PYfocals

Software Project Plan

Tim Fanelle, Christopher Hufham, Patrick Kenny, and Jaime Lujan

Table of Contents

Introduction

Project objectives

Building off existing face-tracking technologies, our software will be a program to map common facial movements to generic inputs on a computer as determined by the user. For example, the user could bind a winking motion to the F11 key. The ability to control other software in this way will enhance accessibility for users and can be easily modified for a multitude of uses. We will demonstrate one such use by mapping the facial movements to separate buttons on a virtual controller allowing users to play games with only facial movements.

Major functions

- Access to user's webcam
- Interpreting varied facial movements into distinct categories
- Continuously watching the user's face in wait of a recognized input
- Binding facial movements to keyboard inputs through the operating system
- Interfacing keypresses to other software

Scope

The scope of the facial tracking features will largely be determined by the functionality provided in the libraries we will use, which will be further developed as we move forwards in the project.

Speed may be a performance issue due to differing frame rates between the user’s camera and the frame rate of software the user is mapping inputs to. Should it require real-time inputs (like a video game), there could potentially be significant issues with the target software depending on exactly how timing-dependent it is.

The effectiveness of the software will largely be dependent on the user’s hardware and environment. The user’s webcam must have high enough capture quality that the algorithm is able to recognize changes in the facial components. Because of this, there are many possible issues with the user’s environment that could impede the face tracker’s functionality, such as inadequate lighting. While these issues are unavoidable, we will do our best to mitigate them.

Project Estimates

Historical data used for estimation

Because of advancements in facial recognition over the years, the time to write code to detect basic facial features should only be a few hours. The jump from detecting facial features to facial actions is a few days on average. Aswell, the average time to get actions (of any kind) to bind to keyboard inputs should be largely copy-and-paste after the first one.

Estimates

Since our project is building off already existing software, a significant amount of the development time will be spent reading documentation of previously-created open-source projects and examples. Using a three point estimation technique we can estimate this project to take about 9 days worth of work divided across 4 people.

As of the completion of this project, the amount of work estimated was fairly accurate. However, the amount of time it took to finish the work was much longer than expected. Problems such as other classes and uneven skill sets affected the time more than expected.

Project Risks

Risk matrix

	Negligible	**Marginal**	**Critical**	**Catastrophic**
Likely		Minor issues with opencv and other libraries		
Possible	Meeting / communication problems	Trouble with differing webcams / devices	Major issues with opencv and other libraries	
Unlikely			Not enough time to finish construction	Inability for software to distinguish between enough unique facial movements
Rare			Library incompatibilities with particular hardware	

Risk identification and resolution

- **Minor issues with opencv and other libraries:** As we are using libraries that the majority of the team are unfamiliar with, getting used to their individual intricacies and minutia will take time. However, all members are accustomed to learning new frameworks quickly.
- **Meeting / communication problems:** As there are many members of the team, it is always a possibility that meeting conflicts could arise due to busy schedules. Additionally, communication errors between members can be an issue. Both of these risks are helped mitigated by having comprehensive documentation so that team members are always on the same page, and having the documentation hosted on the cloud so that if a member cannot make it to a meeting they will still understand how to proceed with development.
- **Trouble with differing webcams / devices:** Not all webcams are made equal, so there will be variance in how the software will perform with them. Low frame rates and image quality are the main issues concerning different video capture devices. The best way to test for these issues is to try the software using as many different devices as possible.
- **Major issues with opencv and other libraries:** Once again, the majority of the team is largely unfamiliar with the libraries we are

planning to use. There is a slight possibility that we are unable to use them for the exact purposes we are planning to. Despite the large amount of time we have spent researching the various functionality and examples given for these libraries, this will remain a possibility until the major functions of the software are implemented.

- **Not enough time to finish construction:** There is a very limited amount of time left in the semester, and it must be stretched between all of our classes. It is a possibility that we reach a point where we do not have the time required to construct the software exactly according to specification. This risk is slightly mitigated by making sure we do not intend to include any features that aren't necessary.
- **Inability for software to distinguish between enough different facial movements:** While we have found comprehensive examples for facial tracking of certain aspects of the face, others must be left to our own ingenuity. There is a risk that we will not be able to include the features we have intended to. Should this happen, the software will still be completed, but will not contain enough functionality to be useful for its original intended purpose.
- **Library incompatibilities with particular hardware:** The libraries we are planning to use are not expected to encounter any issues with specific hardware brands or interfaces. However, if they do, then the software will be entirely unusable on machines with that hardware.

Software Quality Assurance

Description

Quality of the software will be assured through multiple different practices. All code will be examined by every member of the team after it is written to ensure it is a correct solution to the problem at hand. The outline of the program’s code was designed with every member’s input and participation. We are using a repository on GitHub to hold the codebase in a public cloud where the most recent changes can be viewed at any time for reference. Additionally, it allows us to roll-back a change should it break functionality or introduce major bugs. Features can also be constructed on separate branches so that they do not conflict during development. Using unit tests was considered, but due to our unfamiliarity with them and the restrictive amount of time left for the project it was ultimately decided to be an inefficient use of our remaining time.

Schedule

Project work breakdown structure

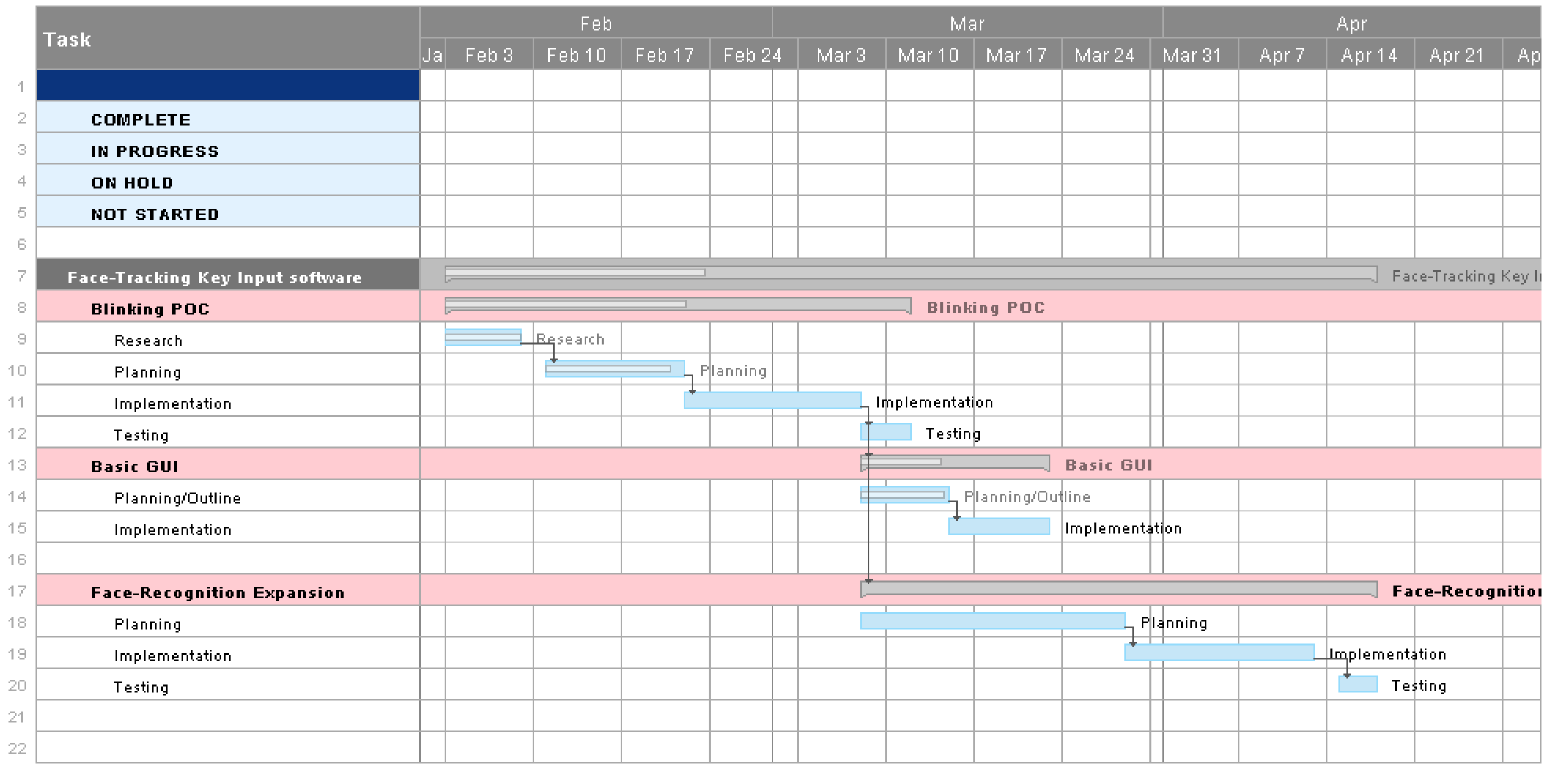
We are currently using the waterfall model as our development model because it works well within a class format. More specifically, we plan to follow the communication, planning, modeling, construction, and deployment steps in order without changing much. Below is the estimated structure of the schedule.

- **Weeks 1-2, Team formation and brainstorming:** This initial period is to get a feel for the class, assignment, and team. After that, the goal is to brainstorm a software idea that can be finished by the end of the class and agreed upon by all the members of the team.
- **Weeks 3-4, Developing outline and plans:** After the concept of the software is decided, an outline of it will be created. This is so that we can decide what aspects should and shouldn't be included in our time frame.
- **Weeks 5-6, Modeling and UI design:** After the scope of the project is defined, the functionality needs to be more formally specified. This should be done by modelling and explaining the specifics of each

feature of the program. Additionally, informal mockups of the UI should be created so that the design of the software is clear.

- **Weeks 7-8, Design and code planning:** Once all the functionality is specified, the design of the code must also be specified. This should include planned classes, functions, data structures, and more. This is so that the actual coding process goes as smoothly as possible without any major conflicts or time losses.
- **Weeks 9-12, Construction:** This period of time will be spent coding the project. By the end of this period, all features should have complete functionality, but will not be expected to be bug-free.
- **Weeks 13-15, Testing and deployment:** The final portion of the schedule is time reserved for testing and debugging. We plan to test it as comprehensively as we can, but due to the nature of the software and its deadline there will be cases we can't check for. Once we have tested as much as we can within the time frame, the software will be packaged up for distribution and finalization.

Timeline chart (Gantt chart)



Project Resources

People

Tim is the most experienced with OpenCV and is currently using it for an honors thesis project with Dr. Narayan. Patrick is skilled in reading documentation and finding examples of similar projects for us to research. Chris is adept at designing models and systems for the software functionality. Jaime has experience in handling real-time software inputs and control.

Additionally, every member of the team is familiar with coding practices and techniques in python, which should make for rapid development both because of our familiarity and the general nature of the language.

Hardware

To run the software, the user will require a laptop or desktop PC with a webcam device connected. The framerate and image quality of the webcam will significantly impact the usefulness of the software. We will be developing on Windows machines, but will be cross-platform with macOS and Linux should the libraries we use support them.

Processing speed is not expected to be a major issue, but it is possible that problems could arise because of it.

Software

We are looking through various libraries and frameworks for facial tracking and other useful technologies such as opencv, tracking.js, and dlib.

We are expecting to use python as our main coding environment, due to its large number of external libraries and quick coding style. Once we begin construction, we will keep the codebase in a repository on Github for source control and collaboration.

Activity Log

Reading time

We are researching articles and tutorials on the internet for related technologies and applications. Examples are listed in the appendices.

Meeting log

- **1/25/19:** This time was spent mostly learning about the project requirements and the formation of the team as a whole. After that, we brainstormed different ideas for our project. We hadn't finalized our goal by this point, but we were beginning to narrow down the area in which we knew we would be working. The meeting lasted approximately 90 minutes.
- **1/31/19:** The beginning of this meeting had every member of the team present different ideas of applications for our project. Ultimately, we settled on the idea of facial tracking being used as inputs. We began to scope out similar projects and examples to use as groundwork. We also started to fill in the outline for this document. This meeting lasted approximately 90 minutes.

- **2/18/19:** After the first pass of this document was complete, we began to work on the model document. In order to do that, we had to more comprehensively define the functionality of the software. We created many diagrams detailing the specifics of the processes we were hoping to employ. We also made rough mockups of the UI so that we could all share a similar vision of the intended final project. This meeting lasted approximately 90 minutes.
- **3/25/19:** This meeting was mostly spent planning our presentation. The majority of our time was devoted to understanding how we could present the most important aspects of our software in the clearest and most efficient way. After we had that planned out, we began to briefly discuss the design specification document and how to best approach it. This meeting began at 4:10 PM and ended at 5:05 PM, lasting a total of 55 minutes.
- **3/28/19:** This meeting was centered around the design specification document. In order to do this, we had to plan ahead for what we thought was the best way to code the software. We began designing the specifics of the algorithms used to detect facial movements. We additionally began creating formal mockups of the UI. This meeting began at 4:52 PM and ended at 5:45 PM, lasting a total of 53 minutes.

- **4/3/19:** This meeting was mostly spent cross-referencing the various progress made since the previous meeting. We made sure that we were on the same page on topics such as file structure/variable naming conventions, also making sure that the goals for the design specification document were consistent and attainable. This meeting began at 5:40 PM and ended at 6:35 PM, lasting a total of 55 minutes.

Appendices

Articles we will be researching:

- https://www.pyimagesearch.com/2017/04/17/real-time-facial-landmark-detection-opencv-python-dlib/
- https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/