Worcester Polytechnic Institute
Digital WPI

Major Qualifying Projects (All Years) Major Qualifying Projects

March 2017

Face Analytics Web Platform

Gordon Gao
Worcester Polytechnic Institute

Julie Franca Valim
Worcester Polytechnic Institute

Rayan Abdullah Alsoby
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Repository Citation
Gao, G., Valim, J. F., & Alsoby, R. A. (2017). *Face Analytics Web Platform*. Retrieved from https://digitalcommons.wpi.edu/mqp-all/288

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.

Face Analytics Web Platform

A Major Qualifying Project Report:
submitted to the
faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degree of Bachelor of Science by

Rayan Alsoby
Gordon Gao
Julie Valim

Date: 03/14/2017

Major Advisors: Jacob Whitehill and Mark Claypool

Abstract

Automatic facial expression recognition technology can help researchers to conduct behavioral analyses and to improve the usability of a variety of software systems, especially in the domains of education and computer games. In order to obtain accurate results, it is important to conduct large-scale experiments over hundreds or even thousands of subjects; however, to-date there is no freely available platform to conduct experiments of this scale. This project focuses on creating a reaction recording platform. It will help facilitate creating large scale experiments where subjects from around the world can participate in the same experiment. A web application is created with HTML5 and JavaScript and is hosted on a Node.js server. The application enables subjects to record their facial reactions to a stimulus video and upon submission it is sent to a server which the researcher is able to access.

Table of Contents

1. Introduction

During the last several years, web video streaming has grown and become a standard method of communication as well as a tool for learning. Cisco predicts that by 2018, 69% of all Internet traffic will be video streaming [14]. People and businesses have begun to utilize online video streaming more because of the increase in quality and availability through smartphone and hardware advances. The use of online videos is being used in many fields such as marketing, video gaming, business, and education.

One of the groundbreaking new technologies that has arisen alongside web video streaming is facial analytics. Facial analytics software analyzes the appearance of the face pixels to estimate the movements of facial muscles to predict emotions. There are many facial expression analytic software packages available such as iMotions, Affdex and FaceReader; which they differ by analyzing different metrics [11]. The facial expression recognition field is growing due to the increasing interest in using it for various applications in marketing, focus group research, target demographic engagement testing, and educational research [35].

As facial analytic software has grown in availability and accuracy, researchers have begun to explore ways of measuring users' engagement from their facial expressions. For example, in 2014 Whitehill, Serpell, Lin, Foster, and Movellan explored ways to recognize engagement from students' facial expressions [37]. In 2015, Bosch et al., used facial expressions and body movements to detect students' facial responses as they played an educational game to learn the principles of physics [5]. These studies allow educators to identify when their students are struggling as well as allow educators to improve engagement in educational activities. The genuine and direct feedback collected is one of the reasons why it is desirable to measure people's emotions while they are engaged in some type of media.

One of the many areas that make use of the improved web video streaming technology is education. There are increasing numbers of Massive Open Online Courses (MOOCs). In 2011 the total number of MOOCs was three, but as of January of 2016 the total number of MOOCs was 4550 [33]. With the growing number of online courses, professors want to enhance the quality of their lessons to improve online learning. This could be done by using facial expression analysis to detect student’s engagement during their lectures. With the analysis, professors can refine their lessons to improve learning in their courses.

The rise of MOOCs, along with the growth in the facial expression recognition tools presents a need for more research and larger scale experiments to be conducted. These experiments can provide data which can improve the assessment of online courses. However there is difficulty in conducting these large-scale experiments that explore facial behavior in response to stimulus videos. The difficulty comes from the lack of intuitive software platforms that capture facial responses from a large pool of subjects.

The project requires creating a software platform for researchers to conduct large scale facial feedback experiments. Supposed experiments will consist of: showing a subject a stimulus video, recording the subject's facial reaction, and using expression recognition software to estimate the subject’s emotions while watching the video. An additional requirement is to have an intuitive interface used by the subject participating in this experiment.

The overall system to facilitate these experiments contains: user side web application, a web server, a facial expression recognition server, and a dashboard web application. The overall system helps with the large-scale experiments by allowing the collection of many facial response videos as well as an easy to use dashboard that helps with the visualization of the data for researchers.

Over the course of the project, a web application to record facial responses from subjects was implemented, as well as a server to store those responses and the associated metadata. Our application was developed using a video recording library and stores video information onto a web server.

The second chapter gives a more detailed look at our product. The third chapter provides background information on the tools and libraries used to create the product. The fourth chapter explains the process it took to create the product. The fifth chapter outlines the final product and its evaluation. The sixth chapter describes the future work that could be combined with the product. Lastly, the seventh chapter will conclude this project report

2. Product Description

This section provides an overview of our product deliverable. It discusses four different modules in our project design. Additionally, this section summarizes the requirements we focus on for the modules.

2.1. General Process

The platform consists of four main requirements:

1. Presents subject with stimulus video, and record subject's facial responses to stimulus video
2. Sends recorded video to web server
3. Runs facial expression recognition software on recorded video
4. Displays recorded video, stimulus video and expression recognition data on a dashboard

Each of the requirements above is implemented separately in its own module. Across the four modules different machines are used to achieve the overall product. Modules 1, 2, and 4 are hosted on a Linux server (Web Server). Module 3 is hosted on a Windows machine due to the licensing of the facial expression recognition software (Expression Recognition Machine). A diagram of the four modules is illustrated in Figure 1.

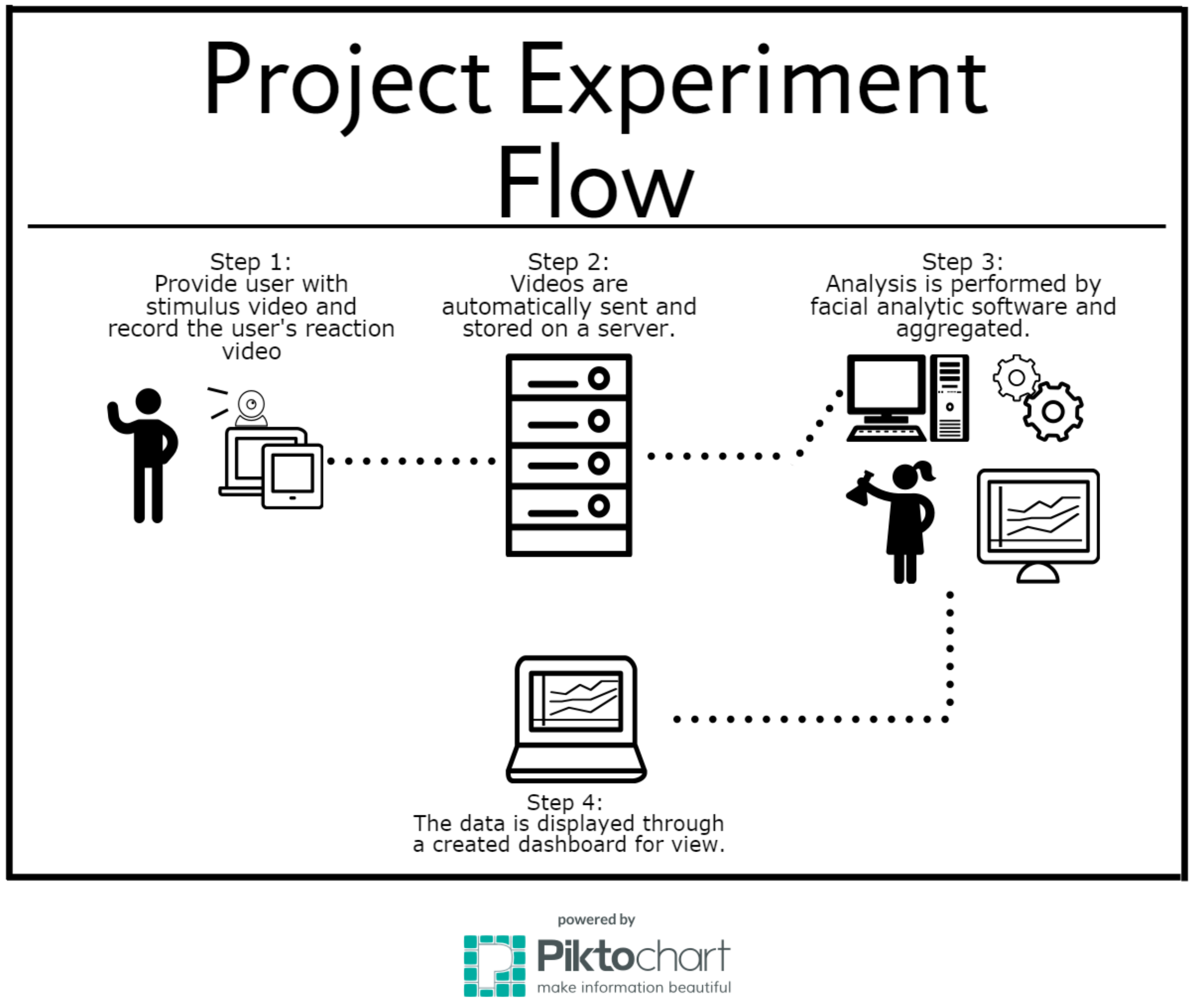

Project Experiment Flow
Step 1: Provide user with stimulus video and record the user's reaction video
Step 2: Videos are automatically sent and stored on a server.
Step 3: Analysis is performed by facial analytic software and aggregated.
Step 4: The data is displayed through a created dashboard for view.
powered by Piktochart make information beautiful

Figure 1. Diagram of the four steps that represent the experiment flow. Each of the steps is implemented by a corresponding module.

2.2. Product Modules

The first module is a web application that provides the ability to record a user's reaction in response to a stimulus video. The recorded video of the subject is the reaction video. The video presented to the subject through the web application is the stimulus video. To obtain subjects for the experiments, a crowdsourcing platform is used to connect subjects to this module. A mockup of the user web application can be seen in Figure 2. The user has access to video control options, including the ability to pause the video, move to any point in the video. The user can also submit his/her reaction recording through a submit button. While the subject is still watching the stimulus video, the subject's reaction is recorded and sent to the web server which is discussed in the second module.

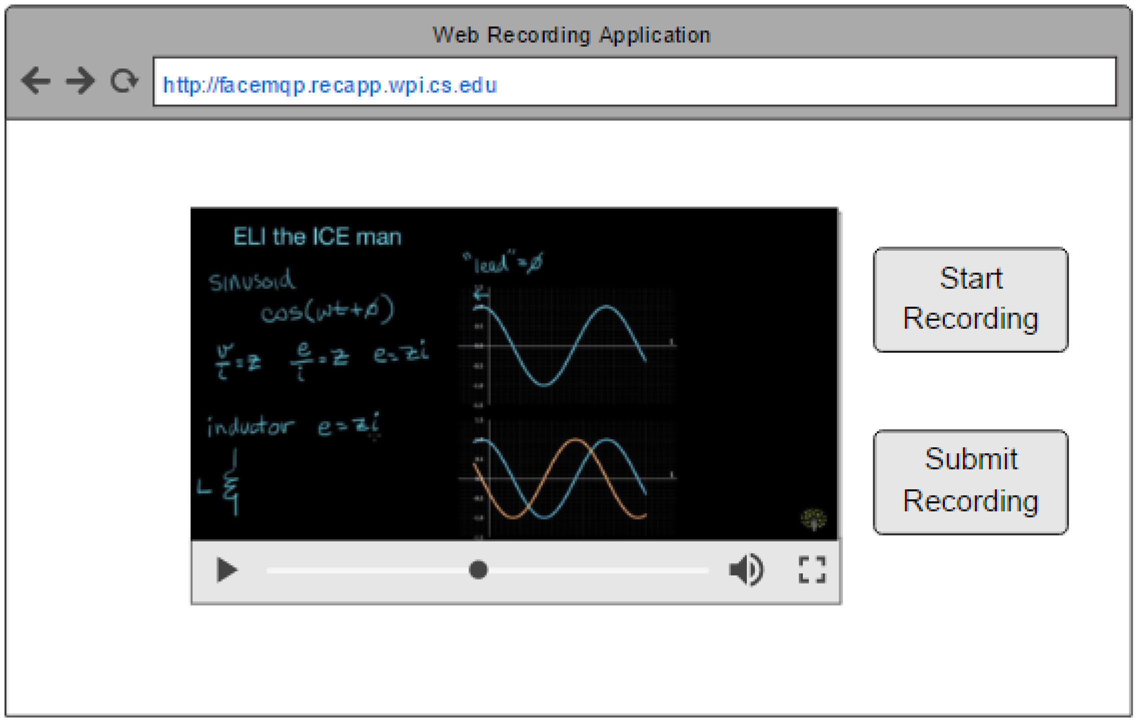


Figure 2. Proposed design layout for the web application. Picture displayed is of video provided by Khan Academy.

The second module is a server that hosts the web application. It stores a directory of recorded videos and user actions to the stimulus video. Then it transfers the recorded videos to the Expression Recognition Machine. This server also hosts the dashboard application described in module 4.

The third module is the process of using the Expression Recognition Machine to receive the recorded videos from the Web Server. Then the facial expression recognition software analyzes the recorded videos and outputs the analyzed data back to the Web Server.

The last module is the dashboard that the researcher uses to visualize the data. The dashboard provides the researcher with the stimulus video, the reaction video, and the graphs that represent that subject's emotions for that experiment. It also matches the reaction video to the corresponding parts in the stimulus video. A mockup of the dashboard can be seen in Figure 3. This figure shows only one view of this dashboard. The dashboard can also present aggregate data from the experiments. The aggregate data summarizes many subject's emotions from watching the same stimulus video. This provides the researcher with information that summarizes many subject's emotions and makes it easier to come to a conclusion about the results.

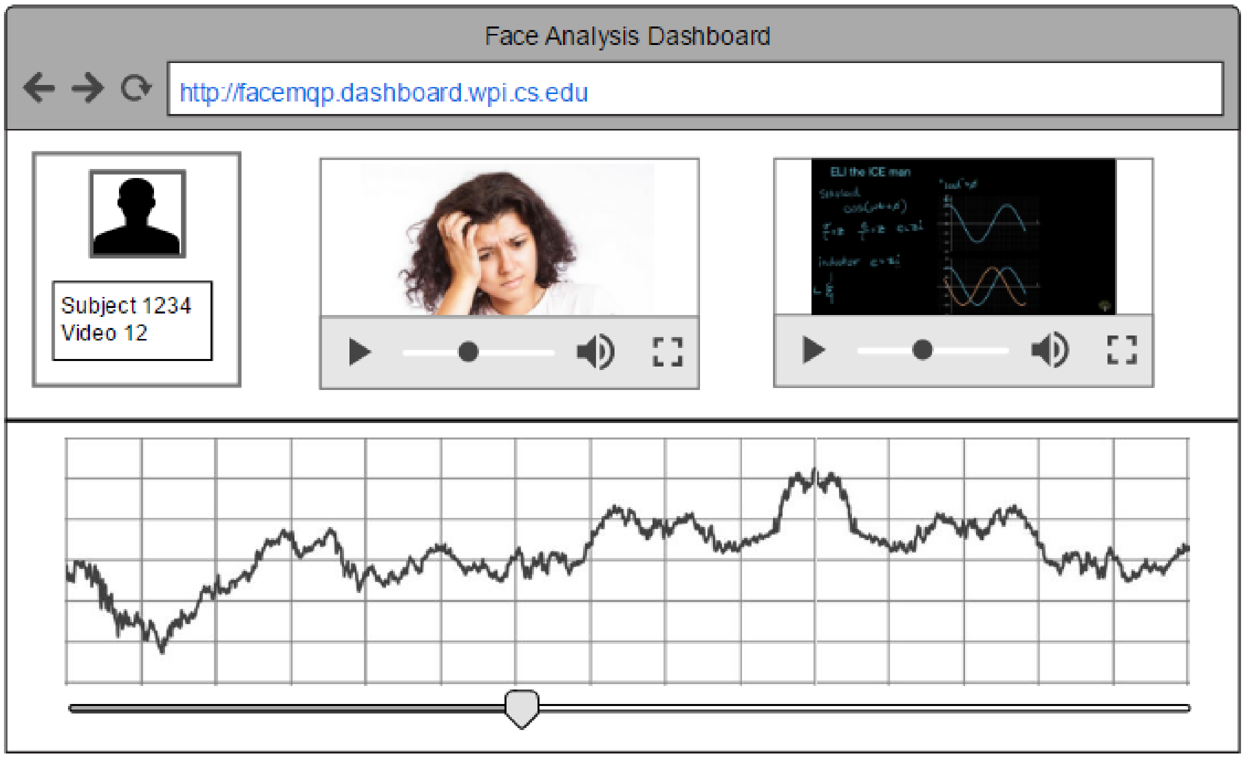


Figure 3. A proposed layout for the dashboard web application.

2.3. Project Specific Requirements

From the modules mentioned, the development process for this project focused on the first and second modules. The requirements for this project were made by separating the platform into several goals. The project's main goal, the creation of a customizable platform for the capture and transmission of video information of sufficient quality to be used by facial expression recognition software was the main focus when creating these requirements.

The videos recorded need to be of a certain quality and format for it to be accepted by the facial expression recognition software. iMotions requires the face of the person in the recorded video to be at least 64 pixels wide. Another requirement for the recorded video is frames per second. A 30 fps (frames-per-second) video provides enough frames for iMotions to run through and analyze and produce meaningful results. The recorded video must be in mp4 or wmv format for iMotions to process it. The videos also need to be in a reasonable size (70-75 KB/second of recorded video) to reduce the bandwidth and accommodate for users with slow upload speed.

Timestamps need to be recorded for user interaction with the stimulus video. This includes recording the clock time as well as the video time in which a user performs some kind of action to the stimulus video such as pausing or fast forwarding. Capturing the timestamps is needed to provide the ability to playback the recorded video and stimulus video in synchrony on the dashboard. Also, the timestamps provide the necessary information to synchronize the facial expression analysis with both the recording and stimulus video on the dashboard.

The recorded videos and their corresponding timestamp files need to be stored in the server in a proper directory system. The directory system consists of holding each recorded video and their timestamp files in their own folder. The directory allows the researcher to keep the recording session information organized. Thus the system is comprised of folders that each relate to an experiment.

Another requirement of the project is to make the video recording module be a web application to make it easy to setup and run by the researcher. By making the module a web application that only requires an initial setup on a server, it reduces the setup time for future experiments. Experiments can be done by accessing the web application instead of downloading and installing a program. An added benefit of requiring the video recording application be a web application was to allow the user of a crowdsourcing Internet marketplace like Amazon Mechanical Turk to interface with it.

These are the major requirements that we followed during the development of this project. These requirements serve as guidelines for the design of the first two modules to be effective components of the overall product.

3. Background

This section contains the background information that was collected as this project progressed. This will include existing software packages that relate to our project as well as information on the technologies that were used, such as API's and libraries.

3.1. Related Work

This section describes currently available platforms that are similar to the scope of our project. These platforms provide facial recognition software capabilities as well as dashboard-like systems that provide the user with the analysis of the recorded videos.

One of the first steps to solve the problem is to understand existing technologies that perform similar tasks to the software platform that is being developed. An example of an existing technology, is Affectiva's Affdex [9]. This platform records facial responses, runs the expression recognition software on the recordings, and then displays the analyzed data in graphs. Affdex enables the researcher to see the analysis of individual responses as well as aggregate emotional engagement. The Affdex application uses a standard webcam combined with their computer vision algorithms to identify key landmarks on the face. Then the machine learning algorithms analyze subtle movements in those landmarks to classify facial expressions and complex emotional responses. The combination of these facial expressions is then mapped to emotions. This application is used by different developers to enable their own applications with emotion adapting features.

Another existing product is FaceReader by Noldus which has very similar functionality to Affdex. FaceReader analyzes six basic facial expressions as well as two neutral type expressions [10]. It also includes other calculations such as gaze direction and head orientation. Noldus uses what they call the Project Analysis Module for the analysis of the facial expressions to gain insight into the effects of various stimuli.

Affectiva’s Affdex and Noldus’ FaceReader differ when it comes to the metrics that are analyzed. Affdex analyzes seven basic emotions in terms of valence, engagement, and attention [11]. In contrast FaceReader analyzes six basic emotions in terms of "contempt (experimental), arousal, and valence" [11]. Affdex uses fourteen facial expression metrics that are similar to the twenty action units that FaceReader utilizes. Affdex and FaceReader perform many of the tasks that our platform requires; however, for this project FaceReader is not open source and lacks customizability. Affdex is open source, but it does not transfer information across a server which is an important component in our project for conducting large scale experiments. So there is a need to create a new distributed system platform that serves the purpose of this project while also being customizable.

3.2. Technologies

The technologies section contains more details on the software and tools that we use throughout this project. In order to develop our own independent system we combine open source products and libraries related to video streaming to a web server.

3.2.1. Recording & Playing APIs

To be able to record and play videos in our application, different API’s to record and play videos had to be investigated.

3.2.1.1. Playing Videos

The YouTube API provides data when the user controls the progress of the video they are watching [39]. There are multiple callbacks built into the API that can record timestamp information based on video controlling action. These timestamps are essential when synchronizing the played video with the recorded video [26].

3.2.1.2. Recording Videos

This project refers to Muaz Khan's MediaStreamRecorder library [19]. An advantage of this library is that it has mobile support. One disadvantage to using MediaRecorder API is the lack of support on some browsers. Currently it supports Chrome version 49 and higher, and Firefox version 25 and higher. Although the lack of support for some browsers is unfortunate, the two browsers Chrome and Firefox alone are used by over 60% of desktop users [1] [2].

3.2.2. Media Encoding

The video conversion software Ffmpeg is used to merge videos together on the server and convert them from webm to mp4. Ffmpeg is a widely used open source cross platform tool used to merge and encode videos. It has a command line tool that can be implemented on the web server [13].

3.2.3.Web Servers

Another component to our project lies in the web server that is hosting the web application and storing all of the recorded videos. The web server in our platform hosts the web application and host all recorded videos. During the research portion of this project, the three main web server frameworks Flask, Node.js and Apache were considered for further use.

3.2.3.1. Flask

The Python web server framework Flask is known to have an easy learning curve. It is easy to start up and maintain as well as being lightweight. Flask is a micro framework mainly used for smaller applications with less requirements than of this particular project [31]. Specifically when it comes to recording and playing videos. Flask is the youngest of the python web server frameworks hence there is a lack of extensive libraries and online documentation.

3.2.3.2. Apache

Apache foundation offers two services for web servers; Apache and Tomcat. The two both have extensive library for web services [3]. The Apache framework does not have easy access to dynamic web service capabilities without an additional tool like Apache Tomcat. The Apache server standalone is more effective responding to static web page requests than dynamic web page requests. Tomcat can handle dynamic requests and is easy to set up and run, but it would be more effective to use Apache for static sites. In order to develop for both static and dynamic web page capabilities well, Apache and Tomcat should be combined together. Since both are required to make full use of the Apache web libraries there is a high learning cost in order to set up and develop for both.

3.2.3.3. Node.js

Node.js is a JavaScript runtime environment that is simple to setup and like Apache, it has an active development community. It is considered to be part of the MEAN stack, which is a JavaScript framework that includes MongoDB, Express, AngularJS, and Node.js [23]. This framework accelerates the development of web applications by utilizing the libraries and tools that are available from the development community. Node Package Manager (NPM) is an online repository that has many open source published projects which can be easily incorporated in other projects. Node boasts key features that distinguishes it from its competitors. Having both client and server implemented in JavaScript is helpful because JavaScript proficiency is already required for the development of the client side. Therefore, the learning curve for using Node.js is less steep [29].

3.2.4. Crowdsourcing Technologies

Crowdsourcing is the act of having a large group of people contribute to a project. Crowdsourcing is often associated with an online community of people working together on a project. In order to gather participants for our MQP effectively, an Amazon Web Service crowdsourcing technology known as Mechanical Turk is used for this project. Mechanical Turk is often used when human processing power is required. Using Mechanical Turk, Amazon members can post Human Intelligence Tasks (HIT) for completion with a requested number of participants and description of the task desired for completion [38]. Once posted, other Amazon users can perform the task and usually receive a small monetary payment as a result. These task are usually simple tasks that cannot be performed by a computer easily. Some examples are editing papers, associating websites or objects with adjectives and pronouns, or identifying an image. The payout is often in increments of cents. For the purpose of this MQP, subjects are obtained through Mechanical Turk and asked to record themselves watching an educational video.

3.2.5. Facial Recognition Software

There are many facial recognition software packages currently available for use. This project will utilize the facial analytic software iMotions [12]. iMotions has many integrated technologies that help analyze subject data, however this project will mainly use their facial expression analysis module. It allows for the use of a regular web camera to record the subject's facial expressions in a non-intrusive way [12]. The facial recognition software iMotions requires either MP4 or WMV files, as input. The software then extracts all the facial expression data from the video, analyzes and aggregates it, and exports raw data to then be used with any statistical program to be further analyzed.

4. Methodology

The section goes into detail on the process by which the product was developed. Throughout the development process there was constant feedback from our advisors based on their specifications. The project started with the user interface first and then functionality was added incrementally to ensure that there was always a working version.

4.1. HTML5 vs Flash

The first crucial component of this project is to enable the subject to watch a stimulus video while recording their facial responses in synchrony. Prior to choosing which APIs are used over the course of the project, a clear decision needed to be made regarding which web development platform would be used. The two main options are HTML5 and Flash. Flash media player has considerably more documentation online; however HTML5 is the standard and latest version used by web applications [15].

Since the project requires a reliable media transmission to catch as wide an audience as possible, HTML5 was chosen for this project. The biggest advantage when using HTML5 is that it is native to browsers and no plugins are necessary for the web application to run. Flash requires the user to install a flash plugin if it is not already installed. As well as Flash lacking mobile support, having a web application that does not require installing additional software for the user is important to this project because it is more inviting for users [28]. Thus HTML5 was chosen as the platform for development of this project.

4.2. Media Recorder

MediaRecorder is the main API for video recording over HTML5, hence it was chosen to be used for this project [24]. This project refers to Muaz Khan's MediaStreamRecorder library in particular because it has many similarities with this project's scope [19].

4.3. Choosing a Web Server

Node.js is used as a web server instead of Flask or Tomcat to host the recording application and recorded videos. During the server consideration process, the most important quality was ease of use. A server that is quick to set up and run is required due to the time constraints and lack of experience when working with web servers.

Even though the Apache servers have the same capabilities to receive and store videos as Node.js it also requires more knowledge to incorporate all of the libraries to develop an application that is highly customizable and well performing. An effective product requires using both Apache HTTP and Tomcat together. Flask is known to be simple to use and simple to setup, however it is a micro framework mainly made for smaller applications. The functionality that is needed for this project; such as transmitting videos, is not readily available.

Also due to this framework being the youngest of the python web servers there is not much documentation for the functionality widely available. Node.js is a better solution for this project. It is developed in Javascript for both client and server side so there is a less steep learning curve. There are also extensive libraries and open source packages readily available for the functionality that is needed for the video recording and transmitting application.

4.4. Recording Interface Development Process

In order to display a video, a YouTube link was first embedded onto the webpage. However, just embedding the video did not provide enough functionality, so a video controller API was required to provide more functionality such as recording state changes and other controls. This project uses the YouTube video API in order to display the stimulus video. To display the video in an HTML page, the YouTube video player is loaded into the page and the embedded link to the stimulus video is then inserted as the source for the YouTube video player. When creating the player object there are multiple options that can be set such as height and width of the player allowing for customization [39].

After embedding the video player, the video events needed to be captured. In order to capture video events, an event handler is initialized to the YouTube player. Events are actions that are invoked when interacting with the YouTube player. Event handlers are needed to invoke actions based on changes in the player. Two events that are handled for are onReady and onStateChange. The onStateChange event helps with invoking actions when the state of the player changes - e.g. playing, and paused. The onStateChange event is used to generate the timestamps. It provides a way to know when the state of the video is changed and what state the player is currently in [39]. To enable the syncing process, a set of timestamps from both videos is used for matching. Timestamps are a text recording of the actions performed by the user on the YouTube video player followed by the current time in the recording and YouTube video.

4.5. Video Player Events & Timestamps

After successfully displaying the stimulus video, the next step was implementing the logic to receive the actions performed by the user while they are watching the stimulus video. As mentioned earlier, the YouTube video player provides event handler functionality. In this project, a function is called by the event handler that then produces a collection of timestamps on the client side and sends them to the server whenever the state of the video changes. In order to transmit to the server, the Javascript library was used. Using Socket.io provided us with the capability to send files to the Node.js server. In order to provide functionality for a dashboard and to aid in future synchronization problems, a list of timestamps is sent to the server. The list provides a simple to access collection of data on the actions the user performs. The timestamps text file will be used by the dashboard as it will help in synching the stimulus video with the recorded video [39].

4.6. Recording Videos

The next main addition to our process was video recording. To record a video on a webpage using HTML5, MediaRecorder API and Muaz Kahn's MediaStreamRecorder library are used [24][19]. The recording function in MediaStreamRecorder includes options such as the resolution and the length of chunks to customize the recording process. Currently the captured video resolution is set to 640x480 because it is a resolution that provides decent quality while maintaining a small size. With this resolution one second of video requires approximately 75 KB. Having the user wait while the recorded video is uploading is undesirable especially with long videos.

To avoid that, the recorded video is be split into chunks. Each of these chunks are sent to the server as soon as they are available. The chunks are sent continuously to the server while the recording is happening and the user is watching the stimulus video. This process decreases the time the user has to wait after completing an experiment, since the uploading is happening during the experiment.

4.6.1. Trials

Two factors were involved in choosing the chunk length: the size of each chunk and the number of frames lost between chunks. When the recording web application was tested, a 3 second chunk resulted in missing frames of the recorded video if the total video length was around 10 minutes or higher. For example, our tests consisted of 10 minutes of recorded video. This 10 minutes would become 9 minutes and 53 seconds when the chunk sizes were 3 seconds.

To assess this flaw three different machines were used with different specs to do multiple trials of 10 minute recordings (specs of these machines can be seen in appendix A). In these trials, multiple chunk lengths (3, 10, 20, and 40 seconds) were tested on the different machines. A phone stopwatch was used to count the elapsed time, and the watch was recorded by the recording web application. Then the time difference was recorded between the video time and the stopwatch time in a spreadsheet. After completing the different trials, the results showed a correlation between the chunk size and the missing frames from the recorded video. The results can be seen in figures 4, 5, 6, and 7.

The figures below show that as the length of the recorded video increase, the difference between the video time and the stopwatch time increases. The x-axis lists the time in seconds of the trial. The y-axis represents the time difference in seconds between the recording video length and the actual time the recording took. This was rounded to the nearest whole second. Each point in the graph shows the difference between the actual length and recorded length at that time on the x-axis.

After testing the different chunk lengths on different machines the trials had to be done on a single controlled platform. Therefore, there were multiple trials of the same chunk lengths on one machine. The results of these trials confirm our assumption that shorter chunks lead to missing frames. The results of the last trial can be seen in figure 8. After reviewing the data, 40 second chunk sizes was decided for use in the project to reduce the amount of data lost.

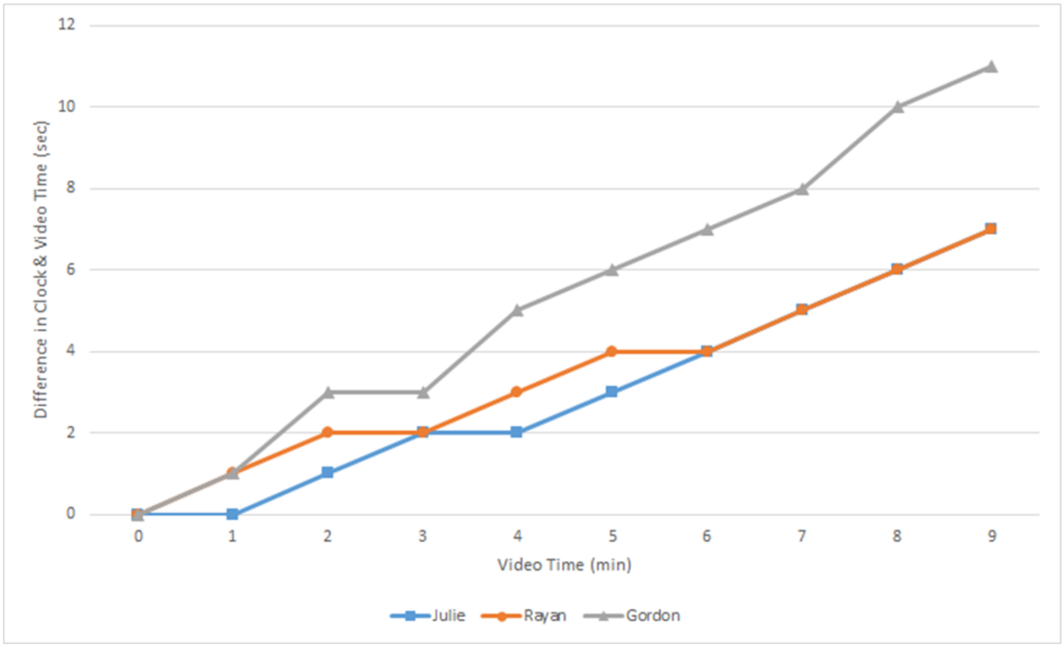


Figure 4. Graph for testing the missing frames problem with sending 3 second chunks across all machines

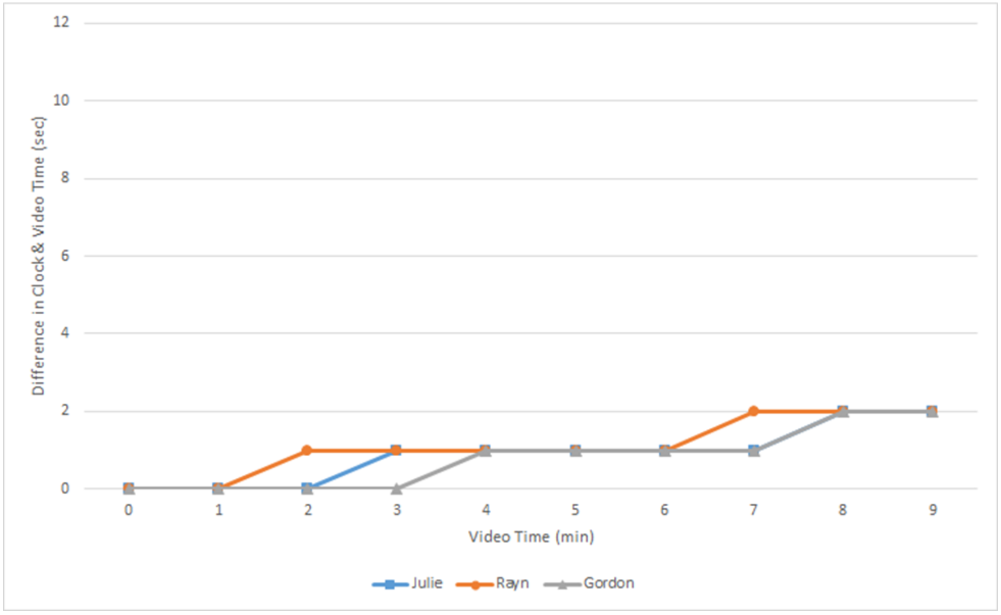


Figure 5. Graph for testing the missing frames problem with sending 10 second chunks across all machines

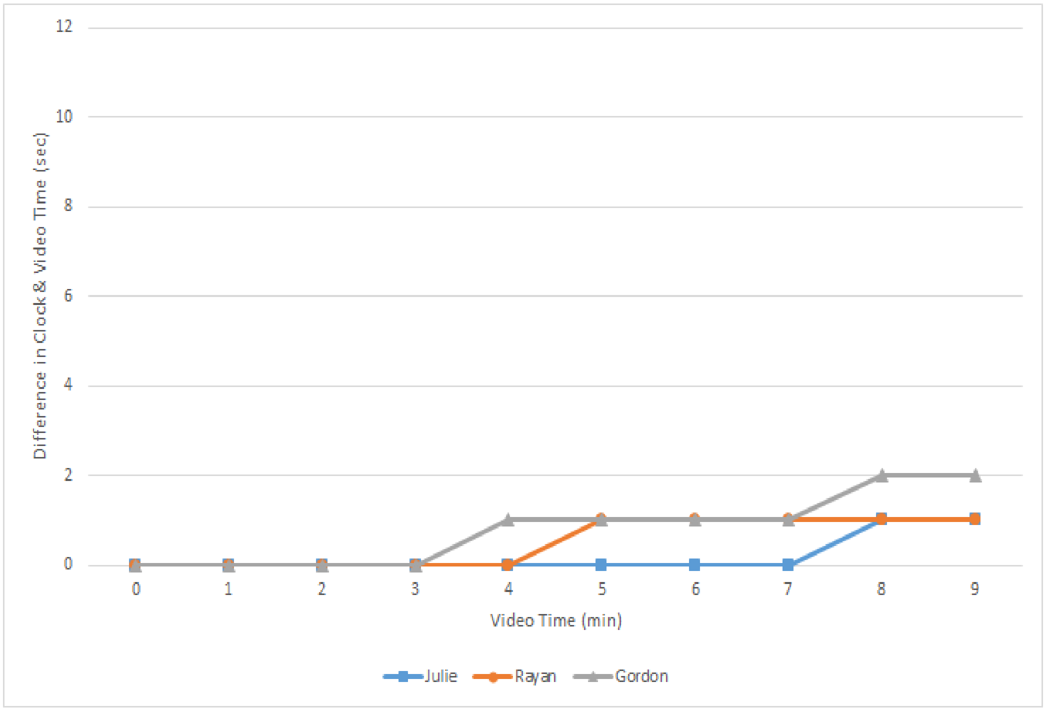


Figure 6. Graph for testing the missing frames problem with sending 20 second chunks across all machines

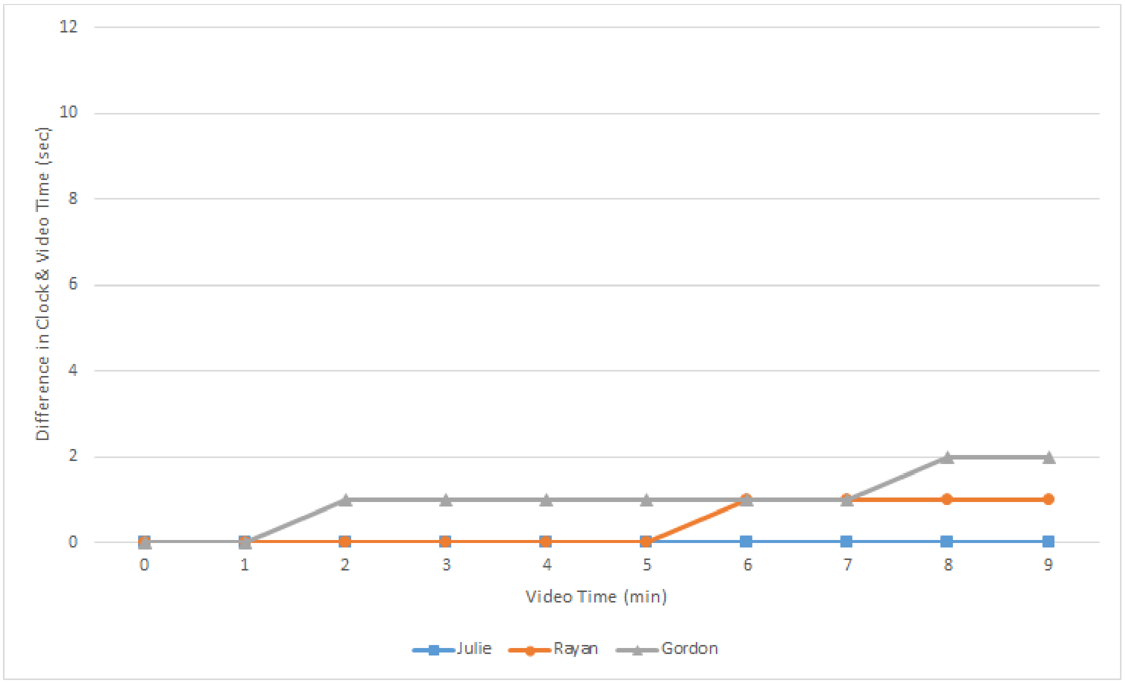


Figure 7. Graph for testing the missing frames problem with sending 40 second chunks across all machines

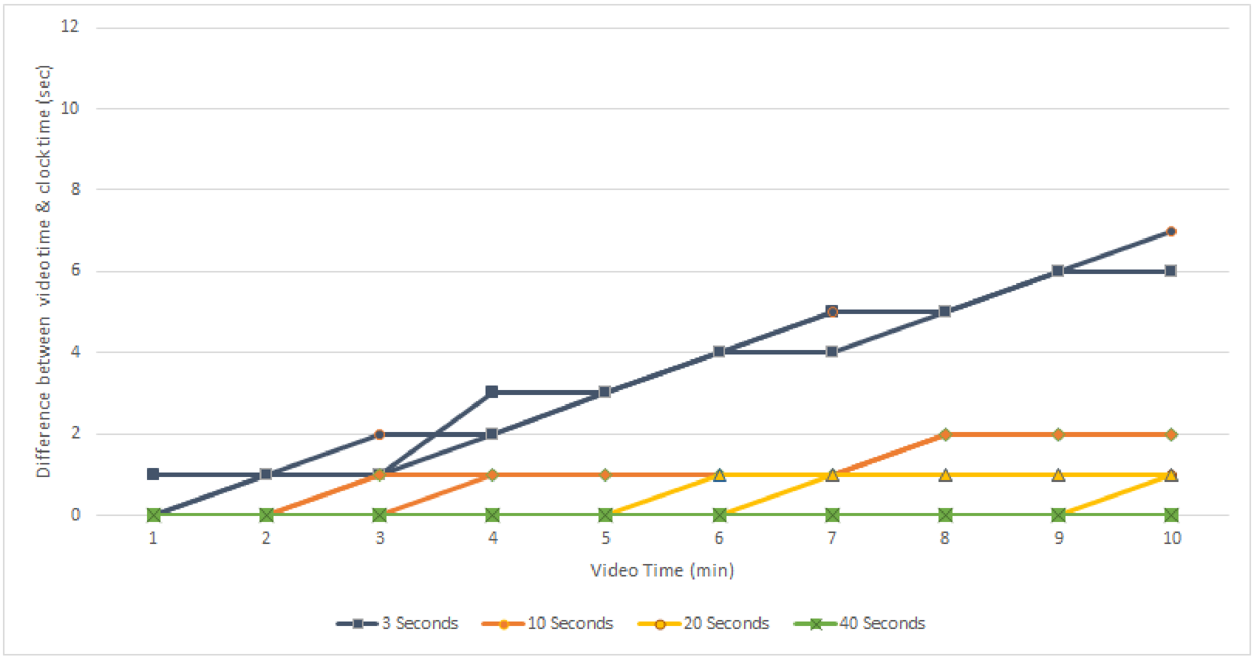


Figure 8. Graph for testing the missing frames problem with sending all different chunk sizes only on Rayan's machine

4.7. Create a Web Sever & Upload Recorded Videos

4.7.1. Web Server Configuration

Since one of the product requirements was to develop a web application that is compatible with Mechanical Turk, the web page needed to configure to the mTurk interface. In order to have our recording application displayed on Mechanical Turk it needs to be HTTPS format. Transitioning from a hyper-text transfer protocol (HTTP) website to a hyper-text transfer protocol secure (HTTPS) website lets users know that any information being transferred to and through the website is protected [22]. Data is encrypted through the use of a SSL certificate, which is an object that contains public and private keys that can encrypt data preventing anybody else from accessing the data [36]. Public and private keys are used to encode the data and then decode them once they have reached the desired user. After successfully using Node.js to load a web page, the SSL certificate was provided by WPI. The SSL certificate proves authenticity and allows our website to be HTTPS.

4.7.2. Web Server Directory

The web server contains a directory to store all of the recorded video files as well as the corresponding timestamp files. The directory is structured in a way that for every assignment or hit from mTurk it has its own folder in the directory. This makes it so that each experiment has its own folder location.

4.7.3. Merge Video Chunks on Server

Once the chunks have been sent to the server, the backend development phase was started. The chunks arrive in a webm format and a tool is needed to convert it to mp4 automatically. In order to merge the chunks on the server the Ffmpeg and fluent-ffmpeg libraries are used [13]. The ffmpeg library is needed to provide video conversion capabilities. Fluent-ffmpeg is a JavaScript library that provides the ffmpeg commands to be run through the Node.js server. Using this library helps keep our system consolidated and prevents the need to run external scripts for ffmpeg commands.

The merging and encoding in ffmpeg happens in a single command. Ffmpeg is provided with a list of webm chunks as input, and give the following settings for the encoding of the resulting video: encode it with codec 'h264', format it as 'mp4', and set the frame rate to be 30 fps. After all the chunks have merged successfully they are deleted to save space.

4.8. Design Layout for Mechanical Turk

Once the functionality was completed, work was done on improving the user interface. This included deciding where exactly on the interface the stimulus video would be displayed, as well as the recording video and all of the buttons that control the actions. It is important that the user interface is clean, simple and intuitive for the user. The editing of the interface was done through html. The videos were first centered and placed prominently. Thus it was decided to have as little on the UI as possible. A screen capture of the web application is shown in figure 9.

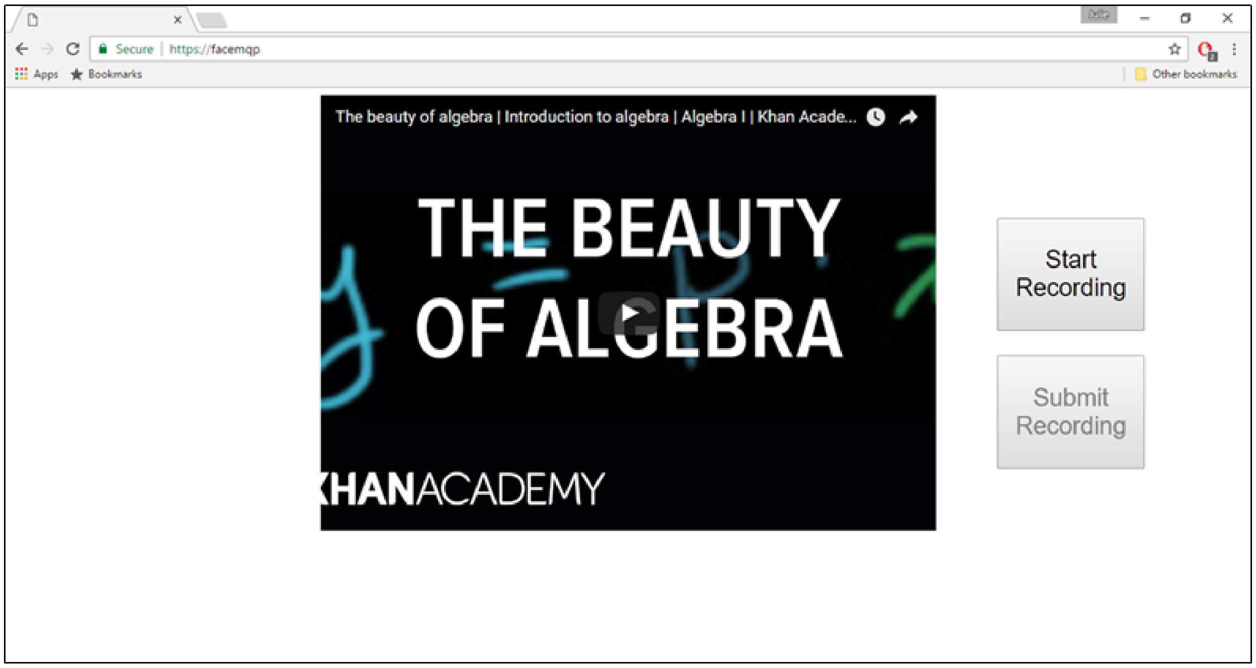


Figure 9. Screen capture of the user interface for the web application

The main focus is the stimulus video, thus it is placed in the center of the interface. To the right of the stimulus video are two buttons to start and submit the recording. They are large and very distinguishable to the user to keep the layout simple.

4.9. Connect and Test Web Application with Amazon mTurk

After creating a working version of our web application, work began on assimilating the application into a HIT to be used on mTurk. Our development took place in the Mechanical Turk sandbox development environment. A sandbox environment is a development tool that provides the same functionality as the program it is based upon, but acts solely as a test environment. The tasks performed in a sandbox environment are not part of the final product. HITs can be posted and performed without cost in the mTurk sandbox environment, which was helpful during the testing process. Experimentation with mTurk consisted of posting a working HIT that refers the user to the web application, successfully having a user perform the HIT and having their recorded video transmitted to the server with timestamps, and finally having the user receive “payment” in the sandbox mode.

Each Amazon mTurk subject is given a unique assignment ID number that correlates the user with the HIT he/she is working on. Using the ID number as an identifier the recorded data is transmitted to the server and stored in a created folder based on the ID. Using this method of storage, each successfully completed experiment is stored in its own folder.

Having the web application connected to mTurk also created new requirements for the interface. Since the subject needs to accept the HIT in mTurk before starting the experiment, a thumbnail of the video is placed in the location of the stimulus video. This can be seen in Figure 10.

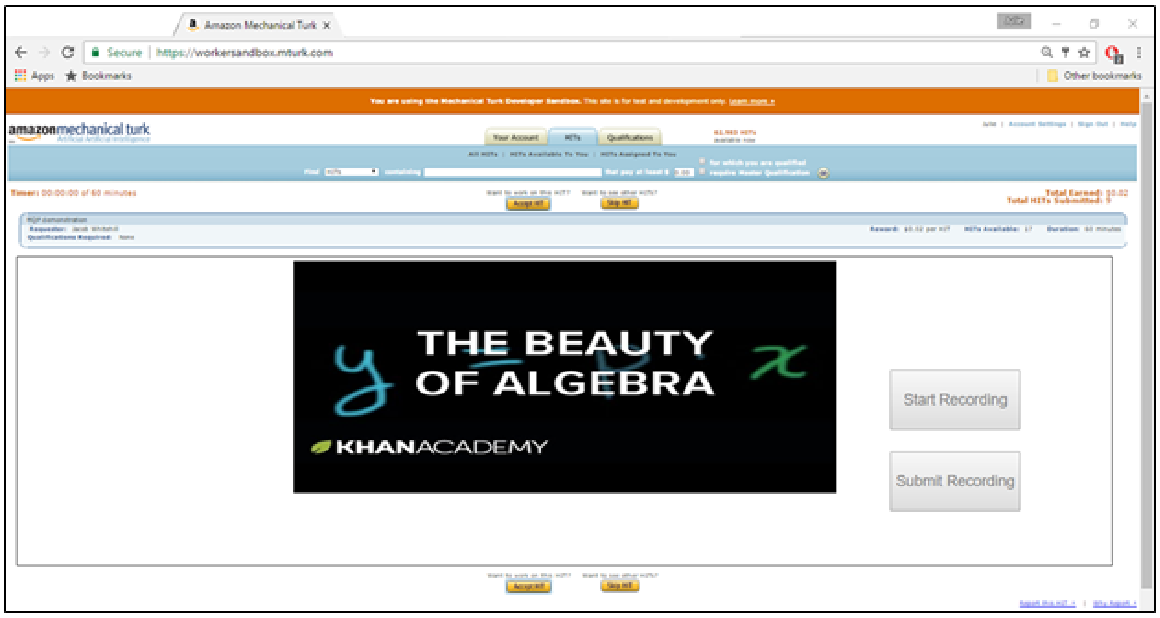


Figure 10. Screen capture of the web application when in connection to mTurk. This figure shows what the screen looks like before the subject has accepted a HIT.

This thumbnail is then replaced by the stimulus video after the subject has accepted the HIT and clicked the start recording button. This button as well as the submit recording button is shown to the right of the stimulus video. The start recording button will also begin playing the stimulus video. A design decision for this interface was to keep both buttons disabled until the subject accepted the HIT from mTurk. Once the HIT has been accepted the subject can then press the start recording button. This is shown in Figure 11.

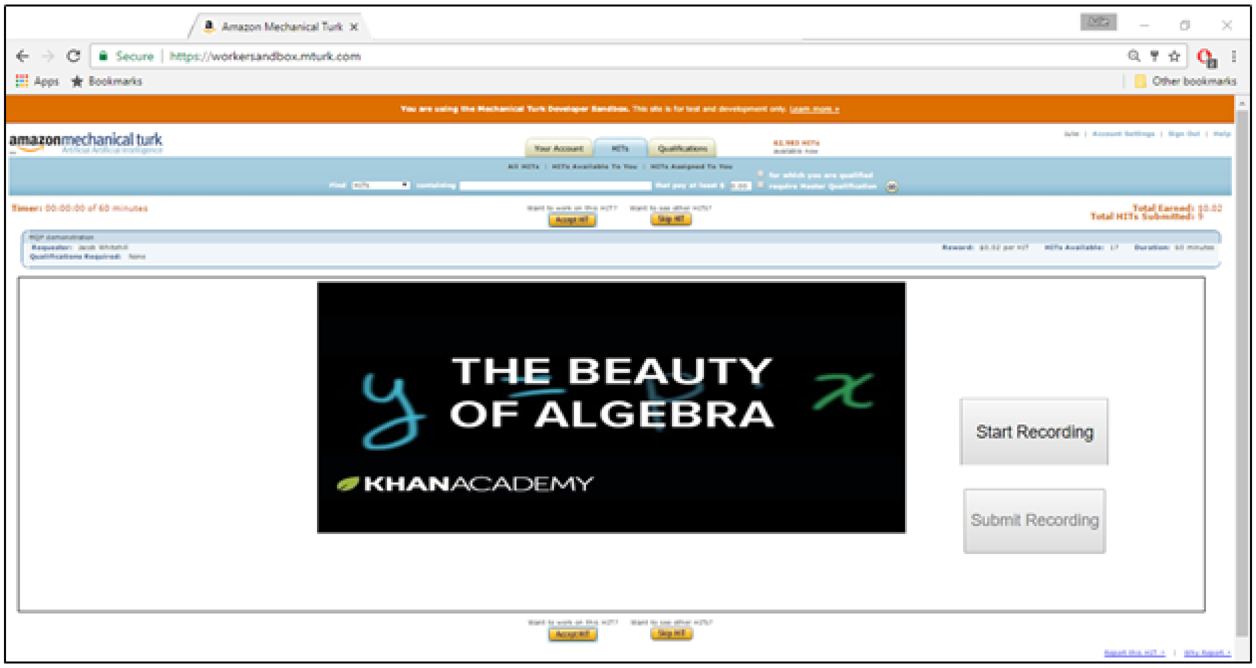


Figure 11. Screen capture of the web application when in connection to mTurk. Shows what the screen will look like after the subject has accepted a HIT. Here the start recording button is enabled.

However the submit recording button is kept disabled until a set amount of time has passed. This was chosen to keep subjects from sending experiments that are too incomplete for them to be useful. The submit recording button ends the experiment and submits the HIT.

4.10. Running Facial Analytics on Recorded Videos

Finally, in our process, the video can be inputted into the iMotions software. In order to test the process of our platform that recorded video was run through iMotions to see if the produced data was correct. The software exports raw data in large TSV files [34]. The files include data such as evidence for the 10 emotions; joy, anger, surprise, fear, contempt, disgust, sadness, confusion, frustration, and neutral. This evidence is the log-odds (base 10) that the face displays a specific emotion, conditioned on the pixels of the subject’s face. If the evidence value is positive one then that indicates the probability that the face displays that emotions is ten times as high as the probability that it does not display that emotion, based on the pixels of the subject’s face.

They also include data such as action units relating to the movement of facial muscles, as well as data on yaw, pitch and roll. The files are large because the data is presented for every frame in the recording video. Thus the information given by iMotions is detailed and time precise. The TSV files provide information to be graphed to show the subject’s facial responses to certain parts of the stimulus videos. These graphs are to be presented in the dashboard. Examples of these graphs can be seen in the Results section.

5. Results and Evaluation

This chapter describes the results & evaluation of our project. This section compares the outcomes to the project requirements. The evaluation shows that the requirements are all met and the system works as it should.

5.1. Requirements Accomplished

In order to evaluate our project we refer back to the requirements we established in section 2. The requirements were:

1. The recorded videos are of sufficient quality for iMotions to be able to detect the face
2. The recording and storage of timestamps by the recording application is successful.
3. The recording interface works as a web application.

5.1.1. Video Quality

We successfully accomplished our first requirement. The videos stored by the recording application were of sufficient quality to be processed by the iMotions software. We came to this conclusion after creating a test video with our application, and having iMotions run on this video produced accurate results. iMotions requires the participants' facial expression be at least 64 pixels wide. There is a recording display window that may be turned on to allow the test subject to position him/herself sufficiently in front of the web camera. Our recording video is 640 pixels wide and allows for sufficient room to display the subject's facial expression. Along with the iMotions requirements the project met the video requirements set to provide quality videos for facial expression use. The reaction videos are 30 fps, and average approximately 75 KB per second of recorded video.

5.1.2. Recording & Storage of Timestamps

The end product was able to successfully record and store timestamps of the subjects' actions for synchronization. The timestamps were made to store the time of the reaction video and the stimulus video on any user action. Actions are appended to a list and stored in a text file that is displayed in the same folder as the reaction video. The folder is dynamically created when a recording is submitted for approval. Using the assignment ID each assignment is completed and stored onto the web server into a folder with the assignment ID as the name. The product implements this automatically.

5.1.3. Functionality as a Web Application

The recording interface fulfilled the requirement of working as a web application. What this means is that the subjects will not need to download or install any additional software to be a part of the experiments. The subjects will be able to open the web application on their browser and use it there. This is important because having to download or install additional software is a hassle for the subject. As well as subjects might be turned away from the project because of the possibility of having to put in the extra work of installing software. Our application makes it extremely simple for the subject to use, as they only need to open it in a browser.

5.2. Overall Product Evaluation

The recording interface is designed as a web application. It is hosted on a web server and works on the Mechanical Turk sandbox mode, which acts as the same environment as the release version of Mechnical Turk, as well as working as a stand-alone application. The web server is hosted by WPI and runs on Node.js. It allows for multiple user connections and acts as a storage system for the recorded videos. The web application has a SSL certificate and is allowed to run on the Amazon Mechanical Turk environment.

The product went through several tests in order to determine if it was successful. We tested that a subject was able to preview our HIT (Human Intelligence Task) and then accept it in order to begin work. When previewing the HIT, any functionality in the web application is disabled until the HIT is accepted. This is done to prevent Mechanical Turk users from performing the HIT without receiving payment. Only when the HIT has been accepted and the recording process has begun can the HIT be submitted. If the video is not deemed acceptable, the researcher can reject it. The application also displays the submit and start recording buttons prominently on the HIT page to ensure any user performing the HIT can complete the HIT transaction with minimal confusion.

The recording web application is confirmed to be sending the reaction video in chunks to the server. The chunks can be seen in the corresponding assignment ID folder until the subject submits this HIT, which triggers a function on the web server to call ffmpeg to merge the chunks into one video. After submitting the HIT, a merged video of mp4 format is created with the previously mentioned video qualities.

In order to verify that the videos were of sufficient quality, a test trial with a student subject from WPI was performed using our recording web application. The subject was given a video of a man eating an insect and recorded through our application. The video was then passed into iMotions. The iMotions software produced a TSV (Tab Separated Values) file which was then loaded into excel and graphed to test if the video would provide useful data.

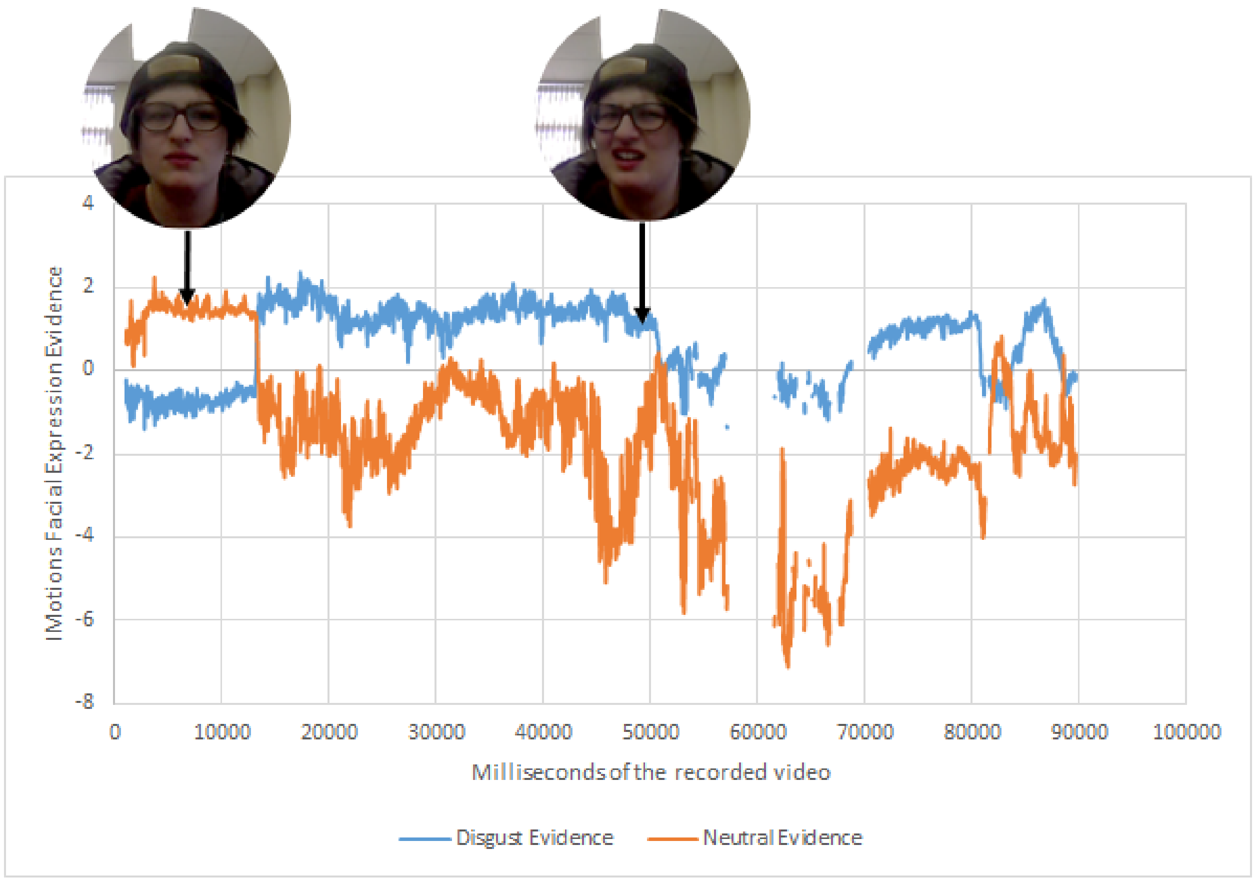


Figure 12. Displaying the subjects' emotions with the result produced by iMotions.

Figure 12 displays a sample output of iMotions graphed in Excel. The graph shows disgust versus a neutral expression as an example. iMotions also predicts joy, anger, surprise, fear, contempt, sadness, confusion and frustration. (Additional graphs can be seen in Appendix B). In this graph, each value corresponds to a predictive score based on the expression of the test subject. The graph shows that a video recorded through our platform can be used by iMotions to display results correctly.

As seen in figure 12, the subject showed disgust around the 50000 millisecond mark which is displayed in the graph results. Near the beginning of the video, the subject shows a neutral expression which is also represented in the graph by the elevated line for neutral. Since the graph results proved to be meaningful, we conclude that our first two modules were completed successfully and managed to record and transmit videos for use in iMotions. The blank portions of the graph represent the video frames in which no face was found, for example if a hand was covering the face.

6. Future Work

This section provides suggestions on what should be added and improved to have the fully working system that was mentioned in the Product Description section. Section 2 describes the different modules that need to be implemented for the product. The first module referring to the web application that records the reaction of a user watching a stimulus video. The second module is a server which hosts the web application and handles the recordings sent from the user. The third module has a couple of steps: it transfers the videos from the server to the machine hosting the facial emotion recognition software, it processes the reaction video through the software, and sends the results back to the server. The fourth module takes the results from the third module and displays them along with the reaction and stimulus videos.

Over the course of this project we implemented the first and second module, but the rest of the process is currently done manually. We have a few suggestions for the creation of the third and fourth modules in order for it to be a completely autonomous function system.

For the third module we have outlined how it could work and how it could be done. First, the Node.js server will be signaled when a recording is completed and received. Then a File Transfer Protocol will initiate and send the video to the Emotion Recognition Machine. On the machine there will be a script that will be signaled when a new reaction video is received. At this point, the reaction video will be processed through the emotion recognition software with a script. After that the result a (.TSV file) will be sent back so the server and placed in the proper folder based on the reaction video sent.

For module four, a dashboard web application needs to be created to display the reaction video, stimulus video, and the results of the emotion recognition software. The reaction video and stimulus need to be played in synchrony to reflect what the user was watching. The timestamp file will be used to synchronize the videos. On the dashboard application a timer would be initialized in the background to compare the clock time of when an action was made in the timestamp file to that timer. When the two times are equal or very close to each other (~10 ms), the action from the timestamps list should be executed on the YouTube player.

During development we did not complete the implementation of the video synchronization aspect of the dashboard. We used video.js to play the reaction video on the interface. Video.js is a video player framework that can be integrated with the node.js server. We managed to get the two videos to be in synchrony in a single play through where the reaction video is playing and the actions are applied to the stimulus video based on the timestamps list. However, neither of the videos can be controlled in our implementation. This is because providing the ability to control the videos brings the need to keep track of more variables and states.

The other aspect of the dashboard is the emotion recognition charts. From our brief research we found multiple JavaScript libraries that can be used to create line charts, but we have not had the chance to test them. Those libraries are: chart.js, and some d3 libraries like c3.js. After implementing the line chart, it can be overlaid with a timeline that is controllable. This timeline can be based on the reaction video time and when it is clicked the reaction video will jump to that time.

In the future, the dashboard can expand to accommodate multiple researchers. Each researcher can login to see his/her experiments. Additionally, multiple researchers could set experiments at the same time. All the things mentioned previously will help in improving the system and enhance its functionality.

7. Conclusion

In conclusion, the goal of this project was to help researchers conduct large scale facial analysis experiments. In order to accomplish this goal, we divided and developed the platform in four modules. The first module is a web application that records the subject's facial expression as they are presented with a video, and sends it to a web server. The second module is the web server. It receives the facial reaction videos and stores them in a directory along with timestamp data on the videos. The third module is a process in which the reaction videos from the server were sent to the machine with the facial expression analysis software.

The software is run on the videos, and the videos are then sent back to the server. The fourth module is a dashboard application in which the reaction videos and their corresponding stimulus videos are shown in sync with the facial expression data for the researcher. This project focused on and completed modules one and two. Modules three and four are described as future work, as a possibility for another group.

This project had four requirements:

1. Record and transmit reaction videos recorded of sufficient certain quality and format for use by the facial expression recognition software.
2. Record and transmit Timestamps need to be recorded for user interaction with the stimulus video.
3. Store recorded videos and their corresponding timestamp files in a proper directory system on the server.
4. Develop the video recording module as a web application

The final project accomplished all requirements. Researchers can use this product to quickly and easily obtain data for facial analytics experiments.

8. References

[1] "A Browser Market Share". September 2016. Accessed October 05, 2016. https://www.netmarketshare.com/browser-market-share.aspx?qprid=0.

[2] Analytics Without the Bots. McDuff, Daniel, Abdelrahman Mahmoud, Mohammad Mavadati, May Amr, Jay Turcot, and Rana El Kaliouby. "AFFDEX SDK: A Cross-Platform Real Time Multi-Face Expression Recognition Toolkit." Alumni Media MIT. May 7-12, 2016. Accessed October 05, 2016. http://alumni.media.mit.edu/~djmcduff/assets/publications/McDuff_2016_Affdex.pdf.

[3] "Apache, Apache. 2016. Accessed October 07, 2016. https://www.apache.org/

[4] "AutoIt - AutoIt." AutoIt. 2015. Accessed October 07, 2016. https://www.autoitscript.com/site/autoit/.

[5] Bosch, N., D'Mello, S., Baker, R., Ocumpaugh, J., Shute, V., Ventura, M., & Zhao, W. (2015, March). Automatic detection of learning-centered affective states in the wild. In Proceedings of the 20th international conference on intelligent user interfaces (pp. 379-388). ACM.

[6] "Browser & Platform Market Share." W3Counter: Global Web Stats. September 30, 2016. Accessed October 05, 2016. https://www.w3counter.com/globalstats.php.

[7] CollegeDegrees360. "Confused". Posted [July 2012] Online Image. Flickr. 11 October, 2016. https://www.flickr.com/photos/83633410@N07/7658298768

[8] Dutton, Sam. "Simple.info: Simplest Possible Examples of HTML, CSS and JavaScript." MediaStream Recording. Accessed October 05, 2016. https://simpl.info/mediarecorder/.

[9] "Emotion Recognition Technology - Affectiva." Affectiva. Accessed October 05, 2016. http://www.affectiva.com/technology/.

[10] "FaceReader." Facial Expression Recognition Software: FaceReader. 2016. Accessed October 05, 2016. http://www.noldus.com/human-behavior-research/products/facereader.

[11] "Facial Expression Analysis the Definitive Guide." IMotions_Guide_FacialExpressions. 2016. Accessed October 05, 2016. http://nefsummit.org/wp-content/uploads/2016/06/iMotions_Guide_FacialExpressions_2016.pdf.

[12] "Facial Expression Analysis Solutions - IMotions." IMotions. Accessed October 05, 2016. https://imotions.com/facial-expressions/.

[13] Ffmpeg, Ffmpeg Home Page. Accessed October 10, 2016 https://www.ffmpeg.org/

[14] Hofstetter, Thomas. "Online Video Trends for 2016 - What You Need to Know." Points Group LLC. 2016. Accessed October 05, 2016. http://www.pointsgroupllc.com/online-video-trends-2016/.

[15] “HTML5”, Mozilla Firefox. Accessed October 07, 2016 https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

[16] "Installation and Configuration." PHP:. Accessed October 07, 2016. http://php.net/manual/en/install.php

[17] "JS Bin." - Collaborative JavaScript Debugging. October 23, 2015. Accessed October 05, 2016. https://jsbin.com/zomujiyexu/1/edit?js,console,output.

[18] Khan Academy. "ELI the ICE man". Filmed [Aug 2016]. YouTube video, 2:32. Posted [Aug 2016]. https://www.youtube.com/watch?v=2yqjMiFUMlA

[19] Khan, Muaz. "MediaStreamRecorder Demos." WebRTC » MediaStreamRecorder ® Muaz Khan. August 2016. Accessed October 05, 2016. https://www.webrtc-experiment.com/msr/.

[20] Knight, Will. "Facial Analysis Software Spots Struggling Students." MIT Technology Review. July 01, 2013. Accessed October 05, 2016. https://www.technologyreview.com/s/516606/facial-analysis-software-spots-struggling-students/.

[21] LOHR, STEVE. "As Travel Costs Rise, More Meetings Go Virtual." The New York Times, July 22, 2008. Accessed October 06, 2016. http://www.nytimes.com/2008/07/22/technology/22meet.html.

[22] Mansfield, Matt. "What You Need to Know About Changing From Http to Https." Small Business Trends. April 22, 2015. Accessed October 05, 2016. https://smallbiztrends.com/2015/04/changing-from-http-to-https.html.

[23] "MEAN.IO - MongoDB, Express, Angularjs Node.js Powered Fullstack Web Framework MEAN.IO - MongoDB, Express, Angularjs Node.js Powered Fullstack Web Framework." MEAN.IO. 2014. Accessed October 05, 2016. http://mean.io/#!/.

[24] "MediaRecorder API." Mozilla Developer Network. July 20, 2016. Accessed October 05, 2016. https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder_API.

[25] "MediaStreamRecoder Demo (w/ MediaSource)." MediaStreamRecoder Demo (w/ MediaSource). Accessed October 05, 2016. https://rawgit.com/Miguelao/demos/master/mediarecorder.html.

[26] Mozilla Developer Network. September 26, 2016. Accessed October 05, 2016. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe.

[28] Newman, Jared. "The Agonizingly Slow Decline of Adobe Flash Player." Fast Company. August 18, 2015. Accessed October 07, 2016. https://www.fastcompany.com/3049920/tech-forecast/the-agonizingly-slow-decline-of-adobe-flash-player.

[29] "Node.js." Node.js. 2016. Accessed October 05, 2016. https://nodejs.org/en/.

[30] Ramsey, By Doug. "Computer Software Accurately Predicts Student Test Performance." UC San Diego News Center. April 14, 2014. Accessed October 05, 2016. http://ucsdnews.ucsd.edu/pressrelease/computer_software_accurately_predicts_student_test_performance

[31] Ronacher, Armin. "Welcome." Flask Web Development, One Drop at a Time. 2016. Accessed October 07, 2016. http://flask.pocoo.org/.

[32] Scott Frees. "A place for node.js in the computer science curriculum". J. Comput. Sci. Coll.30, 3 (January 2015), 84-91.

[33] "State of the MOOC 2016: A Year of Massive Landscape Change for Massive Open Online Courses." Online Course Report. 2016. Accessed October 06, 2016. http://www.onlinecoursereport.com/state-of-the-mooc-2016-a-year-of-massive-landscape-change-for-massive-open-online-courses/.

[34] Taggart, Richard W., Michael Dressler, Poonam Kumar, Shahroze Khan, and Jean F. Coppola. Determining Emotions via Facial Expression Analysis Software. Pace University. Accessed 2016.

[35] Valstar, M. F., Almaev, T., Girard, J. M., McKeown, G., Mehu, M., Yin, L., & Cohn, J. F. (2015, May). Fera 2015-second facial expression recognition and analysis challenge. In

Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on (Vol. 6, pp. 1-8). IEEE.

[36] "What Is HTTPS?" HTTP to HTTPS. 2016. Accessed October 05, 2016. https://www.instantssl.com/ssl-certificate-products/https.html.

[37] Whitehill, J., Serpell, Z., Lin, Y. C., Foster, A., & Movellan, J. R. (2014). The faces of engagement: Automatic recognition of student engagement from facial expressions. IEEE Transactions on Affective Computing, 5(1), 86-98

[38] "Working on HITs." Amazon Mechanical Turk - Welcome. 2016. Accessed October 07, 2016. https://www.mturk.com/mturk/welcome?variant=worker.

[39] "YouTube Player API Reference for IFrame Embeds | YouTube IFrame Player API | Google Developers." Google Developers. August 11, 2016. Accessed October 05, 2016. https://developers.google.com/youtube/iframe_api_reference.

9. Appendices

9.1. Appendix A

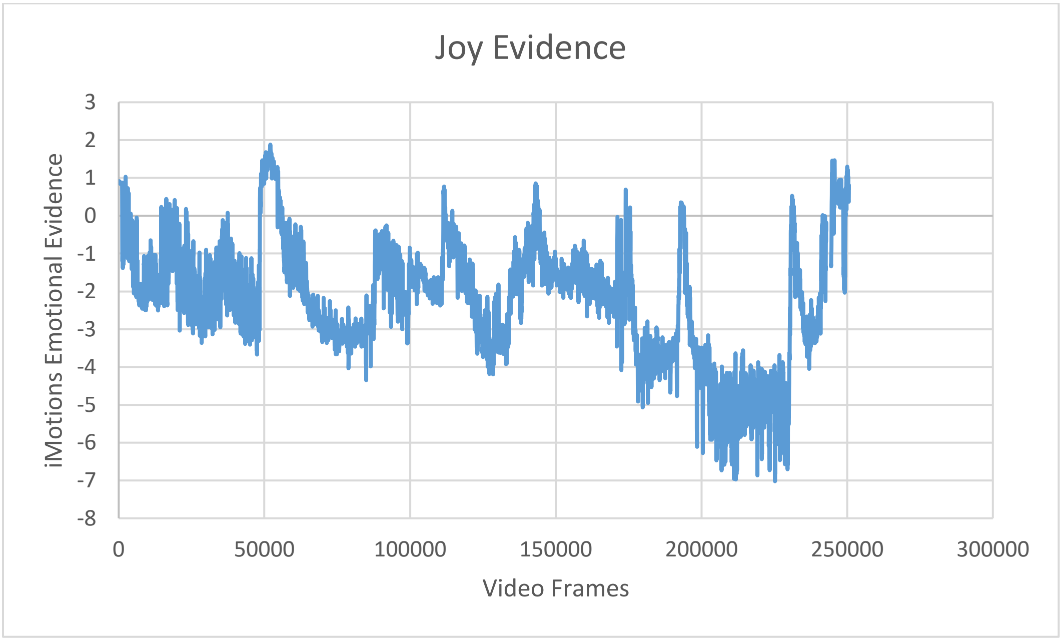
Specs of the machines used to test our product with the recording application

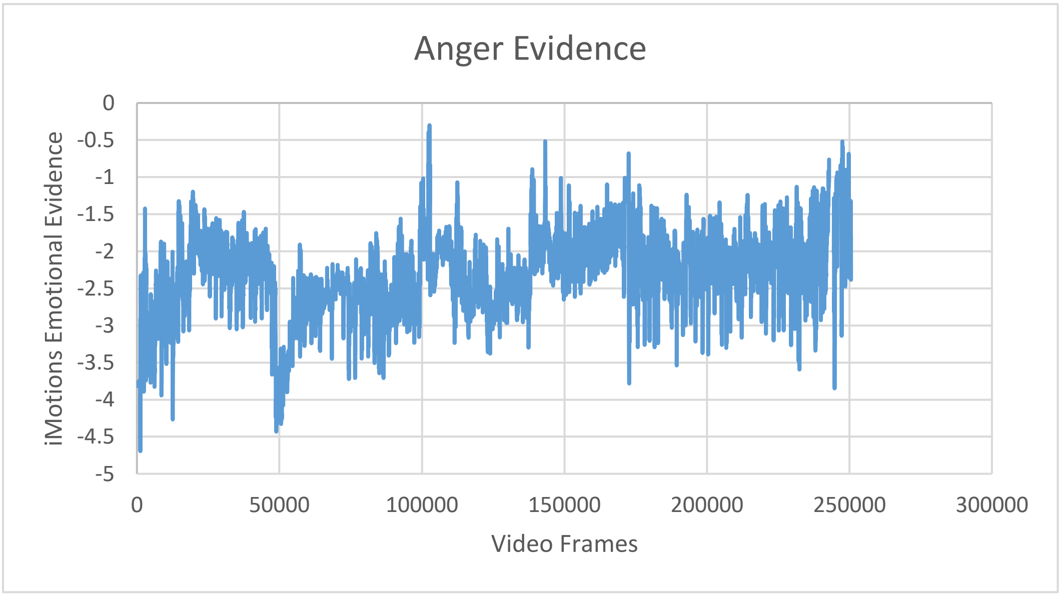
Specs	CPU	RAM
Gordon's Machine	AMD A6-5357m up to 3.5 GHz Dual Core	4GB
Julie's Machine	i7-3537u up to 2.5GHz Dual Core	8GB
Rayan's Machine	i7-4720HQ 2.6 GHz Quad Core	16GB

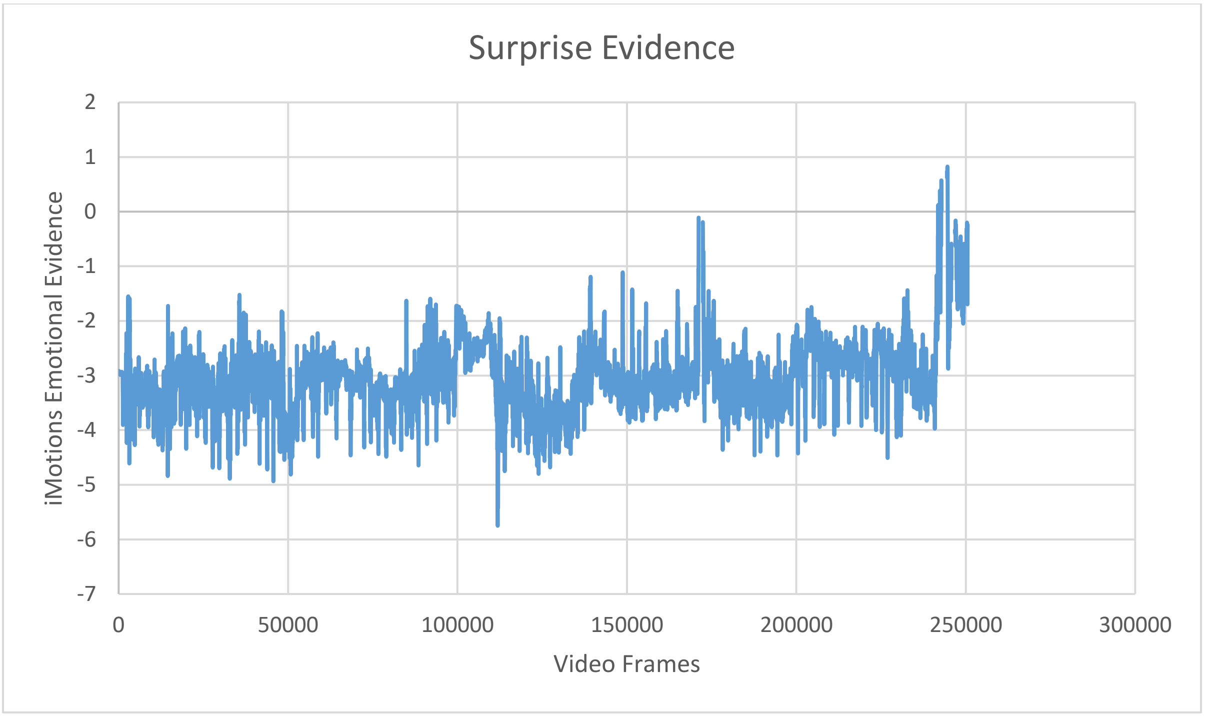
9.2. Appendix B

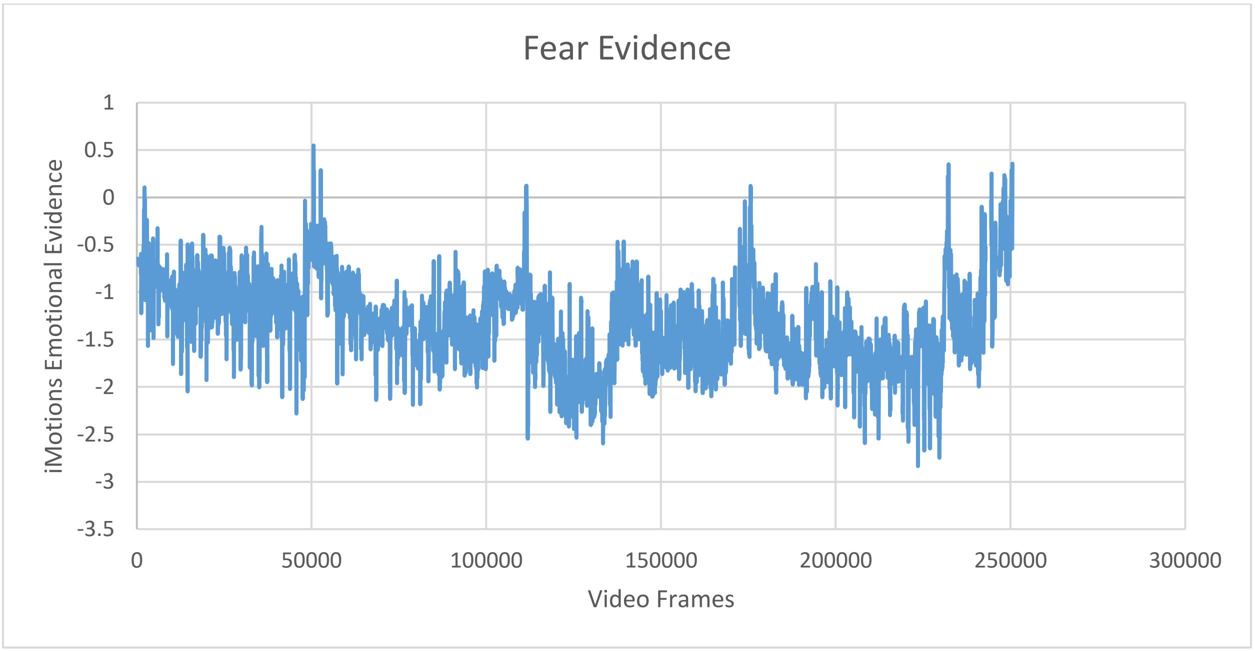
Additional Graphs from iMotions data

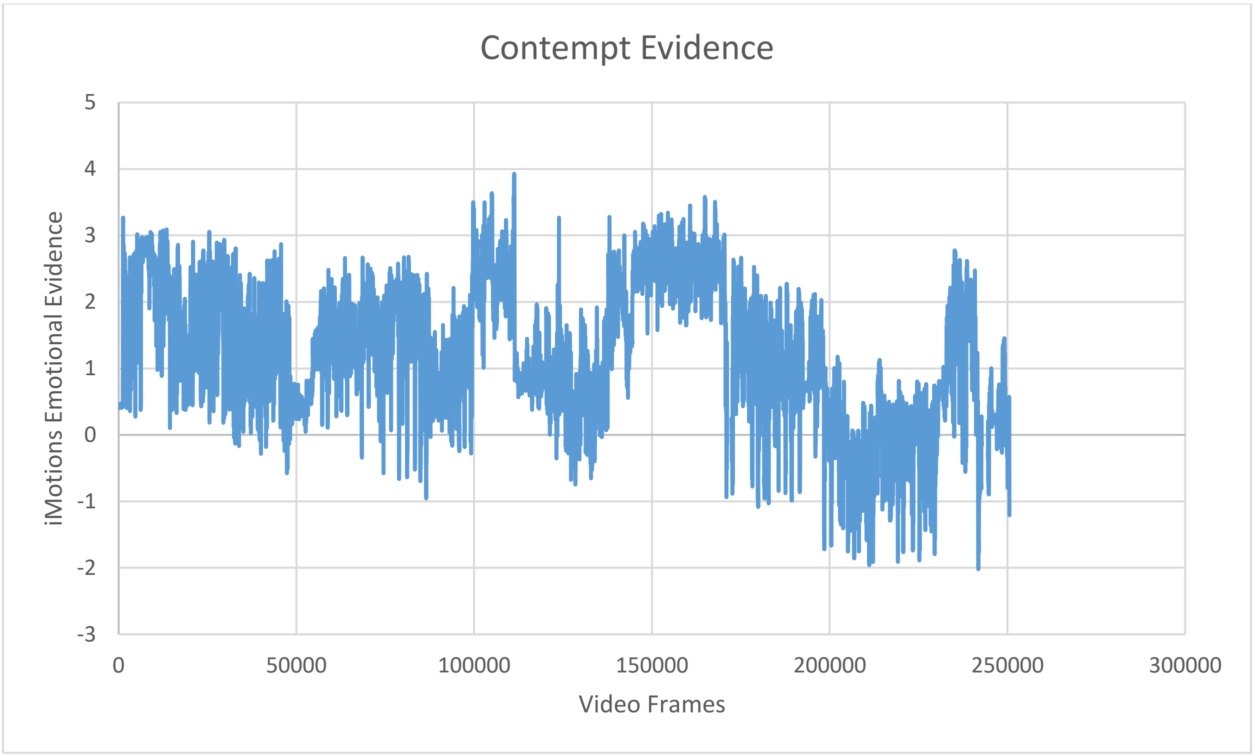
Each graph below represents each of the ten emotions that iMotions analyzes. For each graph, the x-axis represents the frames of the video. The y-axis represents the evidence of the emotions as analyzed by iMotions.

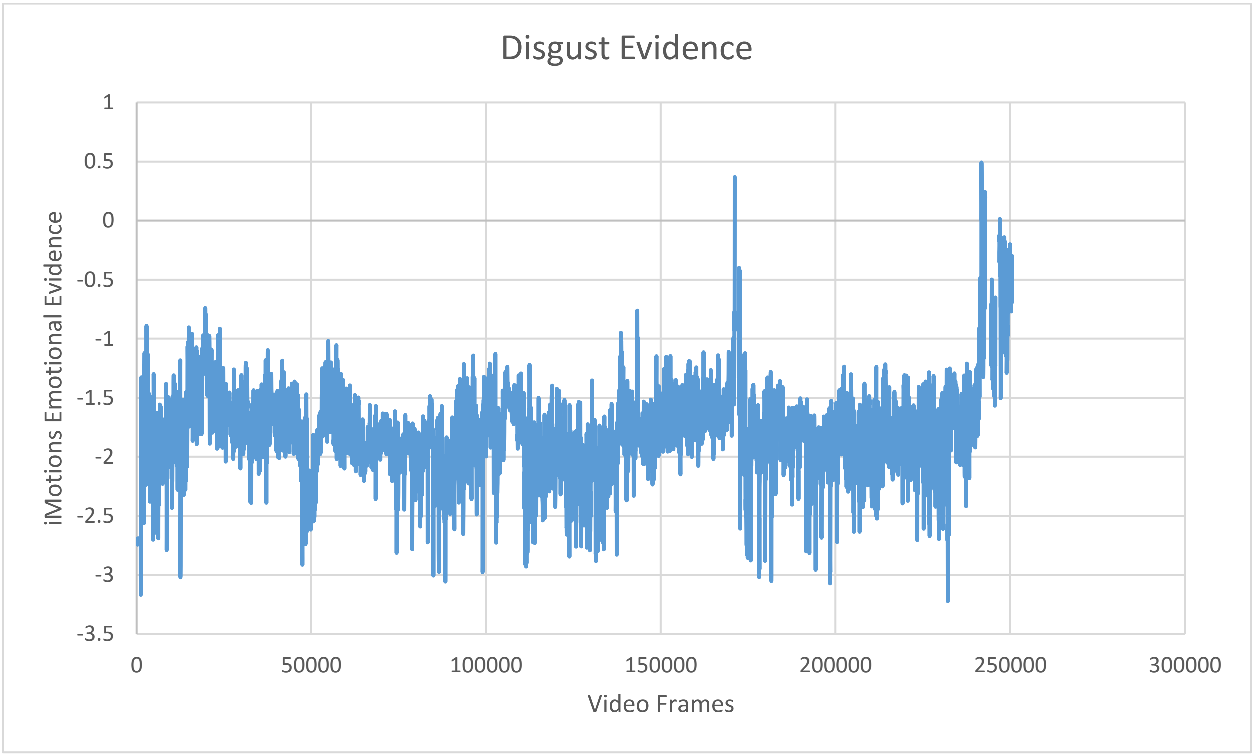


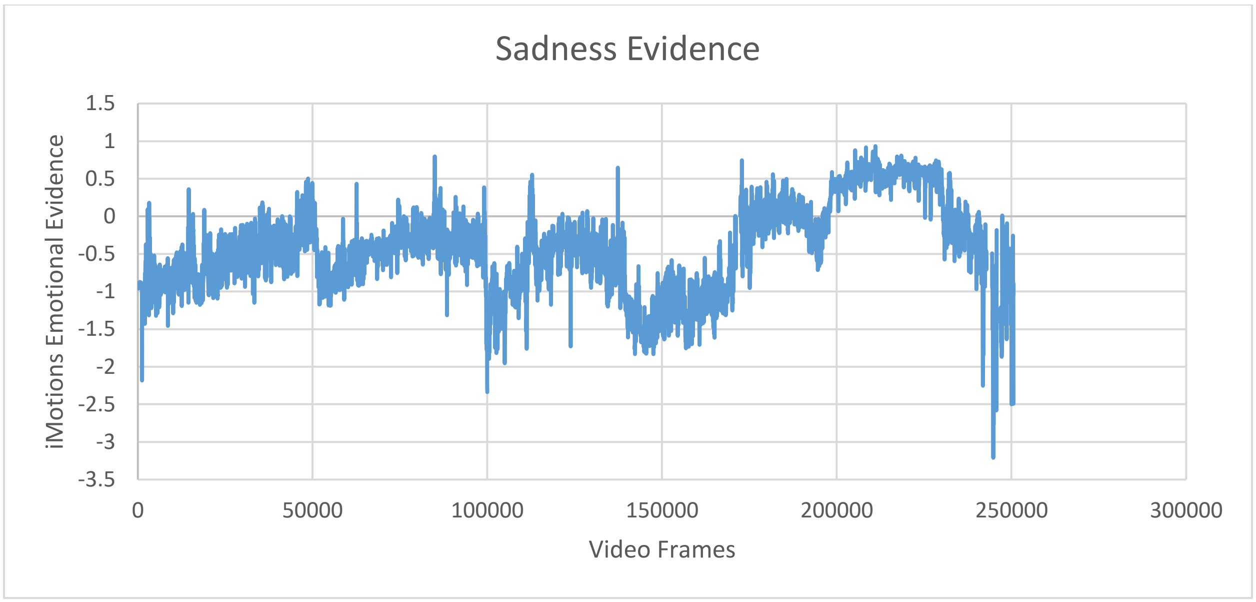


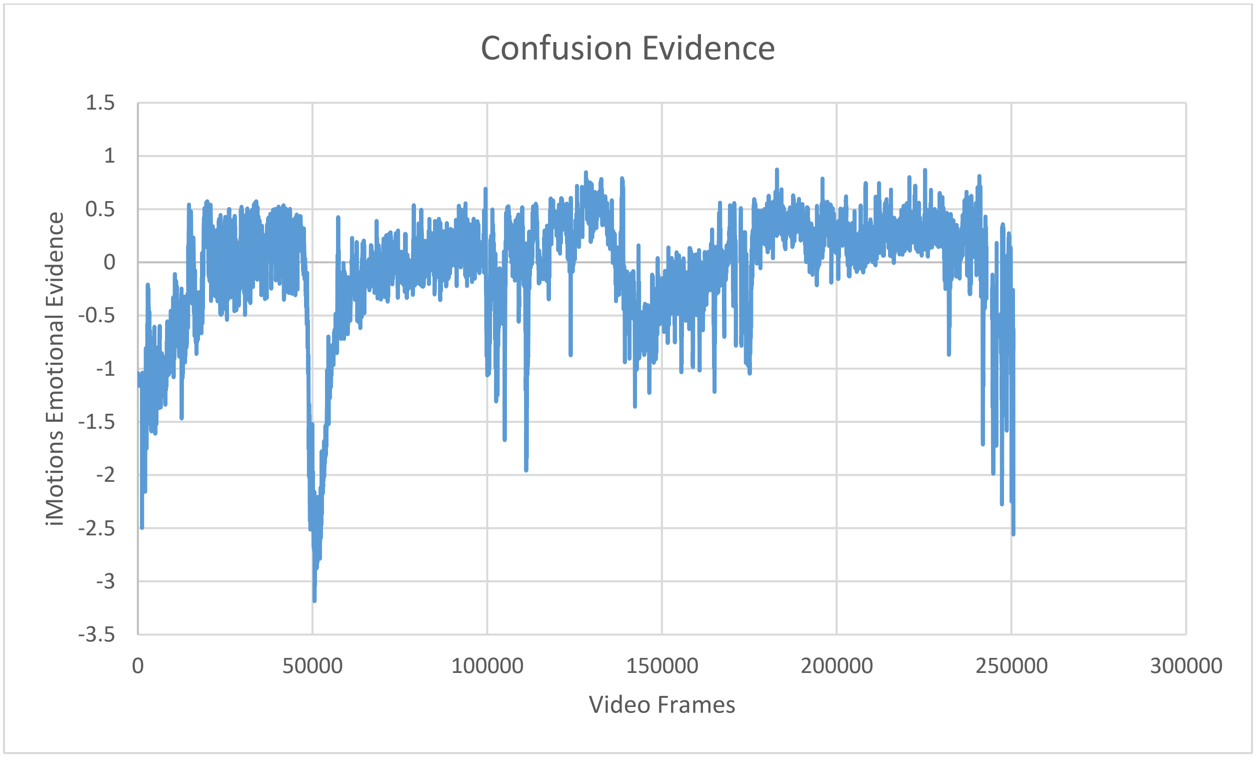

Surprise Evidence
iMotions Emotional Evidence
Video Frames
2
1
0
-1
-2
-3
-4
-5
-6
-7
0
50000
100000
150000
200000
250000
300000

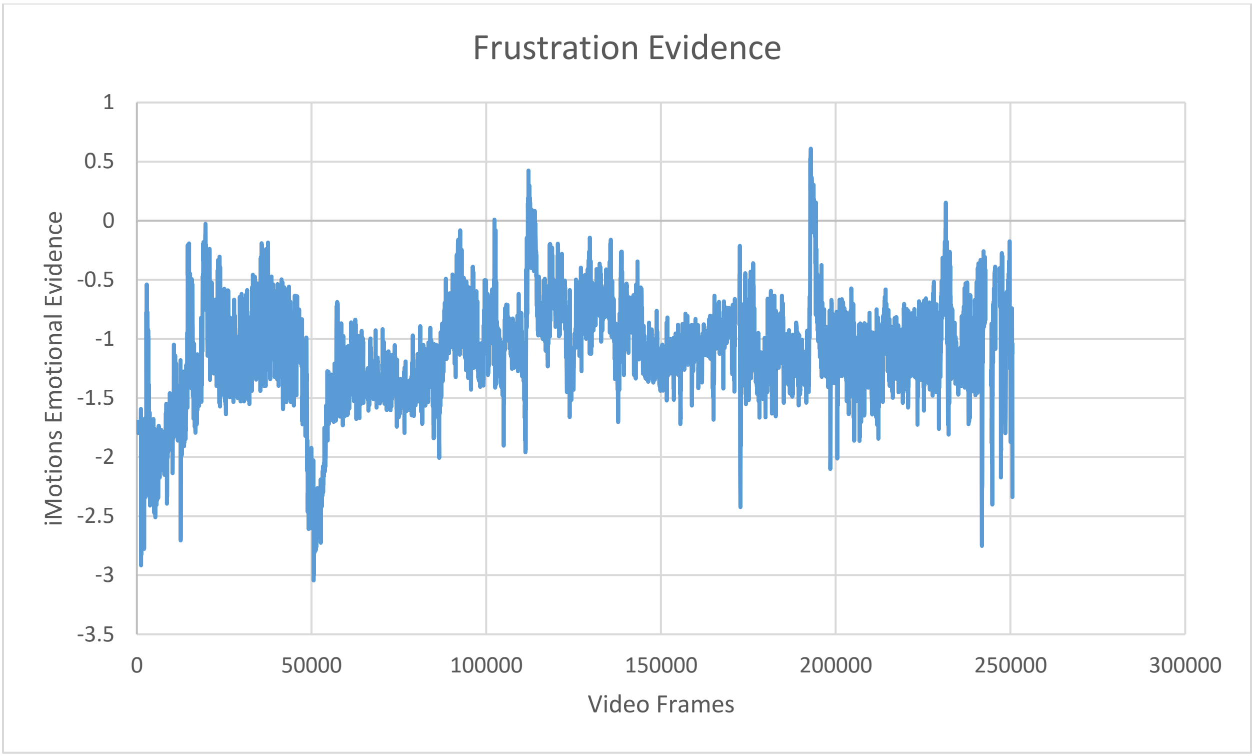

Fear Evidence
iMotions Emotional Evidence
Video Frames
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
50000
100000
150000
200000
250000
300000

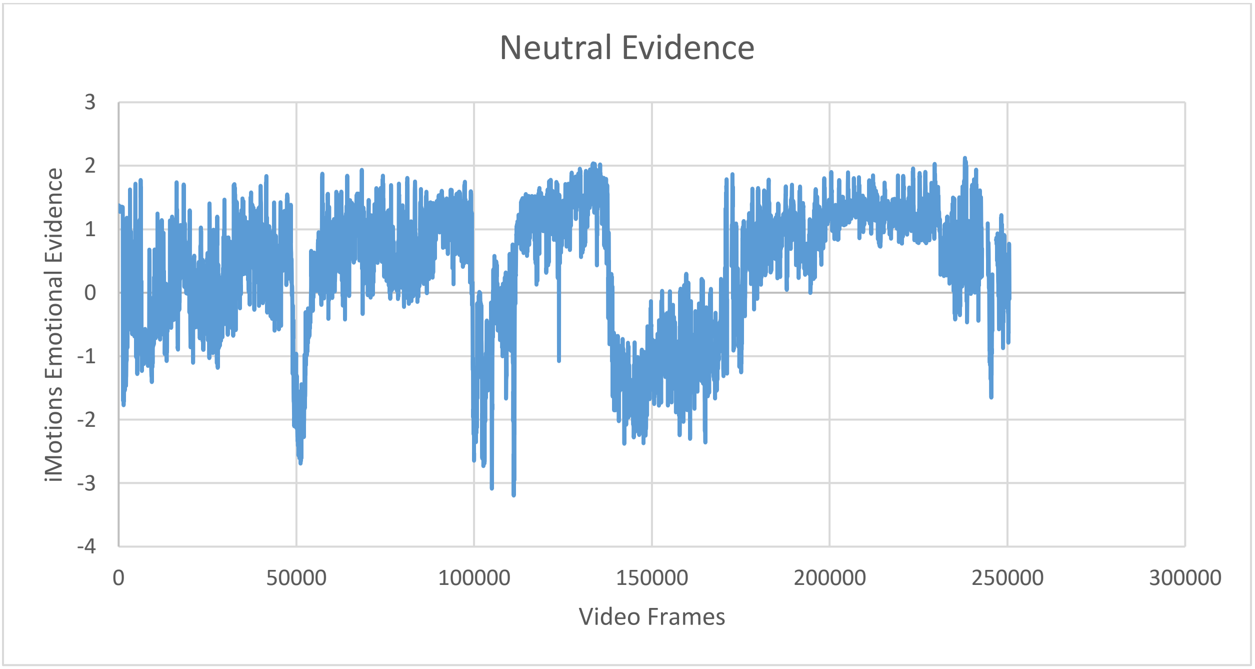

Contempt Evidence
iMotions Emotional Evidence
5
4
3
2
1
0
-1
-2
-3
0
50000
100000
150000
200000
250000
300000
Video Frames


Disgust Evidence
iMotions Emotional Evidence
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
50000
100000
150000
200000
250000
300000
Video Frames


Sadness Evidence
iMotions Emotional Evidence
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
50000
100000
150000
200000
250000
300000
Video Frames


Confusion Evidence
iMotions Emotional Evidence
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
50000
100000
150000
200000
250000
300000
Video Frames


Frustration Evidence
iMotions Emotional Evidence
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
50000
100000
150000
200000
250000
300000
Video Frames


Neutral Evidence
iMotions Emotional Evidence
3
2
1
0
-1
-2
-3
-4
0
50000
100000
150000
200000
250000
300000
Video Frames