtrusive.

Desiring consistency, I chose a single visual theme of a head outline (by using an oval) across all the notifications.

3.3 Implementation

Here I discuss implementation details of each system and the features they provide.

3.3.1 Process & Shared Memory Manager

Upon initialization of the application, a process and separate shared memory manager are created before anything else is loaded. The process manager provides a way to control process execution, including restarting if it crashes, process-level separation of tasks, and general purpose data pipes between child processes with callbacks. These features increase performance on multi-core systems, and allow for separation of concerns. The process manager is responsible for bootstrapping the other components by starting separate processes for each component with the required shared memory structures. The shared memory manager creates a block of shared memory that is shared by each component process.

3.3.2 Posture Detection

The first step toward notifying users of their poor posture is building a model of it to make decisions with. For this, Python, OpenCV, and PoseNet were used. After this component is started by the process manager, the user's desired webcam is opened and video frames begin streaming in. The full frame is mapped to shared memory. Captured frames internally use 3D Numpy [15] arrays (X, Y, RGB) which makes mapping it to and from shared memory straight forward.

The video frame is then passed along the image processing pipeline, which includes a greyscale conversion and resizing it to the desired input shape of the PoseNet model. The final image processing step is PoseNet itself, which attempts to run on a GPU but will fall back to a CPU if necessary. PoseNet does multi-person pose detection such that, for each

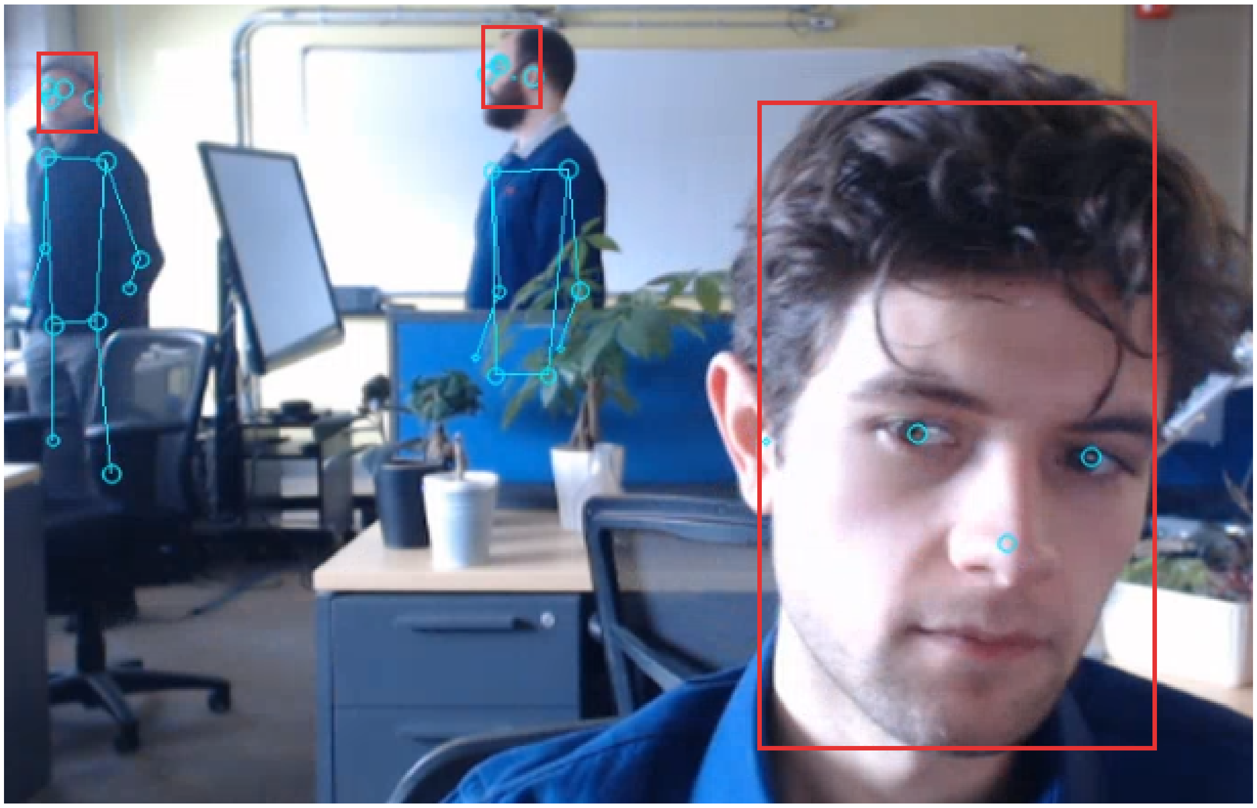


Figure 3.2: All detected bounding boxes shown by the red outline.

detected person, it provides 2D positions for each detected feature point, the confidence of that position, and the bounding box of that user's face. The application user is distinguished from any other results by selecting the result with the largest face bounding box, shown in Figure 3.2. That user's PoseNet outputs are then also loaded into shared memory and posture metrics are calculated, for example; Head roll angle is calculated using

$$\theta = \tan\left(\frac{RE_y - LE_y}{RE_x - LE_x}\right)$$

RE and LE are the 2D positions of the right and left eye.

Most of the aforementioned tasks are computationally intensive and because of the performance and usability constraints, the user can configure values that increase performance but in turn trade-off detection frequency and accuracy.

3.3.3 Notifications

Qt [16] and PyQt5 were used as the UI framework for implementing all of the visual components of the system. A native UI framework is required for performance and usability reasons and Qt already implements many needed features such as ignoring mouse and keyboard events on the window and window background transparency. QML was used because of the powerful and composable animation system combined with the many primitive shapes allowed for fast development while not sacrificing aesthetics.

Whether or not to act upon poor posture when the detection system finds it is controlled by the notification system. Just because poor posture has been identified does not necessarily mean a posture intervention should be delivered. User metrics such as how often this class of notification is delivered, the user's feedback of notifications, computer resource utilization, and more (fullscreen, actions-per-minute, ...) are used when deciding to deliver a notification.

When a notification is delivered, the user may optionally decide to give feedback. This is implemented with a simple "thumbs up" or "thumbs down" button that are found by right clicking the system tray icon. These responses represent whether or not the user wanted the previous notification to be delivered when it was.

User feedback, and other data such as how often notifications are delivered and how often poor posture is detected are gathered and used to build a model of the user's notification preferences.

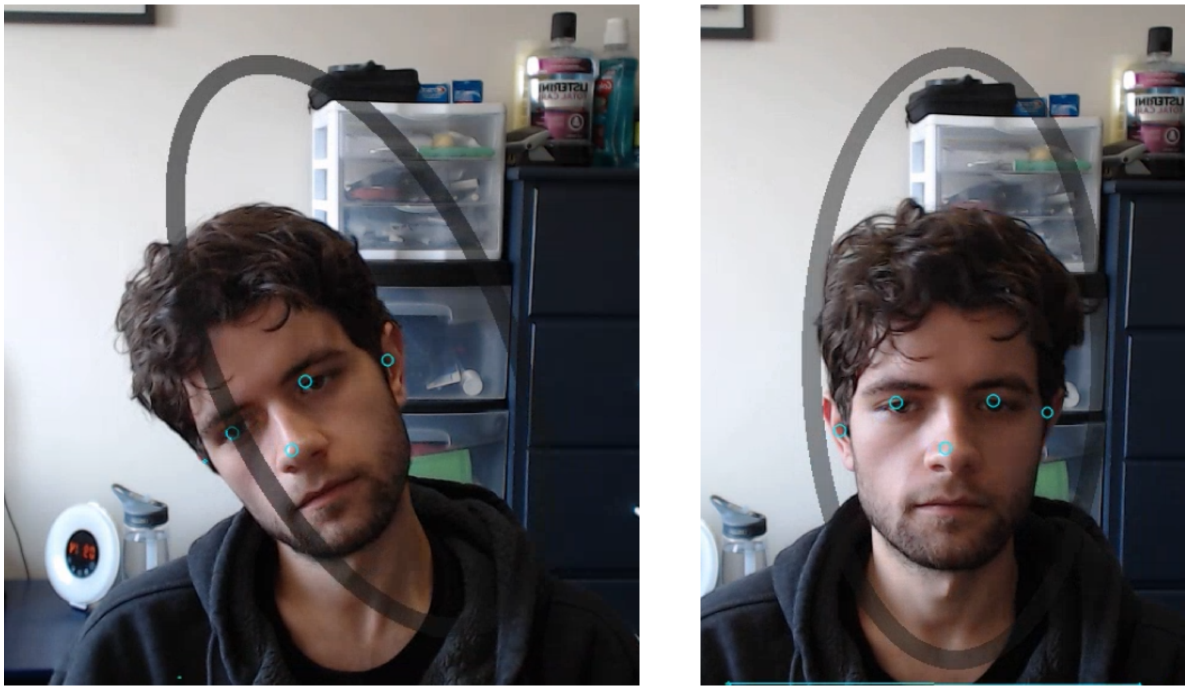
Calibration happens shortly after the GUI finishes initialization or when the user requests it. When the user is calibrating, they sit upright with a correct posture and they center and resize the calibration box over their head and then hold it for the calibration period. Internally, this reads from shared memory the bounding box of the user's face. Each frame the x, y, width and height of the bounding box are independently used to update an average value for x, y, width and height. These averages create the calibrated bounding box. While the bounding box is being calibrated, the angle of the user's head is also calibrated so that any minor perspective skewing caused by the position of the webcam may be mitigated.

After this calibration, the application begins working in the background, reading from the camera and monitoring the user's posture. Upon detection of poor posture, a notification, such as Figure 3.3 is delivered. The intent is the notification will remind the user to correct their posture without interfering with the user's work.

The notification shown by Figure 3.3 is intended to notify the user that their head is rolled too far, and to prompt them to tilt it upright. For demonstration it is shown over the camera feed but could be over any program. The notification is animated and begins with the standard head outline appearing centered on screen, rotated at the same angle as the user's head and then rotates to an upright position. The rotation duration, colour, and fade in & out duration are all user configurable.

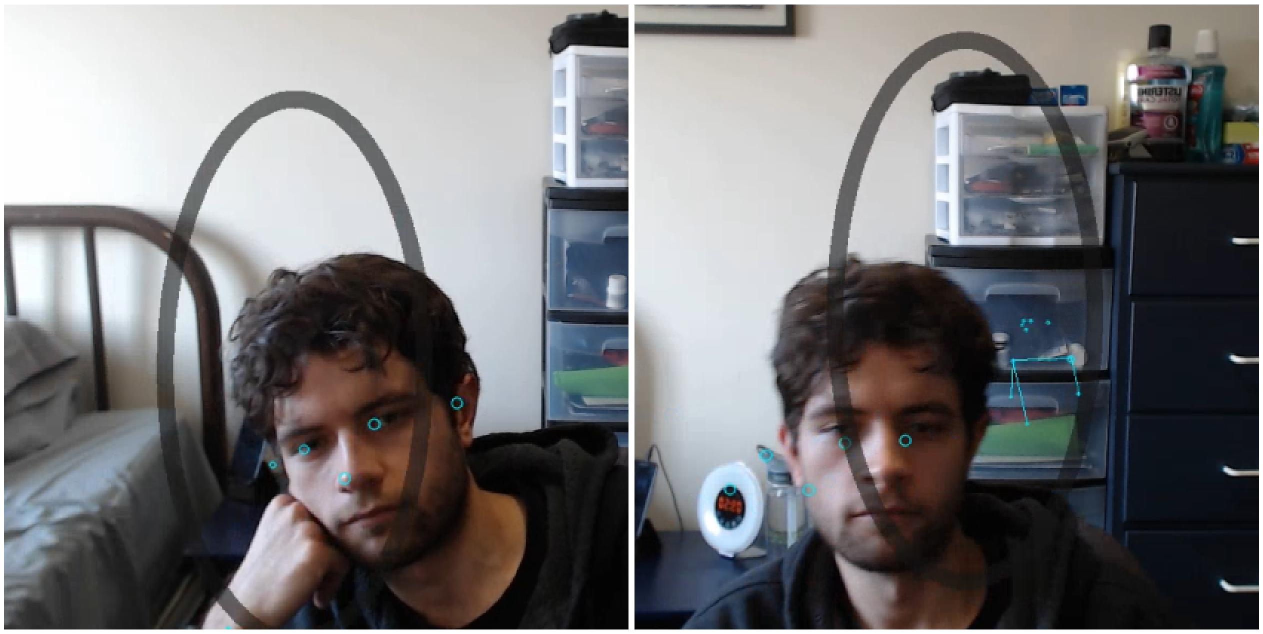
Similar to the head roll notification, the head offset notification uses the same head outline and is user-configurable. It notifies the user when their head is offset too far up, down, left or right or front-to-back. This notification has two separate animations, the front-to-back animation begins with the head outline fading in centered on the screen, but resized according to the ratio given by $\frac{A_f}{A_c}$ where A_f is the area of the currently detected face bounding box and A_c is the area of the calibrated bounding box. The outline then resizes until the size ratio above is 1. The other animation, shown in Figure 3.4, begins by fading in offset from center of the screen, and then moves back to the center of the screen.

Figure 3.3: A head roll notification being delivered, shown over top the user's camera feed.



(a) Beginning of the notification. (b) End of the notification.

Figure 3.4: A head offset notification being delivered, shown over top the user's camera feed.



(a) Beginning of the notification. (b) End of the notification.

Chapter 4

Conclusion

I presented a system that computer users may use to detect and monitor their posture, as well as visual feedback to help the user to correct their posture. The three-part system gathers data from affordable hardware that is accessible to consumers. The real-time posture data gathered is then used to notify the computer user of any poor posture they could correct by using a minimally intrusive visual on the users screen. This system attempts to build healthier sitting habits in users with gentle interventions in a way that minimally interferes with ongoing tasks.

4.1 Future Work

The proposed system can easily be extended; this work represents a solid framework for building a low-intrusion posture notification system. However there's still work to do; a user evaluation measuring how effective the posture notifications are would be first. Some others include extending the posture detection system by using newer pose detection models, with more pose data. More intervention notifications would naturally follow. Third is expanding the user metrics collection to incorporate computer usage-patterns. And finally a usability extension would be to add a camera multiplexer, by using the mapped video output from the shared memory block, the camera would remain open and usable for other software such as video conferencing.

Bibliography

[1] Jullia Birsan, Diana Stavarache, Maria-Iuliana Dascalu, and Alin Moldoveanu. SpiMO: Sitting Posture Monitoring System. In *RoCHI*, pages 143–146, 2017.

[2] Aviroop Biswas, Paul I. Oh, Guy E. Faulkner, Ravi R. Bajaj, Michael A. Silver, Marc S. Mitchell, and David A. Alter. Sedentary Time and its association With risk for Disease incidence, mortality, and hospitalization in adults: A systematic review and meta-analysis. *Annals of Internal Medicine*, 162(2):123–132, 01 2015.

[3] Brian Bowman, Niklas Elmqvist, and TJ Jankun-Kelly. Toward visualization for games: Theory, design space, and patterns. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1956–1968, 2012.

[4] David W. Dunstan, Bethany Howard, Genevieve N. Healy, and Neville Owen. Too much Sitting - A health hazard. *Diabetes Research and Clinical Practice*, 97(3):368–376, Sep 2012.

[5] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Convolutional Networks for Real-Time 6-DOF Camera Relocalization. *CoRR*, abs/1505.07427, 2015.

[6] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model. mar 2018.

[7] Bernardino Ramazzini. De morbis artificum diatriba [diseases of workers]. *American Journal of Public Health*, 91(9):1380–1382, 2001. PMID: 11527762.

[8] M Robbins, IP Johnson, and C Cunliffe. Encouraging good posture in school children using computers. *Clinical Chiropractic*, 12(1):35–44, 2009.

[9] Jannique G.Z. [van Uffelen], Jason Wong, Josephine Y. Chau, Hidde P. [van der Ploeg], Ingrid Riphagen, Nicholas D. Gilson, Nicola W. Burton, Genevieve N. Healy, Alicia A. Thorp, Bronwyn K. Clark, Paul A. Gardiner, David W. Dunstan, Adrian Bauman, Neville Owen, and Wendy J. Brown. Occupational sitting and health risks: A systematic review. *American Journal of Preventive Medicine*, 39(4):379 – 388, 2010.

[10] BBC: Sitting straight 'bad for backs', Nov 2006.

[11] Correct and Incorrect Posture. `https://www.howtogeek.com/349796/six-tips-to-help-save-yourself-from-poor-computer-posture/`.

[12] Lumo lift. `https://www.lumobodytech.com/lumo-lift/`.

[13] Seated computer usage. `https://www.seekpng.com/ipng/u2q8e6a9i1r5e6a9_active-sitting-and-good-postures-sitting-at-desk/`.

[14] Fix-Posture. `https://fix-posture.glitch.me/`, May 2019.

[15] Numpy: Numerical python. `https://numpy.org/`, Jan 2020.

[16] The qt company: Qt. `https://www.qt.io/`, Feb 2020.

[17] Riverbank computing: Pyqt5. `https://www.riverbankcomputing.com/software/pyqt/intro`, Mar 2020.

[18] Workrave. `https://workrave.org/`, Apr 2020.