Capstone Final Report

Hands Free E-Reading Apparatus

Group: L02

Names: Syed Naim, Chris Aiello, Gureet Grewal, Shaokai Yuan

Introduction: (Motivation & Background)

Many book readers spend time in awkward positions when reading. Such as; hunching over a book, or reading on their back, and even reading on the bed with a bent back. Positions such as these are damaging to your long term posture. For example, constantly leaning your head forward results in long term neck and back pain. Even reading on your back may result in tight shoulder muscles and a stiff neck, due to keeping your hands and neck steady.

Our motivation for this project was to create a sleek and modern electronic system, that utilizes hand gestures to interact with an electronic book reader or pdf file. The reason behind this system was to reduce the physical strain on the body due to poor reading posture over an extended time.

Project goals and planned approach:

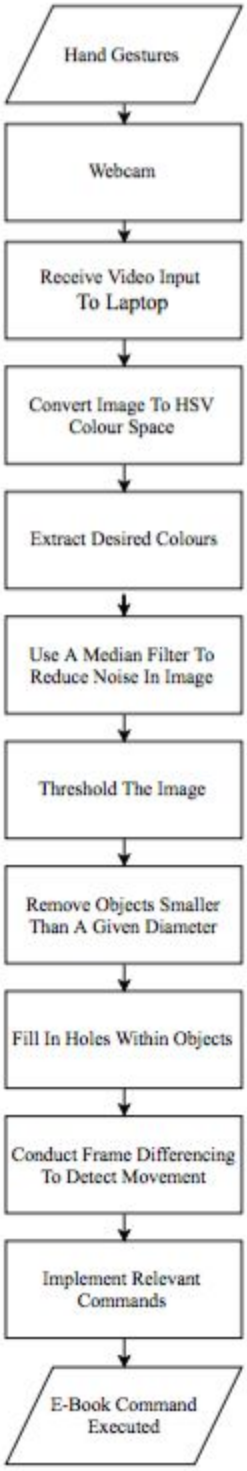
Our plan A approach was to use a USB webcam to capture video input and do image processing on it to detect movement. Once movement is detected the related command would be masked by the keyboard or other sources.

Our plan B approach was to use the Kinect system to achieve the same goal.

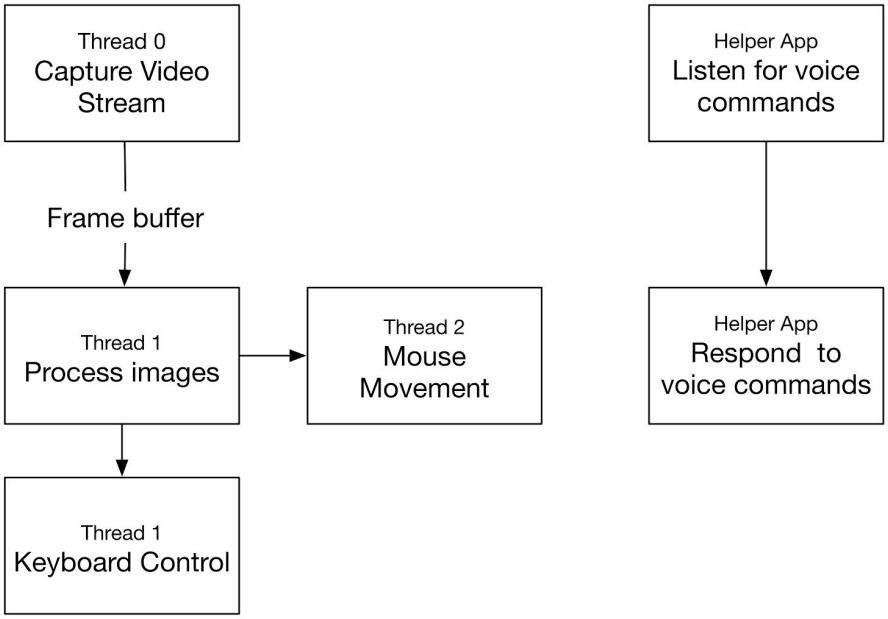
Actual Approach: (Methods/Theory/Implementation)

We implemented a variety of functions in MatLab/C++/Swift & OpenCV, including turning pages left and right, zooming in and out, and mouse cursor control with left click functionality. Voice commands for all of the gesture controls were implemented, along with a voice command to bookmark a page.

The following flowchart describes our actual approach used for the webcam based approach:



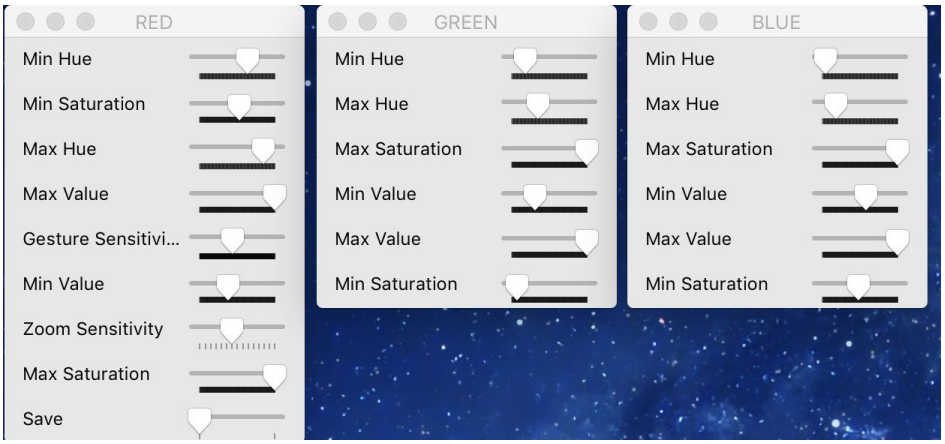
Low-Level Software Architecture (above)



High-Level Software Architecture (above)

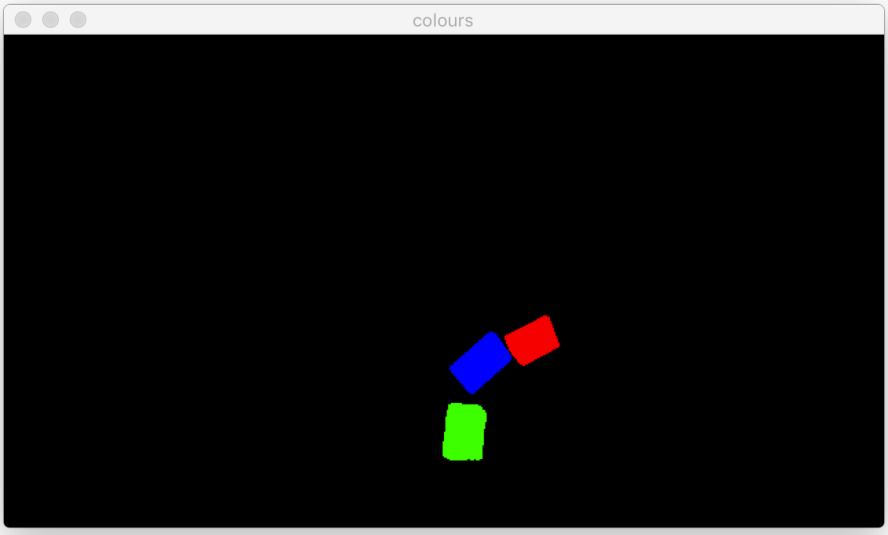
A multithreaded architecture was adopted to minimize bottlenecks in the image processing pipeline. Capturing frames from the camera and the processing of these frames occurred in separate threads so that the more resource-intensive task of processing the images would not interfere with the real time video acquisition. Control of the mouse was handled in a separate thread for the same reasons.

Colour detection was first implemented by manipulating the RGB pixel values of the captured images. After reading through the OpenCV documentation, we realized that our goal of isolating specific colours could be more easily achieved by converting the images to the HSV colour space. HSV uses one variable (**H**ue) to specify the colour, and two other variables to specify the colour's intensity (**S**aturation), and brightness (**V**alue). This allowed for more robust and intuitive adjustment to various lighting conditions, as usually only the **S** and **V** parameters need adjustment between environments. With these more precise parameters, the desired colours can be more easily extracted.



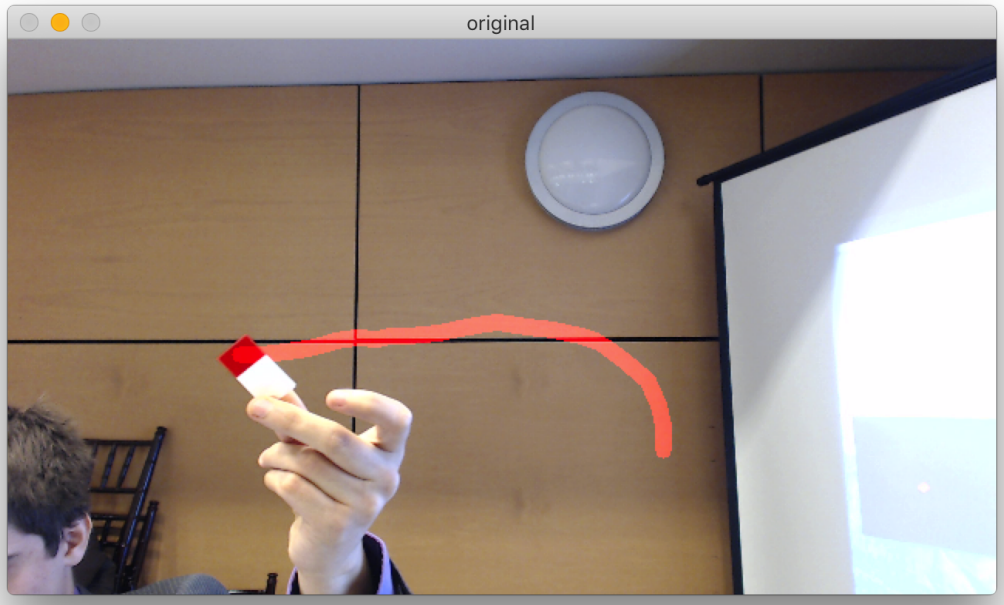
Variable Parameters

Once the desired colours had been extracted, the image was filtered and thresholded (converted to black and white). Next, small background objects were eliminated, and any visual holes in the primary item to be detected were filled in.



Combined detection of colours

Next, the x and y coordinates of the detected object are calculated and stored. By comparing x and y values between consecutive frames, motion can be detected.

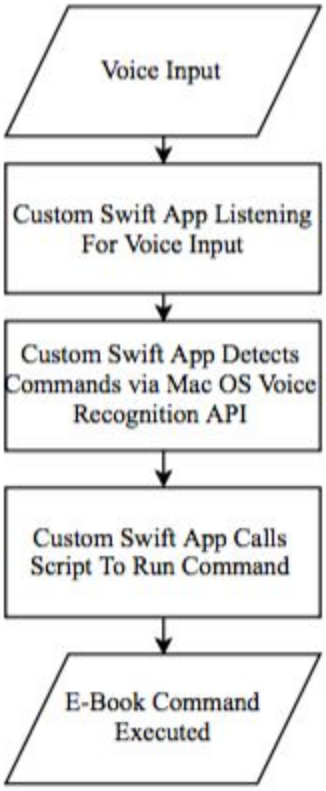


Red tracked across the screen

These steps were repeated for each of the three colours: red, green, and blue. Alternative colours can be selected arbitrarily if desired. The relevant gesture was selected based on what colours were detected. Only red indicated a page-turn. Red and green indicate a zoom. Only green indicates mouse movement. Green and blue indicates a left click. Actions are only triggered after the level of detected motion passes a predefined threshold.

Completing these gestures satisfied the basic requirements of our project. As an added feature, voice commands were added. A basic application was written which used the Mac OS X voice recognition API to listen for and respond to voice commands. This feature was a late-term addition and we did not perform the raw voice processing ourselves, as it was handled by the operating system.

The following flowchart describes our approach used for the voice recognition approach:



High-Level Software Architecture (for voice, above)

For plan B, we using C# to implemented our design. The working function includes page turning and zoom in/out through body language and several common commands through voice recognition such as open/close the file. For the hardware, instead of webcam, we using the Kinect sensor.

The following flow-chart describes our actual approach for the Kinect based method. Input: Hand gestures -> Kinect Sensor -> Receive data Input through depth camera and regular camera to laptop-> Convert data to 2D-body data-> using coordinate change of joints to implement different control function -> sending commands to the pdf reader through key-board masking function-> command executed

The Kinect has the depth camera which able to detect a body with up to 30 joints, by using coordinate of those data at frame rate of 30 FPS, we are able to create the algorithm to detect certain movement. The program is monitoring any motion that the target has, it only counts one direction moving and if the motion meets our duration and moving distance requirement, then based on our algorithm, we decide whether it is a legal action or not, if yes, the command executed and the same command will not be called again until the timer is passed or the body is back to its relative original position to ensure the unwanted movement are not detects as command. The working function includes two ways of page turning and zoom-in/out. By using the array microphone on

Kinect, we also implemented the voice recognition functions which is based on windows speak library. The working function includes open/close the book, zoom-in/out, maximize/restore the window. Overall, the implementation through Kinect sensor has lower accuracy and stability than web-cam based implementation.

Critical Problems Solved:

We explored an alternative using the Kinect however it required having windows OS, certain video cards, and USB 3.0 support. Since half of us owned a MacBook we decided to instead work primarily through a regular USB webcam. Some features were implemented using the Kinect, which was successful, however, we found more success with the webcam.

The initial challenge consisted of figuring how to do image processing on a set of captured frames through the video input. Since our group members lacked any experience in image processing this required a lot of research, and testing through trial and error.

The initial intention was to use gesture controls based solely on hand gestures, without colour recognition. Skin tones were difficult to isolate from the background, and parameters varied greatly between locations based on available lighting. The use of colours allows for more robust recognition, and more precise gestures.

We initially implemented the webcam theory in Matlab, we successfully created commands to turn pages, zoom, and control the mouse. However, the MATLAB's performance was relatively poor, so we reimplemented the process in C++ using the OpenCV (Computer Vision) library. After doing so we implemented voice commands using the built-in Mac OS voice recognition APIs.

Conclusion:

Our final result consisted of an electronic system that could receive either video input and recognize certain gestures to control an E-book. It could also receive voice input and implement the same gestures, along with commands for opening and closing the E-book reading software.

Future Plan:

We believe this, and other similar software has potential market value as a niche product for avid readers who wish to read in a more comfortable manner. Our software can also used as add-on for current e-book reader and it can be easily modified to support other software such as PowerPoint(slide presentation) and Netflix.

References:

MATLAB:
http://www.mathworks.com/company/newsletters/articles/tracking-objects-acquiring-and-analyzing-image-sequences-in-matlab.html
http://www.mathworks.com/help/vision/examples/tracking-based-on-color.html
C++/OpenCV:
http://docs.opencv.org/3.1.0/df/d9d/tutorial_py_colorspaces.html#gsc.tab=0
http://opencv-srf.blogspot.ca/2010/09/object-detection-using-color-seperation.html
OS X Speech Recognition:
https://gist.github.com/bellbind/c93ea23edc5ce5fde901