Bilkent University

GE401

Software Module and Task Specification Report I

eyeCue

Assistant Eye

Team 9

Behiç Buğra Bacanlı

Mevlüt Türker Garip

Burak Işık

Yunus Burak Suçsuz

Melik Koray Üster

Çağlar Varan

Date

16.12.2011

Introduction

This document aims to provide a detailed description of the structure of the overall system and its components. The eye-detection system is designed for the wheelchair used by completely paralyzed people. The name of the product produced by our company EyeCue will be assistant eye. The content of this document will partly be constructed on the previous product specification and requirements document. In this document, the software description will start with a system overview supported by software architecture diagram which depicts the input output interaction between subcomponents of the system. Afterwards, the software modules in these components will be explained with their characteristics and constraints. Finally, the hardware interaction of the software and the environment requirement for the system to run will be shown according to the specifications of the system components.

System Overview

Image analysis is one of the fields in programming where the accuracy is the top priority. Therefore, the eye-detection software is designed to compensate the changing environmental conditions that may affect the image characteristics. These conditions such as change in the sun light, variety of colors or absentness of enough light are making an accurate software very challenging.

The eye-detection software consists of 5 modules such as Image Capturer, Image Grayscale Converter, Image Area Partitioner, Shape Pattern Detector and Motion Detector. These modules are responsible for different tasks through the eye-detection. The images will be taken to the system by Image Capturer from the web cam with the frequency of 10 images per second and will be buffered to arrays for the software to use them. Each two consequent images will be processed by the Image Grayscale Converter and transformed into grayscale with the threshold value set in order to optimize software accuracy. This transformation will compensate the variety in the image colors. Afterwards, Image Area Partitioner will find the connected components in the image with the special algorithms. Among these connected components, Shape Pattern Detector will calculate their areas and find the component nearest to the circle. The connected component that has the biggest circular area in the image is very likely to be the pupil. Motion Detector will locate this component and examine the changes in the coordinates of its center. After identifying the direction of the movement, it will send the corresponding signal to the Engine Driver, which will control the engines accordingly. These interactions are shown in Figure 1 and a flowchart is depicted in Figure 2.

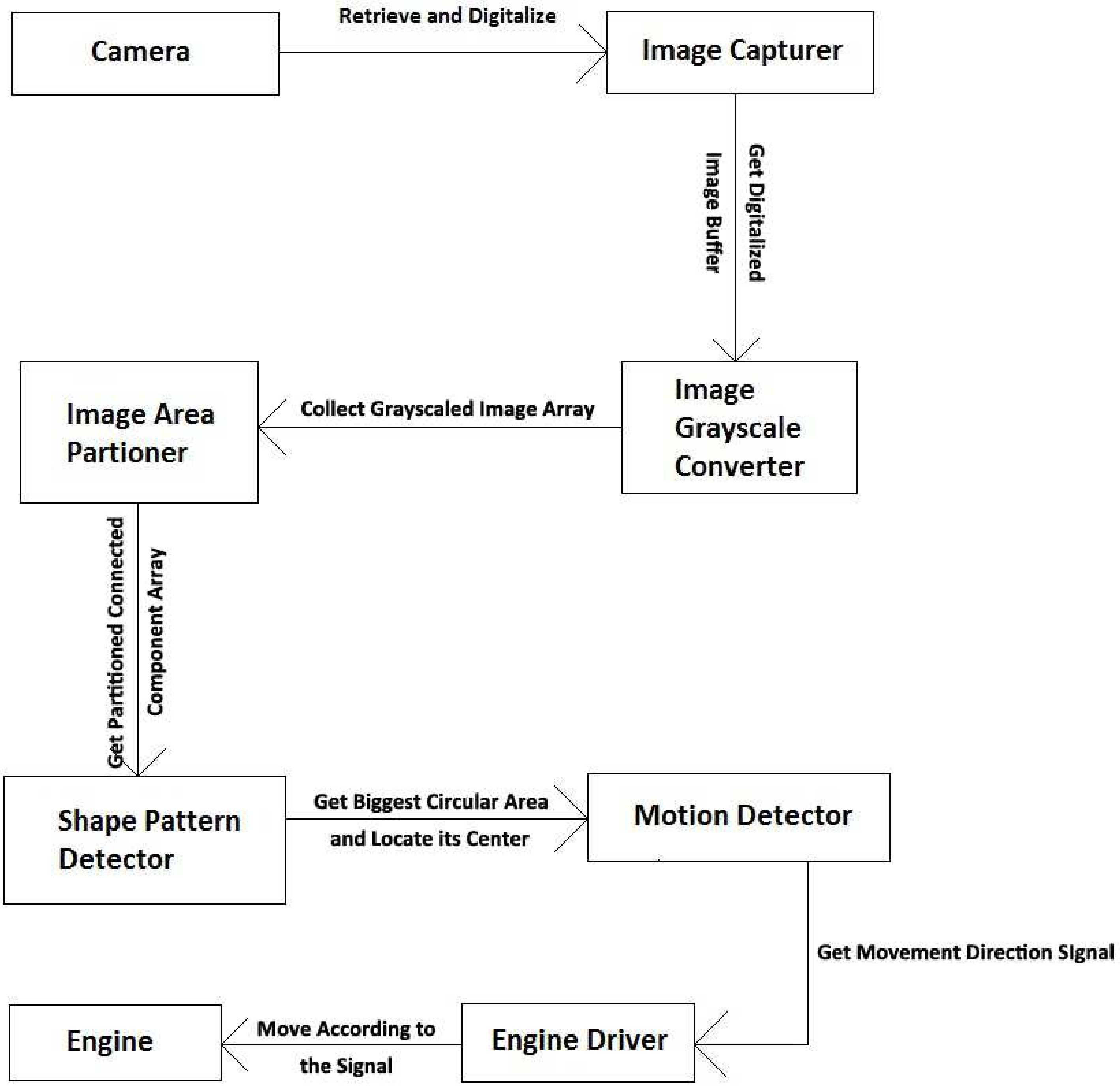
Camera
Retrieve and Digitalize
Image Capturer
Get Digitalized Image Buffer
Image Grayscale Converter
Collect Grayscaled Image Array
Image Area Partioner
Get Partitioned Connected Component Array
Shape Pattern Detector
Get Biggest Circular Area and Locate its Center
Motion Detector
Get Movement Direction SIgnal
Engine Driver
Move According to the Signal
Engine

Figure 1 - Software Architecture Diagram

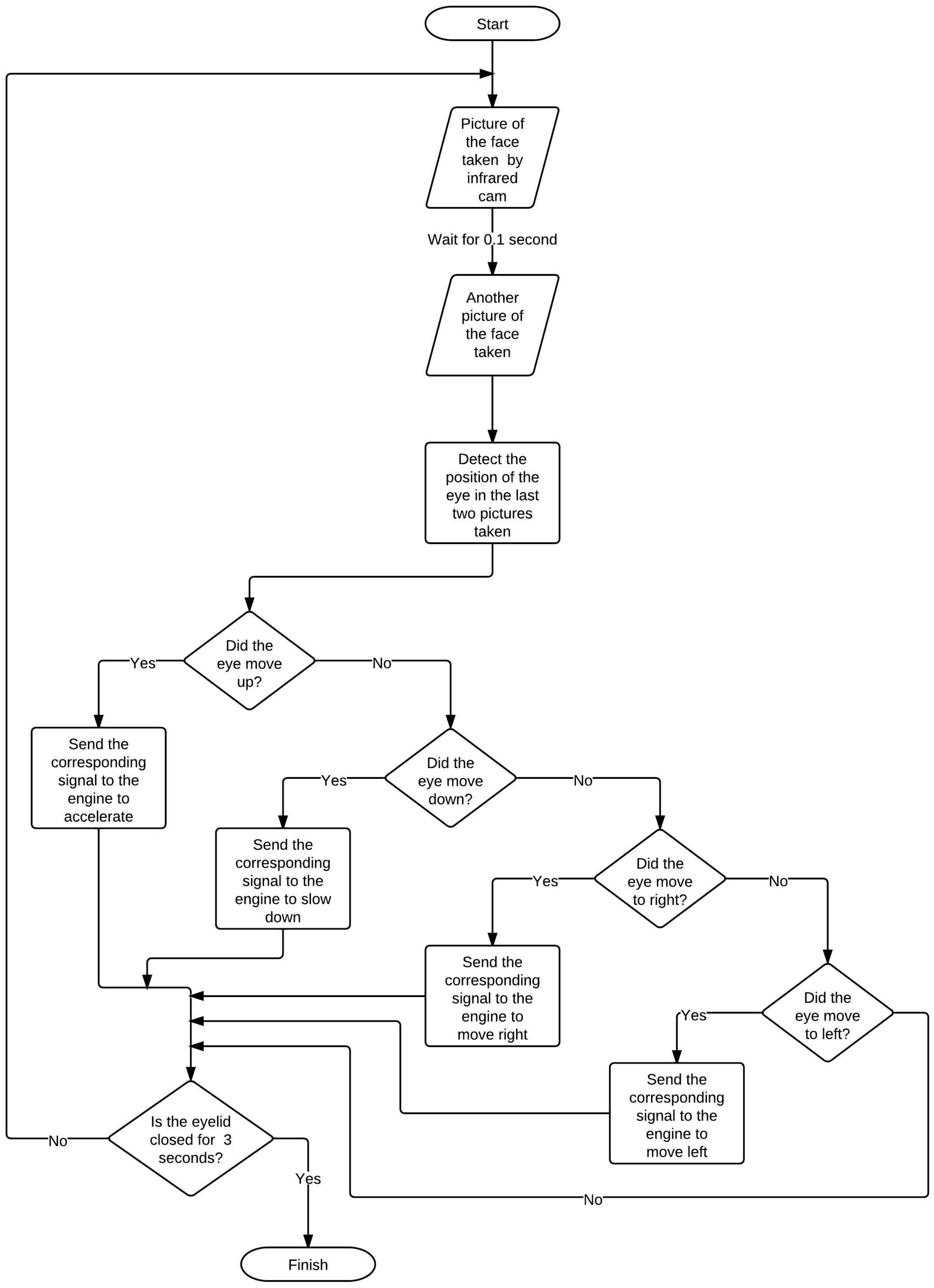
Start
Picture of the face taken by infrared cam
Wait for 0.1 second
Another picture of the face taken
Detect the position of the eye in the last two pictures taken
Did the eye move up?
Yes
No
Send the corresponding signal to the engine to accelerate
Did the eye move down?
Yes
No
Send the corresponding signal to the engine to slow down
Did the eye move to right?
Yes
No
Send the corresponding signal to the engine to move right
Did the eye move to left?
Yes
No
Send the corresponding signal to the engine to move left
Is the eyelid closed for 3 seconds?
No
Yes
Finish

Figure 2 - Flowchart of the Software

Software Modules

Image Capturer (Module)

Definition: According to the product specification document, the detection accuracy of the eye gestures and time efficiency of the tasks are the top priorities of the software. Therefore, this module that captures the images from the camera is specialized to assure the frequency 10 images per second.

Responsibilities: This module is responsible for providing the only input interface with the hardware. It collects and buffers each two consequent images with the frequency 10 images per second. Afterwards, these buffers are passed to the Image Grayscale Converter to be transformed into a format that eye-detection software would work efficiently.

Constraints: Since this module collects data with a constant frequency from the camera, it must be assumed that there will be no delays in the camera. The light in the environment is assumed to be enough for differentiating the shapes in the image. The other important constraint is that the file sizes of the 2 last images taken must be small enough to fit into the memory buffers. Images should not be corrupted when digitalized and buffered.

Uses/Interactions: This module interacts with the camera driver to take images as inputs and buffers of those images are used by the Image Grayscale Converter module. The interaction between this module and Image Capturer is the most crucial step before other tasks to be performed because any mistake done or any corruption of the image data taken would cause all other modules to give wrong outputs.

Resources: This module desperately depends on the memory space allocated. The number of images taken in a period of time is sometimes above the number of images that can be processed by the software at that time. Therefore, these pending images must also be buffered for the future use in order not to lose any detail about the eye position.

Interface/Exports: The module gives an error for any of the constraints that is not satisfied such as response delay of the camera, lack of light and lack of enough memory space for the image buffers. In case of any of these exceptions except the light, program will terminate immediately. In the lack of light, the system will wait until the right condition is satisfied. This component provides digitalized and buffered image arrays to the system.

Image Grayscale Converter (Module)

Definition: The variety of colors in the image is sometimes so much that it makes an accurate eye-detection mechanism almost impossible. Therefore, this module is the converter module that eliminates this kind of situations by converting the images to grayscale.

Responsibilities: Its main role is to work on an image buffer and convert that image to grayscale. The threshold to be used in this transformation is set at the beginning of the program to optimize the accuracy and easiness in differentiating the shapes in the image by other modules. It process the image one pixel at a time and convert it with the threshold.

Constraints: Main assumption in this module is that the parameter image buffer is given uncorrupted. The parameter buffer must also be given in correct format according to how this module works. Threshold must be in a logical range. The processing time should be no longer than 0.3 seconds since successor procedures will require more time to complete their tasks.

Uses/Interactions: This module takes its inputs as a image buffer from Image Capturer with a constant frequency. After processing the image, it gives the grayscale version to the Image Area Partitioner, which is the main part of the software logic, as output. Therefore, the success of this module in using the right threshold for converting the image plays a very important role in the accuracy of the software. If the grayscale version is not produced properly, Image Area Partitioner will not differentiate the different areas in the image.

Resources: This module must be reasonably fast so requires sufficient processing unit. If a microprocessor is used, the circuit must be optimized for matrix access and arithmetic. Since the efficiency in this part is really important, any space sacrifices can be done to boost speed.

Interface/Exports: The module will give an error if the image buffer input parameter is corrupted and the program will terminate. If the threshold happens to be out of a reasonable region, it will be set again to the most logical value again. This module provides grayscale version of the input image to the system to be used efficiently for easier pattern recognition.

Image Area Partitioner (Module)

Definition: It is the most important part of the software. It is the module that processes the grayscale image and partitions different connected components. In more understandable terms, it is the process of labeling different shapes in the image.

Responsibilities: This module is responsible for analyzing the image and differentiating the shapes according to the their colors and areas. It enables examining the image further with its different shapes. It gives an array of connected components that the client can reach by the label numbers.

Constraints: It does not have a parameter constraint other than getting an uncorrupted grayscale image buffer. However, there are more important constraints like time and space. Since the information of the connected components are stored in an array to be passed as an output, the memory space must be enough to keep this connected component array. The performance is the most important constraint because this module uses a complicated recursive approach to identify different connected components. Since it will be the module that consumes most of the execution time, any improvement will contribute significantly to the overall performance.

Uses/Interactions: This module gets its input from Image Grayscale Converter. It does not really check the quality of the parameter but its success pretty much depends on the quality of the grayscale buffer image. After it computes the connected component array, the output will be given to the Shape Pattern Detector. Since it is the main part of the software logic, the total accuracy rely on how much this modules succeeds in its task.

Resources: This module consumes the memory and the processing power more than others because it uses a complicated recursive approach which may fill out the program stack very easily. Since the total time required for the software is mostly specified according to this module, a good processor will contribute a lot to the efficiency in this stage of the program.

Interface/Exports: In case of getting corrupted parameter or overflowing the stack because of the recursion, the module will throw an exception and terminates the program. This modules provides an array as output that contains the information of partitioned components.

Shape Pattern Detector (Module)

Definition: It is the module that processes the connected component array and finds most circular shape that has the largest area. This returned shape will more likely to be the pupil to be tracked by its movements.

Responsibilities: This module is the last step of the image analysis part of the software. It is responsible for finding the most circular area that has the maximum value. In order to achieve this, it iterates through the connected components and finds their areas by pixel counting. Then, it computes what the ideal circular area should be with the radius of each connected component. It returns the information of the connected component whose area is the largest and closest to the ideal circular area.

Constraints: The connected component array must contain at least one connected component. The input array should not be corrupted. The structure of the connected component array must be in proper format that the shape detection algorithm would work. The algorithm should be able to detect at least one circular area with a low mistake possibility.

Uses/Interaction: This module gets its input from Image Area Partitioner as a connected component array. After finding a desired connected component, it gives the label number of that component to the Motion Detector to find its center and detect any changes with consequent images.

Resources: The module requires a processing unit that is optimized for matrix accesses and arithmetic. It does not use considerable space therefore improvements in performance by sacrificing from the space may be logical.

Interface/Exports: An exception will be given in case there is no connected component in the input array or it is corrupted. This module provides the information about the connected component which is very likely to be the pupil. This information is simply the label number.

Motion Detector (Module)

Definition: It is module that finds the center location of the connected component, whose label number is given as an input, and stores it to compare with the one in next image. In other words, it detects the direction of the motion of the eye.

Responsibilities: As described above, its responsibility is to find the direction of the eye movement and signal the engine driver for corresponding wheelchair motion.

Constraints: The label number passed as a parameter must be a valid number, belonging to a actual connected component. The location of the center found must be in the range image dimension. There must be an initial calibration value to compare with the first image results because the algorithm always uses the past values.

Uses/Interaction: The success of the program does not really depend on this module since it does very straightforward calculations. However, the initial calibration value must be selected carefully. The module gets its input from Shape Pattern Detector as an label number of the connected component and the component array. The returned result will be passed to the engine driver to generate a corresponding signal to the engine.

Resources: It uses the required library to communicate with the engine driver. The module does not really require much processing power but requires a fair amount of space to store some of the past calculated locations of the eye.

Interface/Exports: The calculated directions will be sent to the Engine Driver and the corresponding signals will be generated in the driver to move the engine. The module throws an exception if the location or label number are not in the logical ranges.

Hardware Interface

The main input to the system will be taken from the camera which is capable of taking at least 10 images per second. This image will be stored in the memory to be used in the processing unit. In the software running in the processing unit, Image Capturer will gather the data from it with that constant frequency and pass it to the system. Image Grayscale Converter will digitalize and convert the image to grayscale. Image Area Partition will divide the image into different connected components according to their shapes and colors. Shape Pattern Detector will filter the connected component array and return the information of biggest circular area. Motion Detector will calculate the location of the center of that connected component and compare it with the past calculated location. The corresponding message will be given as output to the Engine Driver and it will translate the message to the signal that engine can understand. This motion signal will be given as output from the processing unit to the engines. The engine will move according to this signal.

As I/O requirements, the camera must produce digital output for the image taken and it should be little enough to fit into memory allocated for image buffers. Processing unit must be optimized for matrix accesses and arithmetic. The memory must be large enough to contain all required image buffers and recursive function calls.

Software System Requirements

The system must work in almost real time to prevent possible accidents. The maximum total allowable time is 1.5 seconds. Therefore. the time requirements for each module must be like below:

Image Capturer (0.1 second), Image Grayscale Converter (0.3 second), Image Area Partitioner (0.5 second), Shape Pattern Detector (0.4 second), Motion Detector (0.1 second), sending signal to the engine (0.1 second).

The memory must be large enough to contain all required image buffers and recursive function calls. For each image whose dimension is 500 X 500 px, a double array with the size 500 X 500 must be stored. Since there will be 10 images taken per second, the memory must be able to handle and allocate almost 10 MB data per second. The memory access must be optimized for efficient matrix accesses.

Processing unit must also be optimized for matrix arithmetic because most of the software modules do very intensive and complex matrix operations. Achieving all these tasks in almost real time may require a processing unit of a laptop. These processing power needs will be more accurate when the testing and optimization stage of the software development starts.

References

[1] http://www.mathworks.com/help/toolbox/images/exampleindex.html

[2] http://www.mathworks.com/image-video-processing/?s_cid=global_nav

[3] http://www.slideshare.net/dansaffer/making-good-design-decisions