**Software Requirements Specification**

**for**

**Pupil Detection System: GUI**

**Version 1.0 approved**

**Prepared by Lauren Ernst, Colleen Pawlicki, & Jim Reilly**

**BIEN 4290**

**4/10/2018**

**Table of Contents**

**Table of Contents……………………………………………………………………..……...1**

**Revision History…………………………………………………………………………..….2**

**1. Introduction………………………………………………………………………………..3**

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Product Scope

1.5 References

**2. Overall Description………………………………………………………………………..4**

2.1 Product Perspective

2.2 Product Functions

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

**3. External Interface Requirements………………………………………………………...6**

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

3.4 Communications Interfaces

**4. System Features…………………………………………………………………………...7**

4.1 File Input

4.2 Start Button

4.3 Image Display

4.4 Coordinates View

**5. Other Nonfunctional Requirements……………………………………………………...8**

5.1 Performance Requirements

5.2 Software Quality Attributes

**6. Other Requirements ...………………………………………………………………….…9**

**Appendix A: Glossary**

**Appendix B: Validation Test Plan**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Jim Reilly | 4/3/2018 | Created the formatted document | 0.1 |
| Colleen Pawlicki | 4/8/2018 | Completed the Introduction | 0.2 |
| Colleen Pawlicki | 4/9/2018 | Completed the Overall Description | 0.3 |
| Jim Reilly | 4/9/2018 | Completed Validation Test plan | 0.4 |
| Lauren Ernst | 4/9/2018 | Completed External Interface Descriptions | 0.5 |
| Lauren Ernst | 4/9/2018 | Completed Appendices | 0.6 |
| Jim Reilly | 4/10/2018 | Completed Non-Functional Requirements and System Features | 0.7 |
| Lauren Ernst | 4/10/2018 | Produced and reviewed final document | 0.8 |

# **Introduction**

## **Purpose**

The purpose of this document is to provide detailed specifications for the Pupil Detection software package. Specifically, this document details the requirements and specifications for the graphical user interface (GUI). This document details the original software requirements. Additionally, this document introduces the motivation behind the creation of the product and defines the problem statement. The document will emphasize the user experience and how the functionality of the product directly impacts the user, as the document focuses on the GUI subsystem.

## **Document Conventions**

The headings and sub-headings, e.g. “1. Introduction” and “1.1 Purpose”, will be in bold to facilitate readability and organization. Any acronyms introduced in the document will first be written in full text with the acronym in parentheses adjacent to the phrase. Later usages of the phrase will be referenced by the acronym. Please consult Appendix A for definitions of commonly-used terms and abbreviations.

## **Intended Audience and Reading Suggestions**

This document is intended for the other developers working on the product, i.e. the developers involved with other subsystems. The document will be thorough enough such that individuals concerned with the overall development with the product and less concerned with the technical details will also find value. The remaining sections of the document contain details about the overall functionality of the product, the interfacing between the GUI, the hardware subsystem, and the software subsystem, the overall system features, and other nonfunctional requirements. The document contains four appendices: Appendix A is a glossary of commonly used terms, Appendix B contains models to better describe interactions between subsystems, Appendix C contains items that require closure, and Appendix D is a test validation plan for the GUI subsystem. If the reader needs to quickly find a particular section, reference the Table of Contents for the corresponding page numbers.

## **Product Scope**

The product scope pertains to the GUI subsystem of the Pupil Detection software package. The GUI needs to solicit the input from the user, call the pupil detection algorithm developed in a separate subsystem and pass the inputs to it, receive the output from the algorithm and present the results to the user in a graphical way. The aim of the GUI is to develop an interface that makes it easy for the user to understand the results of the algorithm and evaluate the success of the software package. Due to dependencies between the GUI subsystem and the other subsystems, the document will reference the other subsystems when applicable. The detection algorithm will not be described in detail, but, rather, treated as a “black box” component, as only the inputs and outputs of the algorithm are required to detail the specifications of the GUI subsystem.

## **References**

This document references several documents relating to the product requirements, including:

[1] BIEN 4290 Biocomputer Engineering Design Laboratory II: Final Project: Deliverable #1

[2] BIEN 4290 Biocomputer Engineering Design Laboratory II: GUI Team

This document was created using the IEEE Software Requirements Specifications template:

[3] IEEE SRS. Wiegers, Karl E. 1999.

[4] Matlab GUI Guide Documentation: https://www.mathworks.com/help/matlab/guide-or-matlab-functions.html

# **Overall Description**

## **Product Perspective**

The specifications described in this document are for the GUI subsystem alone. The GUI is a component of the Pupil Detection system. The other subsystems of the Pupil Detection system are being developed concurrent to the GUI. The goal of the overall system is to detect the pupil in images showing eye tracking movement. The algorithm correctly identifies circles within images input by the user. In short, the purpose of the GUI is to provide the correct input files to the main program and display the pupil outline on the input image after the correct location in the image has been determined. Figure 1 below shows the major components of the overall system and the interactions between each of the components. The GUI subsystem only directly interacts with the main program.

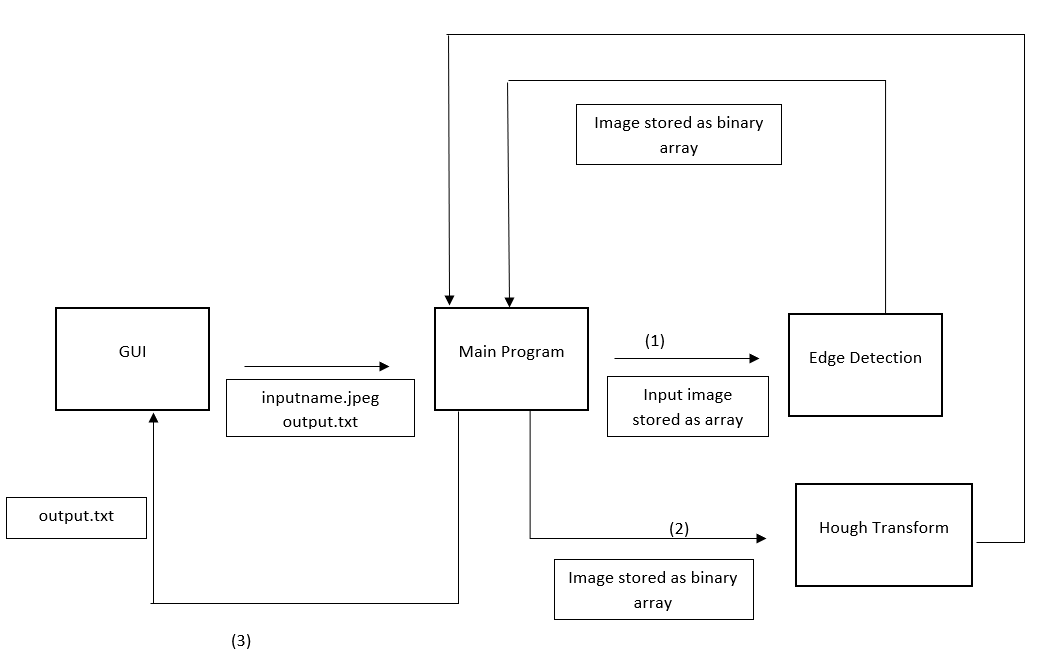


Figure 1: High level data flow diagram with overview of system components

## **Product Functions**

The GUI needs to obtain two inputs from the user: the name of the JPEG image to be processed and the name of the output text file to which the pupil center coordinates and radius values are written. The GUI will contain a button that, when pressed, calls the detection algorithm housed in the main program and passes the two inputs to it. The GUI will include error handling that keeps the button disabled and prevents execution of the main program if the files do not exist. The button will change color to green when all of the error checks have passed. The GUI will read the contents of the text file populated by the algorithm and draw the outline of the pupil. The pupil outline will be superimposed on the input image and displayed to the user next to the original image. More detailed specifications are below. Figure 2 below shows the interaction between the GUI and the main program.

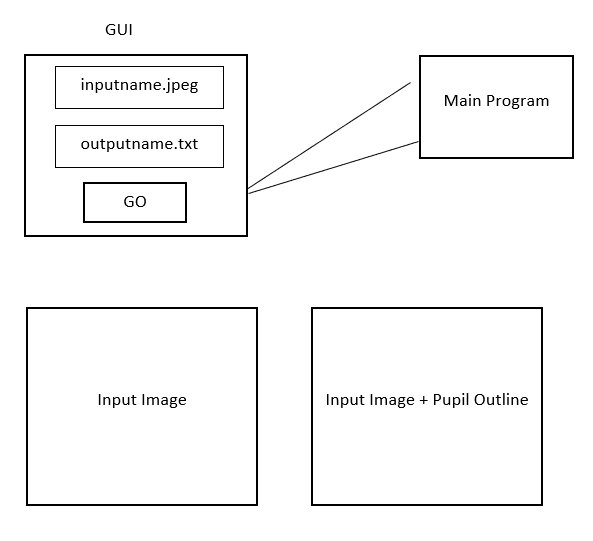


Figure 2: Graphical display of connection between GUI and main program

## **User Classes and Characteristics**

The overall system provides the user with the original image superimposed with the outline of the pupil. The user needs to know the file name of the JPEG image that will be processed, as well as the name of the output text file that will be populated with the results of the algorithm. The system will be most effective when the user is present, as he or she needs to press the button to execute the algorithm. As such, the system will be most useful when processing a few images at a time, rather than processing several images in bulk.

## **Operating Environment**

The GUI will be implemented with a software platform that is both compatible with the main program and facilitates the drawing of a circle given the center coordinates and radius. The selected platform will also require the use of an image viewer to display the image with the superimposed pupil outline.

## **Design and Implementation Constraints**

The primary constraint to the implementation of the GUI is compatibility with the main program. The teams responsible for the main program and GUI subsystems need to maintain strong communication during planning and development to ensure that the algorithm can be executed properly through the GUI. The language used to implement the GUI does not need to be the same as that used to develop the other subsystems, but the algorithm needs to be executed directly through the GUI.

# **External Interface Requirements**

## **User Interfaces**

A user of this application should be able to run the executable to open the application. See Figure 3 below for the main screen of the user interface. There will be no formal layout standard used, for this will simply depend on the preference of the software developers. The interface will include two input fields for the user to enter the input and output filenames. If both files are valid, the “GO” button will appear green on the screen, indicating that all checks are met for execution. Beneath these input fields, there will be two areas to display images. The left-most box will display the input image, and the right-most box will display the output image with the overlapping circle. At the bottom of the user interface, there will be a “Reset” button. This will be used to clear the filename input fields for new files.

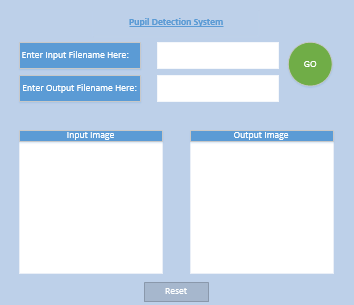


Figure 3: Graphical display of main user interface

The GUI standard buttons that will be used in creating this user interface come from the MATLAB GUIDE documentation [4]. Through the implementation of this functionality, error checking will be handled within this code. The defined window size of this GUI will depend on the size of the user computer screen, and this will be calculated using ratios of pixels per screen dimension. There will be no implementation of complex keyboard shortcuts. There will also be no implementation of multiple user interfaces. Only one will be created, implemented, and executed, to interact with the user.

## **Hardware Interfaces**

This system is simply a software application, running on a computer. In order to call this application from the main program, an executable must be created in the end. The executable must be able to run on a Linux or Windows operating system. No other hardware devices or specific communication ports are required for this system.

## **Software Interfaces**

The application will take in user input of two filenames. The software will check that this data exists and pass it to the main program. Therefore, the image data will be shared to the rest of the system. The files should original be stored somewhere on the local hard drive of the computer. Once the software is complete, the GUI system will receive a flag indicating that the image processing and filtering is complete. This flag will allow the GUI to then display the original image and draw the final image to the user interface.

## **Communications Interfaces**

The only communication required for this application to run is between the main program and the GUI. There are no specific network communication protocols required for this application to run on a computer.

# **System Features**

## **File Input**

4.1.1 Description

Two string fields for providing the input file names to the main program. Next to the string boxes

are two open file dialog buttons for interacting with the file system.

4.1.2 Stimulus/Response Sequences

An open file dialog limiting file type to only .jpeg will prevent improper file inputs. An open file

dialog can also be used for the output file but the user should also be able to write a new file that

does not yet exist.

4.1.3 Functional Requirements

REQ-1: Non-valid file types do not enable the “Start” button

REQ-2: Open file dialog limits filetype selection to only valid types

## **Start Button**

4.2.1 Description

A button to start the main program

4.2.2 Stimulus/Response Sequences

A single push button that is only enabled when valid input file types are provided

4.2.3 Functional Requirements

REQ-1: Non-valid file types do not enable the “Start” button

REQ-2: Valid if proper input files are provided

## **Image Display**

4.3.1 Description

The original image is populated into a plot field and a circle is overlaid

4.3.2 Stimulus/Response Sequences

The original image is loaded after it the start button is pressed and a circle is drawn over the eye

when the main exits with a 0.

4.3.3 Functional Requirements

REQ-1: No image is displayed when the application opens

REQ-2: The original image is loaded when the main subsystem is started

REQ-3: After completion of the main method, the circle is drawn on top the image

## **Coordinates View**

4.4.1 Description

The X/Y coordinates and radius are printed to the GUI

4.4.2 Stimulus/Response Sequences

After main method completes, coordinates to draw circle are written to screen

4.4.3 Functional Requirements

REQ-1: No coordinates are displayed when app opens

REQ-2: Coordinates read from output text file are written to screen

# **Other Nonfunctional Requirements**

## **Performance Requirements**

The GUI’s performance limitation is the speed of the main algorithm itself. The GUI should add limited additional overhead by minimizing the amount of pre and post processing it performs.

## **Software Quality Attributes**

The software building the GUI requires certain design characteristics that would make its implementation more desirable. The use of callback functions attached to form elements of the GUI will limit the amount of possible ways that users can interact with the GUI, thus preventing many possible user failures. By being developed with GUI focused IDE such as MATLAB’s GUIDE or Microsoft’s Windows Form Application, a maintainable final product depending on well tested proprietary code will be produced. The GUI needs to be interoperable with a main program developed by a different development team, so if it is assumed that the form of the main program will be a compiled executable instead of source code, the app will be functional regardless of algorithm implementation decisions assuming a well-tested main subsystem.

# **Other Requirements**

**Appendix A: Glossary**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| GUI | Graphical User Interface; the part of the system that gathers user input, passes the inputs to the algorithm, and presents the output to the user |
| GUIDE | Matlab GUI and app designing tool |
| IDE | Integrated Development Environment; a software application that provides comprehensive facilities for further software development |
| JPEG | Digital format used for compressing image files |
| Pupil Detection System | Used to refer to the overall software package in development |

**Appendix B: Validation Test Plan**

In order to validate GUI design without the existence of the other subsystems, several unit tests have been designed to verify the appropriate behavior following the various possible paths the application could take.  
  
*Test #1: Invalid input file type*

If provided an image file that is not a .jpeg or an output file that is not a .txt, the application should not permit that the user call the main program subsystem at all until input is corrected.

This can be tested by validating that the button is disabled unless both file inputs have a regex passing input file, or enabled when two valid test files are provided.

*Test #2: Main subsystem returns non-success exit status*

The GUI subsystem should only aim to overplot a circle if the main subsystem executed successfully. If the main system returns an exit status that is not 0 it will be interpreted as an error and a dialog will inform the user that there is an issue with the given files.

This can be unit tested by asserting that the drawing function was not called and that an error catching block of code was executed if a test main program that only returns -1 is used.

*Test #3: Main Subsystem returns a success exit status*

If the main subsystem returns a 0 exit status, it is assumed that the calculated output file correctly identified the circle to draw. The output file will be read and a circle should be drawn.

This can be unit tested by providing an output file that already has coordinates provided and using a test main method that only returns 0.