*Sheffield Hallam University*

# Automatically locating the pupil centres in a full facial image

George Colin Wragg

M.Sc. in Computer and Network Engineering
2005-06

Supervisor: Dr. B Amavasai

This report is submitted in partial fulfilment
of the requirements for the degree of
Masters of Science
in Computer and Network Engineering

# Acknowledgements


- I would like to thank Dr B Amavasai for his supervision and comments throughout this thesis.

- Additionally my warm thanks go to Mr R Singleton of Synectic Systems Ltd for ensuring that I had time available for the course whilst still being employed.

- Also to Anand Veeraswamy for ideas on document presentation.

# Abstract

The aim of the thesis is to provide an automated software solution that can determine the centre of the pupils in a full facial infrared video image such that future eye tracking information can be achieved.

This thesis derives a novel approach to enable the centres to be quickly found so eye tracking can be made in real time and applied to the source image for easy verification.

# Table of contents

# List of Figures

# List of Tables

# Chapter 1
# Introduction

## *1.1 Introduction*

The aim of this thesis is to investigate and produce a software solution that will locate the pupil centre in eyes taken from a video image. The areas that will be addressed are:-

- Investigate existing solutions to see if they can be implemented and if not design an alternative method
- Investigate best user environment to enable accurate location
- Analyse typical images to determine best approach
- Design and implement the software
- Test the software against user determined points both with and without the user wearing glasses
- Implement basic tracking for a camcorder .dv format video file.

## *1.2 Motivation*

There is a need in medical circles to be able to video a patient whilst performing tests on them relating to eye movement.

At the moment this tends to be performed visually and is open to interpretation. Using software it should be able to automate this and as the software is deterministically programmed then the results should always be the same.

## *1.3 Deliverables*

The main deliverable from the thesis will be a standalone C++ class that will take a correctly formatted image and find the eye centres in it. The class will be developed for ease of use and speed of detection enabling it to be used in real time applications.

## *1.4 Thesis environment*

This thesis is mainly software driven and as such the following are specific to the author's usage.

### 1.4.1 Test equipment used

The thesis programming was performed using an IBM R51 laptop running at 1.5GHz with 1GB of RAM. The operating system was Ubuntu Linux with the imaging software being powered by Mimas [1] and the user interaction using ncurses [2].

### 1.4.2 Safety issues

As with all things there are certain safety elements that must be observed.

- The correct placement of laptop and posture is paramount and the guidelines on the sales office practice website [3] were followed.
- It is recommended to take a 10 minute break at regular intervals to prevent cramp and ensure that the eyes are not continually viewing objects only a few feet away.

## *1.5 Report structure*

Chapter 2 discusses current work in the area of pupil location and develops the theory that is used in the novel solution presented.

Chapter 3 is concerned with extracting the information from the end users supplied data and developing the algorithm used to verify that the data actually belongs to an eye rather than a random collection of data.

Chapter 4 builds upon the previous chapter by implementing higher level functions that automatically find the eyes and the centre of the pupils. These are further enhanced to take in real world situations of users both with and without glasses. Software is then written to extract eye motion vectors from a camcorder recodrding.

Chapter 5 concludes the thesis and outlines further research that could be carried out.

## *1.6 Summary*

This chapter highlights the need for the thesis and how the following chapters are laid out. It also identifies the equipment setup and safety issues that must be taken into account.

# Chapter 2
# Relevant theory and analysis

## 2.1 Introduction

This chapter looks at the basic theory behind locating the pupil centres in an image in papers that have already been published. These methods are checked to see if they can be applied to our scenario.

The end users environment is then discussed and the proposed method of location of eye centres and the theory behind it is explained.

## 2.2 End user supplied data set

The end user wishes to use a camcorder to record a patient's eye movement. Whilst the .dv format could be used it is not readily applicable to repeatable data when testing.

A program was written[1] to take the supplied file and split it into individual gif format images (as the images are to be worked on as 256 bit grey scale). The images were consecutively numbered and split into separate folders to produce images as typified in figures 1 to 5.

[1] Appendix 2

**Figure 1 - Front image of face with IR illumination**

**Figure 2 - Side image of face with IR illumination**

**Figure 3 - Front image of face with normal illumination**

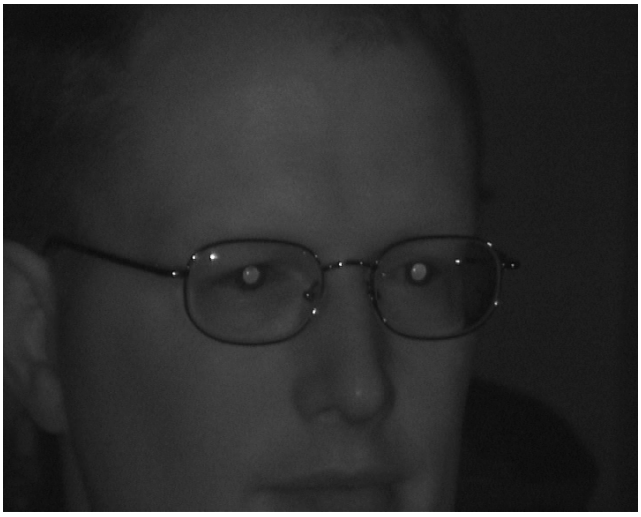


**Figure 4 - Side image of face with IR illumination and subject wearing glasses**

**Figure 5 - Side image of face with IR illumination and subject with eye patch and glasses**

## 2.3 Literature review

Rather than reinventing the wheel, the internet was searched to find what methods are currently available to see if they could be applied to our scenario.

- Thresholding to find the pupil which has the largest dark mass [4, 5]
- Feature extraction using deformable templates [6, 7, 8]
- Region based template deformation [9]
- Searching over all circles for a circle that gives the most abrupt change in pixel value [10]
- Edge detection and Hough transform for circles [11, 12]
- Active contours and discrete active contours to locate circles [13]

- Morphological edge detection and wavelet phase features [14]

The existing current models that were reviewed were rejected due to combinations of

- Complex algorithms meant that speed was sacrificed and as it is a requirement to work on live images speed is sacrosanct.
- A priori knowledge was required and in this application it was deemed inappropriate.
- The algorithms were optimised for iris recognition and expect a high detail image of a full eye. The data set for this application has an eye element that is only approximately 50 pixels wide (7% of the screen width).

## 2.4 Chosen environment

As a requirement was speed of detection it was decided that the best environment would be the infrared images as these provide a "hot spot" caused by the retinal reflection that can be easily found by looking at pixel values. This position can then be used to seed an algorithm that tests whether the pixels do indeed form an eye based on that start point.

The environment is also to be set such that the whole face is framed with the subject looking into the camera and the horizontal extremes being the end of the subject's ears as typified in figure 1. This will provide the largest eye image to work with and place the retina reflection away from the side of the pupil with whilst still preserving the subject's nose and other features[2].

## 2.5 Proposed method

As outlined above, there are no ready algorithms that can be utilised and this thesis presents a method based upon first principles.

[2] as need nose, or other fixed feature, for detecting if face moved rather than eyes

The eye can, in general, be considered to be circular (especially in the x direction and up to the pupil/iris boundary) as can be seen in figure 6 [15]. It is immediately obvious to a human viewer where the boundaries and centres are.

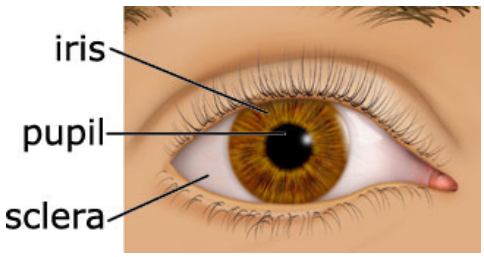



**Figure 6 - Basic eye structure**

It is proposed that slices through the potential eye are taken in the x direction and used as a basis for calculating the x centre position as shown in figure 7.

Notice that the start position $x_p$ has not been placed central as if this information was known before hand there would not have been a need to find it.

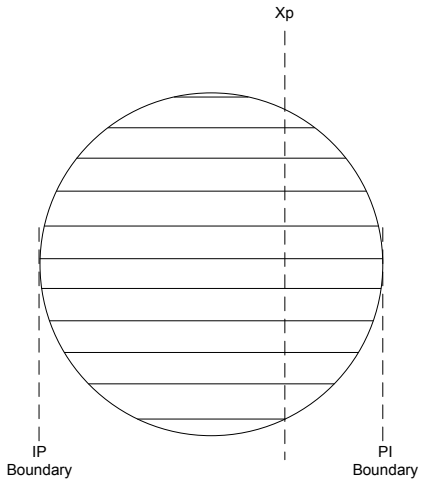



**Figure 7 - Typical sliced data with offset start position**

If each slice is treated by itself then the length of the pupil at that point is given by the addition of the iris/pupil and pupil/iris boundaries (IP + PI). The halfway point of that should be where the centre would be located (IP+PI)/2 from the IP boundary. What is required is the offset to the centre from the initial position (which is treated as offset zero) and with reference to figure 8.

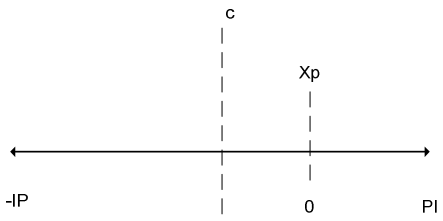



**Figure 8 - Obtaining the centre point offset from start point**

$$C = -IP + \frac{IP + PI}{2} = \frac{-2IP + IP + PI}{2} = \frac{PI - IP}{2}$$

If we repeat this for all the slices that have been taken

$$C = \frac{\sum_{all\ slices} \frac{PI - IP}{2}}{number\ of\ slices} = \frac{\sum_{all\ slices} PI - \sum_{all\ slices} IP}{2 \times number\ of\ slices}$$

The centre is then simply given by Xp + C.

Using the data, taken directly from the Visio drawing package, shown in table 1 and the formula above then the centre can be calculated as in table 2.

| Slice | | Iris/Pupil (LHS) | Pupil/Iris (RHS) |
|---|---|---|---|
| 1 | | n/a | n/a |
| 2 | | 54.513 | 9.153 |
| 3 | | 63.950 | 18.695 |
| 4 | | 69.942 | 24.942 |
| 5 | | 73.673 | 28.673 |
| 6 | | 74.970 | 29.970 |
| 7 | | 74.481 | 29.481 |
| 8 | | 72.090 | 27.090 |
| 9 | | 67.555 | 22.555 |
| 10 | | 60.141 | 15.141 |
| 11 | | n/a | n/a |

**Table 1 - Pupil/Iris Boundaries for eye with offset start position**

| | | Full Values | | | | Pixel Values | | |
|---|---|---|---|---|---|---|---|---|
| | | IP = | PI = | | | IP = | PI = | |
| Slice | | Iris/ Pupil | Pupil/ Iris | (PI-IP) /2 | | Iris/ Pupil | Pupil/ Iris | (PI-IP) /2 |
| 1 | | n/a | n/a | | | n/a | n/a | |
| 2 | | 54.513 | 9.153 | -22.680 | | 55 | 9 | -23.000 |
| 3 | | 63.950 | 18.695 | -22.628 | | 64 | 19 | -22.500 |
| 4 | | 69.942 | 24.942 | -22.500 | | 70 | 25 | -22.500 |
| 5 | | 73.673 | 28.673 | -22.500 | | 74 | 29 | -22.500 |
| 6 | | 74.970 | 29.970 | -22.500 | | 75 | 30 | -22.500 |
| 7 | | 74.481 | 29.481 | -22.500 | | 74 | 29 | -22.500 |
| 8 | | 72.090 | 27.090 | -22.500 | | 72 | 27 | -22.500 |
| 9 | | 67.555 | 22.555 | -22.500 | | 68 | 23 | -22.500 |
| 10 | | 60.141 | 15.141 | -22.500 | | 60 | 15 | -22.500 |
| 11 | | n/a | n/a | | | n/a | n/a | |
| | | | | | | | | |
| sum/slices | | | | -22.5342 | | | | -22.5556 |

**Table 2 - Calculated centre point**

This suggests that the actual centre point is -22.5342 units from the start point. From the Visio package the start point was at approximately 120 and the centre point at 97.434 meaning a shift from the start of 97.434-120 = -22.56 which is well within tolerances (as the values were not measured at 800% magnification, auto snap was used).

Also notice that the value calculated using the whole pixel numbers also provides an offset that is extremely close to the centre even though we have "lost" accuracy in converting to integers (in reality some are higher and some are lower and these errors tend to average out).

If one of the values had been in error by 2 pixels (say noise or incorrect detection) then the effect upon the overall placement would only have been 2/9 = 0.22 pixel incorrect (a human observer could only place at one pixel) and the fact that errors will tend to be random gives great credence to this method.

The location of the pupil y centre follows in a similar fashion, this time slicing the data in the y direction as x is varied around the start point.

## 2.6 Summary

This chapter looked at what was currently available but the methods were rejected as being unsuitable for the end users requirement.

The user data was examined and a user environment created to produce images that would be correctly formatted front facing under infrared illumination.

A novel method of locating the centres was then derived and forms the basis of the software algorithm that this thesis produces.

# Chapter 3
# Initial review of data

## *3.1 Introduction*

The method developed in the previous chapter utilised ideal data and a perfect circle. This chapter analyses the actual data to see if the proposed method is acceptable on real data.

Once the data has been analysed an actual flowchart will be designed to allow the program to be implemented.

## *3.2 The IR illuminated eye in close up*

Figure 9 shows a close up of an eye illuminated by infrared and shows the retinal reflection that will ultimately be used to seed the start point for the eye detector routine.

Starting from the retinal reflection and working out (either left or right, up or down) the pupil/iris border can be given by a sharp change from a light value to a much darker value and the iris/sclera boundary by the change from dark value to a lighter one.

Unfortunately the actual algorithm can not be as simple as this as we have to take into account noise in the image and the fact that the actual data is corrupted by the retinal reflection. To come up with the best routine the underlying data must first be extracted and analysed.

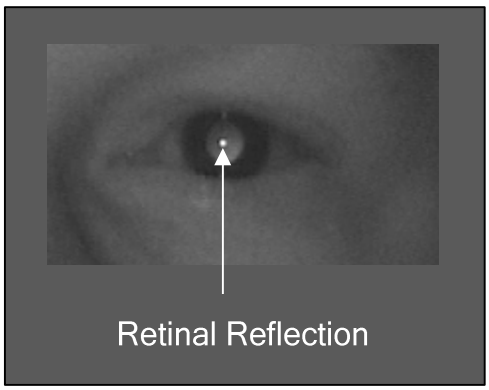



**Figure 9 - IR Image eye close up**

## *3.3 Obtaining eye slices from user input*

The proposed method requires that the iris/pupil to pupil/iris boundary be able to be programmatically determined. To enable this, the input data must first be gathered around the existing eyes in the image for further analysis.

### 3.3.1 The main program shell

As there will initially need to be user interaction in deciding the centres and boundaries a program shell was written[3]. This shell provides the base functions:-

- A call to InitSpecific to allow the users code to initialise any variables they desire.
- TerminateSpecific is called when the program is terminated to allow the user to tidy up any variables they allocated.
- Allow images to be read in from given command line directory using keys 'n' (next image) and 'p' (previous image). Once the image is loaded from disk the main routine will call a user specific supplied routine ImageLoaded to allow it to take further action (such as loading in any associated parameters for that file[4]).
- Once an image is closing (either moving to another or quitting the program with 'q') then ImageClosing is called to allow the user program to save the associated parameters.
- Provide global x,y variables that can be changed and move around the image using the cursor keys . Jump step can be altered using keys '1' (1 pixel), '2' (5

[3] Appendix 3

[4] Typically where the users entered parameters were last. From initial experience there is nothing more frustrating then entering all the eye boundaries for an image and then having to re-enter them next time.

pixels), '3' (10 pixels) and '4' (20 pixels) to enable speedy movement through the image.

- There is also a user callable routine ShowCursorPosition that will show where the current x, y position currently is on screen (as the user may not always want to show it then this was not made automatic).

- Call a user supplied DoUpdate routine when any key is pressed (including any auto actioned keys) to allow the user to process keys.

- When keys pressed/images changed ShowInfo is called to allow the user to show any data they wish to.

### 3.3.2 User determined boundaries

A program was written[5] that used the shell to allow the user to enter information on both eyes. This information included the user decided centres, pupil/iris and iris/sclera boundaries. A typical screen shot showing user interaction is shown in figure 10.

The colouring of the lines (which should be white as grey level 255 used) and border on the right are artefacts of using X-Server and OpenGL combination. As this does not affect the results (saving as a file image would prevent this) and it is what the user will see on screen then it has been left in.

Using the user data the eye was sliced in the x direction (+/-50 pixels) for +/-50 pixels in the y direction from the user centre and saved as a comma separated (csv) file for import into a spreadsheet.

The slicing was repeated in the y direction and also saved as a csv file.

[5] Appendix 4

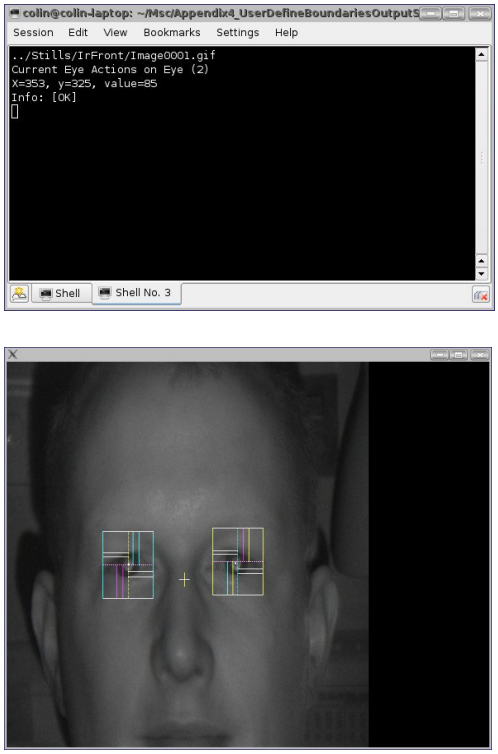



**Figure 10 - Manually entering boundaries**

## 3.4 Analysing the eye slices

The spreadsheet data is shown in tables 3 to 6 which have been condensed to show only the important information around the user defined centre of the eyes. The full data is included on the attached CD.

### 3.4.1 The raw data

From these tables it can be seen where the user thought the centre and boundaries were and a routine must be developed to replicate the user input as much as possible.

| user x | **3[460]** | | | | | | | | | 4[469] | | | | | | | | | | | | 5[481] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| -20 | **73** | 75 | 79 | 81 | 79 | 75 | 78 | 79 | 74 | 74 | 79 | 79 | 79 | 79 | 78 | 76 | 77 | 79 | 80 | 80 | 79 | 81 | 81 | 81 | 85 | 88 | 88 | 88 | 84 | 87 |
| -19 | **79** | 77 | 72 | 66 | 74 | 75 | 74 | 73 | 74 | 73 | 72 | 73 | 74 | 74 | 74 | 74 | 74 | 75 | 78 | 79 | 78 | 81 | 82 | 84 | 84 | 79 | 76 | 78 | 79 | 82 |
| -18 | **67** | 70 | 71 | 71 | 74 | 75 | 73 | 73 | 74 | 73 | 70 | 68 | 73 | 74 | 75 | 75 | 75 | 77 | 78 | 79 | 78 | 76 | 79 | 81 | 78 | 80 | 84 | 81 | 82 | 82 |
| -17 | **60** | 61 | 63 | 63 | 65 | 65 | 63 | 64 | 68 | 71 | 68 | 68 | 71 | 71 | 71 | 73 | 75 | 77 | 75 | 74 | 74 | 77 | 80 | 79 | 74 | 77 | 81 | 82 | 77 | 75 |
| -16 | **58** | 59 | 61 | 61 | 61 | 61 | 60 | 63 | 66 | 68 | 66 | 66 | 70 | 68 | 70 | 72 | 75 | 77 | 75 | 73 | 75 | 81 | 81 | 78 | 79 | 79 | 78 | 78 | 77 | 75 |
| -15 | **50** | 53 | 53 | 53 | 46 | 50 | 53 | 57 | 60 | 59 | 57 | 57 | 61 | 64 | 66 | 67 | 67 | 67 | 70 | 72 | 70 | 75 | 80 | 78 | 73 | 73 | 75 | 75 | 75 | 75 |
| -14 | **49** | 49 | 47 | 44 | 46 | 50 | 51 | 53 | 56 | 54 | 52 | 53 | 58 | 60 | 63 | 64 | 64 | 65 | 67 | 70 | 73 | 73 | 75 | 75 | 73 | 75 | 78 | 75 | 75 | 78 |
| -13 | **47** | 47 | 46 | 45 | 40 | 42 | 42 | 43 | 46 | 49 | 49 | 50 | 52 | 53 | 56 | 58 | 61 | 63 | 63 | 61 | 70 | 71 | 72 | 72 | 70 | 71 | 73 | 77 | 78 | 77 |
| -12 | **51** | 50 | 47 | 46 | 44 | 43 | 40 | 42 | 45 | 46 | 44 | 44 | 49 | 51 | 54 | 58 | 60 | 61 | 60 | 59 | 67 | 67 | 68 | 70 | 70 | 71 | 73 | 75 | 78 | 77 |
| -11 | **60** | 59 | 58 | 58 | 54 | 52 | 49 | 42 | 40 | 44 | 44 | 45 | 44 | 46 | 45 | 45 | 48 | 48 | 46 | 47 | 60 | 59 | 64 | 70 | 67 | 66 | 67 | 64 | 77 | 74 |
| -10 | **62** | 62 | 63 | 64 | 59 | 58 | 54 | 45 | 42 | 44 | 42 | 40 | 47 | 48 | 47 | 45 | 46 | 47 | 48 | 51 | 57 | 58 | 61 | 66 | 67 | 66 | 65 | 65 | 74 | 73 |
| -9 | **69** | 67 | 66 | 65 | 68 | 68 | 66 | 54 | 47 | 47 | 44 | 42 | 40 | 43 | 44 | 43 | 43 | 44 | 46 | 47 | 51 | 54 | 56 | 59 | 65 | 64 | 61 | 64 | 75 | 74 |
| -8 | **69** | 70 | 71 | 72 | 70 | 70 | 68 | 60 | 52 | 47 | 43 | 42 | 47 | 46 | 45 | 44 | 43 | 44 | 46 | 46 | 49 | 52 | 54 | 58 | 64 | 64 | 61 | 66 | 68 | 70 |
| -7 | **73** | 74 | 76 | 77 | 78 | 75 | 77 | 71 | 63 | 53 | 46 | 46 | 44 | 43 | 43 | 44 | 44 | 45 | 43 | 39 | 46 | 49 | 51 | 56 | 60 | 60 | 60 | 63 | 65 | 65 |
| -6 | **80** | 80 | 80 | 79 | 80 | 77 | 78 | 75 | 66 | 53 | 44 | 44 | 47 | 45 | 43 | 44 | 46 | 48 | 48 | 46 | 49 | 49 | 52 | 57 | 59 | 60 | 61 | 63 | 57 | 59 |
| -5 | **79** | 81 | 85 | 86 | 87 | 84 | 85 | 82 | 74 | 60 | 49 | 46 | 44 | 44 | 40 | 39 | 43 | 43 | 40 | 40 | 47 | 47 | 50 | 54 | 56 | 57 | 58 | 58 | 57 | 58 |
| -4 | **81** | 83 | 84 | 84 | 87 | 85 | 86 | 84 | 75 | 64 | 51 | 46 | 45 | 45 | 40 | 37 | 40 | 42 | 41 | 44 | 49 | 47 | 50 | 53 | 56 | 56 | 56 | 57 | 52 | 56 |
| -3 | **85** | 88 | 100 | 97 | 87 | 85 | 94 | 93 | 92 | 81 | 54 | 46 | 47 | 44 | 47 | 46 | 38 | 38 | 40 | 37 | 45 | 50 | 52 | 51 | 52 | 53 | 53 | 56 | 56 | 57 |
| -2 | **95** | 93 | 91 | 88 | 89 | 91 | 98 | 95 | 91 | 77 | 54 | 50 | 45 | 39 | 40 | 44 | 41 | 43 | 47 | 47 | 45 | 49 | 52 | 54 | 58 | 58 | 56 | 54 | 57 | 59 |
| -1 | **104** | 94 | 97 | 94 | 96 | 99 | 101 | 102 | 102 | 87 | 60 | 49 | 47 | 46 | 45 | 43 | 39 | 40 | 43 | 44 | 40 | 44 | 51 | 57 | 58 | 58 | 59 | 59 | 54 | 59 |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0** | **101** | **105** | **99** | **102** | **99** | **105** | **103** | **101** | **95** | **77** | **54** | **44** | **45** | **44** | **42** | **40** | **43** | **45** | **44** | **43** | **43** | **44** | **52** | **58** | **59** | **59** | **64** | **65** | **58** | **58** |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | **100** | 102 | 108 | 109 | 110 | 112 | 109 | 110 | 106 | 85 | 59 | 46 | 44 | 44 | 43 | 44 | 45 | 44 | 44 | 47 | 46 | 45 | 50 | 57 | 59 | 61 | 65 | 65 | 61 | 58 |
| 2 | **106** | 107 | 116 | 116 | 117 | 116 | 115 | 112 | 93 | 65 | 51 | 51 | 49 | 46 | 43 | 42 | 44 | 45 | 44 | 45 | 47 | 49 | 53 | 57 | 58 | 58 | 59 | 60 | 59 | 58 |
| 3 | **111** | 109 | 116 | 121 | 119 | 119 | 123 | 120 | 99 | 70 | 51 | 46 | 46 | 49 | 46 | 40 | 39 | 42 | 41 | 40 | 40 | 47 | 53 | 57 | 60 | 61 | 64 | 66 | 60 | 58 |
| 4 | **112** | 116 | 116 | 119 | 122 | 122 | 124 | 112 | 85 | 63 | 49 | 44 | 43 | 43 | 43 | 43 | 42 | 42 | 41 | 40 | 44 | 50 | 52 | 54 | 58 | 60 | 59 | 59 | 60 | 57 |
| 5 | **118** | 122 | 126 | 121 | 128 | 132 | 129 | 104 | 71 | 51 | 46 | 44 | 43 | 44 | 42 | 43 | 44 | 42 | 39 | 45 | 44 | 51 | 54 | 53 | 53 | 56 | 59 | 60 | 60 | 59 |
| 6 | **120** | 119 | 120 | 126 | 130 | 130 | 110 | 74 | 51 | 54 | 54 | 43 | 45 | 45 | 43 | 42 | 44 | 43 | 42 | 46 | 47 | 52 | 56 | 58 | 60 | 63 | 61 | 58 | 59 | 58 |
| 7 | **123** | 127 | 135 | 136 | 135 | 120 | 97 | 72 | 51 | 44 | 46 | 50 | 45 | 42 | 40 | 43 | 45 | 42 | 43 | 50 | 52 | 56 | 58 | 60 | 64 | 66 | 65 | 60 | 61 | 59 |
| 8 | **131** | 135 | 137 | 137 | 121 | 96 | 68 | 54 | 47 | 44 | 43 | 45 | 45 | 46 | 46 | 45 | 44 | 43 | 45 | 49 | 53 | 59 | 61 | 61 | 63 | 65 | 66 | 64 | 65 | 61 |
| 9 | **120** | 128 | 128 | 122 | 97 | 73 | 50 | 45 | 47 | 44 | 43 | 47 | 44 | 44 | 46 | 47 | 45 | 40 | 44 | 51 | 53 | 59 | 64 | 64 | 63 | 64 | 65 | 64 | 67 | 66 |
| 10 | **95** | 99 | 94 | 76 | 57 | 57 | 52 | 46 | 45 | 47 | 47 | 47 | 42 | 45 | 47 | 45 | 42 | 43 | 47 | 50 | 57 | 61 | 64 | 63 | 63 | 65 | 66 | 65 | 67 | 71 |
| 11 | **68** | 66 | 57 | 51 | 52 | 47 | 46 | 51 | 52 | 44 | 40 | 46 | 47 | 47 | 47 | 45 | 43 | 46 | 52 | 54 | 61 | 65 | 66 | 65 | 66 | 68 | 68 | 67 | 71 | 74 |
| 12 | **54** | 49 | 47 | 46 | 47 | 47 | 46 | 46 | 47 | 44 | 43 | 46 | 39 | 40 | 40 | 38 | 42 | 53 | 61 | 61 | 60 | 66 | 70 | 71 | 71 | 71 | 70 | 66 | 77 | 78 |
| 13 | **49** | 50 | 45 | 45 | 50 | 49 | 46 | 45 | 46 | 44 | 42 | 44 | 44 | 46 | 46 | 49 | 53 | 59 | 64 | 70 | 69 | 72 | 71 | 72 | 72 | 72 | 75 | 77 | 73 | 77 |
| 14 | **39** | 43 | 46 | 44 | 46 | 44 | 42 | 42 | 43 | 42 | 39 | 39 | 44 | 50 | 54 | 59 | 65 | 67 | 68 | 71 | 74 | 77 | 73 | 72 | 73 | 73 | 78 | 80 | 75 | 79 |
| 15 | **43** | 44 | 44 | 49 | 52 | 50 | 49 | 49 | 46 | 45 | 44 | 44 | 50 | 54 | 59 | 64 | 67 | 70 | 70 | 72 | 72 | 74 | 73 | 75 | 79 | 80 | 82 | 84 | 73 | 79 |
| 16 | **52** | 52 | 49 | 53 | 53 | 53 | 57 | 58 | 54 | 54 | 58 | 61 | 60 | 63 | 65 | 67 | 69 | 71 | 72 | 74 | 74 | 75 | 74 | 78 | 81 | 80 | 81 | 80 | 79 | 80 |
| 17 | **63** | 64 | 64 | 63 | 61 | 61 | 66 | 68 | 65 | 65 | 70 | 73 | 70 | 71 | 73 | 75 | 76 | 76 | 77 | 78 | 78 | 77 | 73 | 74 | 78 | 78 | 81 | 82 | 80 | 79 |
| 18 | **68** | 67 | 66 | 67 | 68 | 68 | 70 | 71 | 70 | 68 | 71 | 73 | 72 | 72 | 75 | 77 | 76 | 76 | 78 | 78 | 75 | 77 | 75 | 79 | 81 | 79 | 78 | 75 | 75 | 77 |
| 19 | **77** | 74 | 69 | 72 | 73 | 74 | 73 | 74 | 75 | 74 | 75 | 78 | 73 | 72 | 74 | 74 | 72 | 73 | 76 | 76 | 75 | 75 | 73 | 77 | 80 | 79 | 80 | 80 | 80 | 79 |
| 20 | **77** | 79 | 78 | 73 | 74 | 77 | 77 | 75 | 75 | 73 | 73 | 78 | 78 | 77 | 78 | 78 | 74 | 74 | 76 | 74 | 78 | 78 | 75 | 78 | 80 | 77 | 77 | 75 | 81 | 80 |

**Table 3 - Pixel values around centre - sliced in x direction going right**

| user x | | | | | | | | | 1[440] | | | | | | | | | 2[449] | | | | | | | | | | | 3[460] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | -28 | -27 | -26 | -25 | -24 | -23 | -22 | -21 | -20 | -19 | -18 | -17 | -16 | -15 | -14 | -13 | -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | **0** |
| -20 | 84 | 85 | 85 | 82 | 80 | 80 | 80 | 81 | 82 | 84 | 80 | 77 | 77 | 75 | 76 | 78 | 80 | 75 | 75 | 78 | 81 | 81 | 75 | 73 | 77 | 79 | 80 | 77 | **73** |
| -19 | 94 | 89 | 82 | 85 | 84 | 79 | 81 | 82 | 78 | 75 | 75 | 75 | 74 | 71 | 72 | 73 | 73 | 75 | 75 | 75 | 75 | 75 | 73 | 71 | 71 | 74 | 77 | 79 | **79** |
| -18 | 86 | 86 | 86 | 86 | 84 | 81 | 80 | 79 | 78 | 73 | 70 | 70 | 72 | 72 | 71 | 71 | 71 | 70 | 70 | 72 | 73 | 72 | 67 | 63 | 71 | 71 | 70 | 67 | **67** |
| -17 | 85 | 87 | 87 | 82 | 79 | 79 | 78 | 77 | 75 | 71 | 67 | 66 | 66 | 66 | 65 | 65 | 65 | 64 | 64 | 66 | 67 | 66 | 61 | 59 | 64 | 65 | 65 | 61 | **60** |
| -16 | 86 | 86 | 85 | 77 | 73 | 75 | 74 | 73 | 68 | 66 | 65 | 63 | 61 | 60 | 59 | 58 | 60 | 60 | 59 | 59 | 61 | 63 | 61 | 60 | 58 | 60 | 60 | 58 | **58** |
| -15 | 80 | 78 | 79 | 75 | 73 | 73 | 70 | 68 | 64 | 63 | 59 | 58 | 57 | 57 | 54 | 52 | 53 | 49 | 46 | 50 | 53 | 53 | 53 | 52 | 53 | 52 | 51 | 50 | **50** |
| -14 | 77 | 74 | 73 | 73 | 71 | 68 | 63 | 61 | 64 | 61 | 58 | 53 | 51 | 50 | 49 | 49 | 50 | 46 | 45 | 50 | 51 | 47 | 47 | 50 | 50 | 50 | 49 | 47 | **49** |
| -13 | 75 | 73 | 71 | 70 | 66 | 61 | 59 | 57 | 60 | 60 | 57 | 51 | 47 | 47 | 47 | 46 | 45 | 46 | 47 | 47 | 46 | 46 | 49 | 51 | 46 | 47 | 47 | 47 | **47** |
| -12 | 70 | 71 | 68 | 67 | 64 | 58 | 58 | 57 | 56 | 53 | 51 | 49 | 51 | 52 | 50 | 46 | 45 | 49 | 47 | 44 | 44 | 51 | 54 | 53 | 50 | 53 | 54 | 53 | **51** |
| -11 | 66 | 73 | 72 | 70 | 70 | 67 | 64 | 66 | 61 | 59 | 49 | 47 | 50 | 46 | 47 | 52 | 49 | 50 | 47 | 46 | 47 | 49 | 53 | 63 | 62 | 62 | 62 | 62 | **60** |
| -10 | 72 | 73 | 69 | 69 | 75 | 74 | 66 | 61 | 56 | 53 | 46 | 45 | 46 | 43 | 45 | 48 | 49 | 44 | 43 | 46 | 49 | 49 | 55 | 63 | 66 | 65 | 64 | 62 | **62** |
| -9 | 75 | 75 | 72 | 71 | 75 | 75 | 68 | 63 | 59 | 57 | 50 | 47 | 45 | 43 | 45 | 48 | 46 | 45 | 46 | 45 | 47 | 58 | 64 | 60 | 65 | 66 | 67 | 69 | **69** |
| -8 | 78 | 79 | 78 | 74 | 74 | 74 | 70 | 66 | 52 | 51 | 49 | 47 | 44 | 42 | 42 | 43 | 51 | 45 | 47 | 53 | 56 | 64 | 70 | 66 | 70 | 70 | 69 | 69 | **69** |
| -7 | 82 | 81 | 78 | 74 | 73 | 75 | 72 | 65 | 50 | 49 | 47 | 45 | 44 | 45 | 47 | 48 | 51 | 45 | 50 | 59 | 61 | 67 | 73 | 69 | 72 | 72 | 72 | 72 | **73** |
| -6 | 82 | 82 | 82 | 77 | 71 | 72 | 72 | 66 | 47 | 47 | 47 | 44 | 45 | 47 | 46 | 46 | 46 | 46 | 53 | 60 | 64 | 68 | 71 | 69 | 72 | 74 | 77 | 79 | **80** |
| -5 | 80 | 82 | 84 | 79 | 71 | 72 | 71 | 63 | 53 | 52 | 47 | 40 | 42 | 46 | 44 | 45 | 49 | 45 | 53 | 70 | 73 | 65 | 65 | 74 | 78 | 77 | 77 | 77 | **79** |
| -4 | 84 | 80 | 79 | 78 | 75 | 78 | 71 | 56 | 53 | 52 | 46 | 38 | 42 | 46 | 43 | 45 | 49 | 51 | 60 | 73 | 78 | 68 | 66 | 77 | 74 | 75 | 78 | 80 | **81** |
| -3 | 79 | 82 | 82 | 80 | 80 | 79 | 70 | 59 | 44 | 45 | 44 | 43 | 44 | 43 | 44 | 48 | 51 | 52 | 61 | 71 | 72 | 66 | 72 | 98 | 135 | 103 | 95 | 95 | **85** |
| -2 | 87 | 84 | 81 | 81 | 79 | 71 | 63 | 56 | 46 | 45 | 43 | 42 | 43 | 43 | 44 | 45 | 44 | 59 | 70 | 86 | 78 | 70 | 136 | 218 | 212 | 142 | 104 | 100 | **95** |
| -1 | 84 | 81 | 79 | 80 | 80 | 77 | 71 | 67 | 51 | 49 | 46 | 45 | 44 | 45 | 46 | 47 | 38 | 45 | 63 | 82 | 88 | 67 | 110 | 237 | 255 | 206 | 118 | 94 | **104** |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0** | **85** | **86** | **82** | **77** | **73** | **72** | **65** | **56** | **54** | **47** | **44** | **44** | **44** | **46** | **46** | **43** | **47** | **60** | **67** | **80** | **78** | **88** | **158** | **227** | **214** | **138** | **108** | **104** | **101** |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 81 | 84 | 84 | 80 | 79 | 79 | 73 | 65 | 53 | 49 | 47 | 45 | 42 | 43 | 44 | 43 | 43 | 49 | 66 | 84 | 94 | 91 | 85 | 92 | 107 | 93 | 95 | 102 | **100** |
| 2 | 80 | 80 | 80 | 82 | 80 | 74 | 68 | 65 | 58 | 53 | 50 | 47 | 46 | 45 | 45 | 44 | 40 | 53 | 75 | 89 | 91 | 87 | 91 | 98 | 93 | 95 | 98 | 105 | **106** |
| 3 | 85 | 85 | 85 | 86 | 84 | 77 | 71 | 67 | 54 | 54 | 53 | 50 | 46 | 44 | 44 | 47 | 45 | 47 | 63 | 80 | 85 | 89 | 101 | 105 | 99 | 103 | 108 | 112 | **111** |
| 4 | 82 | 86 | 87 | 82 | 80 | 79 | 72 | 65 | 53 | 56 | 53 | 50 | 48 | 45 | 46 | 51 | 44 | 46 | 70 | 88 | 88 | 93 | 101 | 101 | 105 | 105 | 108 | 108 | **112** |
| 5 | 88 | 88 | 91 | 92 | 89 | 84 | 77 | 69 | 61 | 58 | 57 | 53 | 52 | 52 | 49 | 49 | 43 | 45 | 50 | 63 | 79 | 87 | 92 | 100 | 102 | 106 | 113 | 116 | **118** |
| 6 | 87 | 84 | 84 | 85 | 84 | 81 | 75 | 69 | 59 | 54 | 52 | 50 | 50 | 50 | 47 | 47 | 50 | 47 | 49 | 59 | 78 | 91 | 98 | 101 | 106 | 112 | 115 | 118 | **120** |
| 7 | 89 | 84 | 84 | 85 | 84 | 84 | 84 | 78 | 61 | 58 | 56 | 51 | 50 | 50 | 47 | 49 | 49 | 47 | 47 | 50 | 58 | 73 | 92 | 102 | 116 | 112 | 119 | 126 | **123** |
| 8 | 92 | 87 | 88 | 87 | 81 | 80 | 81 | 77 | 61 | 58 | 56 | 51 | 47 | 49 | 47 | 51 | 51 | 50 | 47 | 49 | 56 | 68 | 86 | 98 | 122 | 124 | 128 | 129 | **131** |
| 9 | 86 | 84 | 86 | 87 | 81 | 79 | 78 | 73 | 70 | 65 | 60 | 53 | 50 | 49 | 46 | 46 | 51 | 50 | 45 | 44 | 47 | 52 | 61 | 74 | 98 | 110 | 119 | 117 | **120** |
| 10 | 82 | 82 | 84 | 85 | 84 | 81 | 78 | 73 | 71 | 64 | 59 | 53 | 52 | 52 | 47 | 45 | 52 | 52 | 47 | 45 | 46 | 45 | 50 | 63 | 89 | 95 | 103 | 102 | **95** |
| 11 | 79 | 80 | 80 | 81 | 82 | 80 | 77 | 75 | 74 | 69 | 64 | 58 | 54 | 53 | 47 | 46 | 53 | 50 | 45 | 45 | 50 | 49 | 45 | 46 | 56 | 58 | 59 | 63 | **68** |
| 12 | 85 | 87 | 85 | 81 | 79 | 74 | 71 | 73 | 69 | 66 | 64 | 58 | 52 | 50 | 46 | 47 | 47 | 47 | 47 | 46 | 46 | 47 | 50 | 52 | 57 | 46 | 46 | 56 | **54** |
| 13 | 86 | 84 | 87 | 88 | 85 | 85 | 85 | 80 | 77 | 75 | 72 | 67 | 64 | 60 | 57 | 52 | 43 | 45 | 45 | 43 | 43 | 46 | 50 | 50 | 47 | 50 | 49 | 45 | **49** |
| 14 | 87 | 85 | 87 | 87 | 85 | 86 | 87 | 85 | 80 | 80 | 77 | 70 | 65 | 61 | 60 | 59 | 50 | 51 | 49 | 44 | 43 | 49 | 52 | 53 | 45 | 40 | 45 | 44 | **39** |
| 15 | 90 | 88 | 88 | 88 | 86 | 90 | 92 | 91 | 83 | 84 | 82 | 78 | 74 | 72 | 71 | 70 | 63 | 63 | 59 | 53 | 52 | 57 | 60 | 61 | 49 | 47 | 46 | 44 | **43** |
| 16 | 88 | 88 | 88 | 88 | 87 | 88 | 91 | 91 | 84 | 84 | 82 | 80 | 78 | 77 | 74 | 71 | 66 | 67 | 65 | 60 | 58 | 60 | 63 | 61 | 58 | 56 | 52 | 50 | **52** |
| 17 | 90 | 91 | 92 | 92 | 91 | 92 | 91 | 90 | 91 | 90 | 87 | 85 | 84 | 85 | 84 | 81 | 78 | 80 | 79 | 74 | 72 | 73 | 73 | 72 | 70 | 63 | 65 | 66 | **63** |
| 18 | 90 | 94 | 95 | 95 | 94 | 94 | 93 | 92 | 88 | 90 | 91 | 88 | 86 | 87 | 87 | 87 | 81 | 84 | 82 | 78 | 75 | 77 | 78 | 77 | 72 | 70 | 73 | 72 | **68** |
| 19 | 88 | 93 | 94 | 93 | 93 | 94 | 93 | 93 | 90 | 92 | 93 | 91 | 88 | 88 | 87 | 86 | 85 | 86 | 85 | 80 | 79 | 81 | 83 | 83 | 77 | 81 | 80 | 75 | **77** |
| 20 | 90 | 94 | 95 | 93 | 94 | 97 | 97 | 98 | 94 | 94 | 93 | 92 | 92 | 93 | 93 | 91 | 86 | 88 | 87 | 84 | 83 | 85 | 86 | 86 | 83 | 80 | 81 | 80 | **77** |

**Table 4 - Pixel values around centre - sliced in x direction going left**

| user y | 3[299] | | | | | | | | | | | 4[309] | | | | | 5[314] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| -20 | **54** | | 53 | 58 | 54 | 53 | 61 | 59 | 61 | 61 | 70 | 71 | 74 | 69 | 77 | 80 | 83 | 84 | 91 | 88 | 90 | 94 | 98 | 94 | 99 |
| -19 | **47** | | 49 | 53 | 54 | 56 | 58 | 54 | 58 | 58 | 65 | 64 | 69 | 66 | 75 | 80 | 84 | 84 | 90 | 90 | 92 | 94 | 98 | 95 | 99 |
| -18 | **44** | | 47 | 50 | 53 | 53 | 57 | 52 | 56 | 56 | 60 | 59 | 64 | 64 | 72 | 77 | 82 | 82 | 87 | 91 | 93 | 93 | 98 | 97 | 99 |
| -17 | **44** | | 45 | 47 | 50 | 50 | 53 | 50 | 51 | 51 | 53 | 53 | 58 | 58 | 67 | 70 | 78 | 80 | 85 | 88 | 91 | 92 | 97 | 99 | 99 |
| -16 | **44** | | 42 | 46 | 46 | 48 | 52 | 50 | 50 | 47 | 50 | 52 | 54 | 52 | 64 | 65 | 74 | 78 | 84 | 86 | 88 | 92 | 97 | 99 | 98 |
| -15 | **46** | | 43 | 45 | 44 | 45 | 52 | 50 | 50 | 49 | 49 | 52 | 53 | 50 | 60 | 61 | 72 | 77 | 85 | 87 | 88 | 93 | 94 | 97 | 94 |
| -14 | **46** | | 44 | 45 | 44 | 46 | 49 | 47 | 47 | 47 | 46 | 47 | 47 | 46 | 57 | 60 | 71 | 74 | 84 | 87 | 87 | 93 | 91 | 92 | 91 |
| -13 | **43** | | 43 | 44 | 47 | 51 | 49 | 47 | 49 | 51 | 46 | 45 | 46 | 47 | 52 | 59 | 70 | 71 | 81 | 87 | 86 | 91 | 87 | 88 | 88 |
| -12 | **47** | | 43 | 40 | 45 | 44 | 43 | 50 | 49 | 51 | 51 | 52 | 53 | 47 | 43 | 50 | 63 | 66 | 78 | 81 | 85 | 86 | 87 | 92 | 93 |
| -11 | **60** | | 49 | 53 | 47 | 46 | 45 | 47 | 47 | 50 | 50 | 52 | 50 | 47 | 45 | 51 | 63 | 67 | 80 | 84 | 86 | 88 | 90 | 91 | 91 |
| -10 | **67** | | 66 | 75 | 63 | 70 | 50 | 49 | 47 | 47 | 45 | 47 | 45 | 47 | 45 | 49 | 59 | 65 | 79 | 82 | 85 | 87 | 91 | 94 | 90 |
| -9 | **80** | | 84 | 89 | 80 | 88 | 63 | 59 | 50 | 49 | 44 | 45 | 45 | 46 | 43 | 44 | 53 | 60 | 74 | 78 | 80 | 84 | 92 | 93 | 94 |
| -8 | **78** | | 94 | 91 | 85 | 88 | 79 | 78 | 58 | 56 | 47 | 46 | 50 | 46 | 43 | 43 | 52 | 58 | 72 | 75 | 79 | 83 | 92 | 92 | 94 |
| -7 | **88** | | 91 | 87 | 89 | 93 | 87 | 91 | 73 | 68 | 52 | 45 | 49 | 47 | 46 | 49 | 57 | 60 | 73 | 77 | 81 | 85 | 94 | 95 | 93 |
| -6 | **158** | | 85 | 91 | 101 | 101 | 92 | 98 | 92 | 86 | 61 | 50 | 45 | 50 | 50 | 52 | 60 | 63 | 73 | 78 | 83 | 86 | 98 | 95 | 96 |
| -5 | **227** | | 92 | 98 | 105 | 101 | 100 | 101 | 102 | 98 | 74 | 63 | 46 | 52 | 50 | 53 | 61 | 61 | 72 | 77 | 83 | 86 | 96 | 96 | 98 |
| -4 | **214** | | 107 | 93 | 99 | 105 | 102 | 106 | 116 | 122 | 98 | 89 | 56 | 57 | 47 | 45 | 49 | 58 | 70 | 72 | 77 | 83 | 89 | 86 | 87 |
| -3 | **138** | | 93 | 95 | 103 | 105 | 106 | 112 | 112 | 124 | 110 | 95 | 58 | 46 | 50 | 40 | 47 | 56 | 63 | 70 | 81 | 80 | 79 | 81 | 89 |
| -2 | **108** | | 95 | 98 | 108 | 108 | 113 | 115 | 119 | 128 | 119 | 103 | 59 | 46 | 49 | 45 | 46 | 52 | 65 | 73 | 80 | 81 | 75 | 81 | 86 |
| -1 | **104** | | 102 | 105 | 112 | 108 | 116 | 118 | 126 | 129 | 117 | 102 | 63 | 56 | 45 | 44 | 44 | 50 | 66 | 72 | 75 | 80 | 81 | 82 | 84 |
| | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0** | **101** | | **100** | **106** | **111** | **112** | **118** | **120** | **123** | **131** | **120** | **95** | **68** | **54** | **49** | **39** | **43** | **52** | **63** | **68** | **77** | **77** | **84** | **81** | **85** |
| | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | **105** | | 102 | 107 | 109 | 116 | 122 | 119 | 127 | 135 | 128 | 99 | 66 | 49 | 50 | 43 | 44 | 52 | 64 | 67 | 74 | 79 | 78 | 81 | 86 |
| 2 | **99** | | 108 | 116 | 116 | 116 | 126 | 120 | 135 | 137 | 128 | 94 | 57 | 47 | 45 | 46 | 44 | 49 | 64 | 66 | 69 | 78 | 77 | 84 | 85 |
| 3 | **102** | | 109 | 116 | 121 | 119 | 121 | 126 | 136 | 137 | 122 | 76 | 51 | 46 | 45 | 44 | 49 | 53 | 63 | 67 | 72 | 73 | 81 | 84 | 85 |
| 4 | **99** | | 110 | 117 | 119 | 122 | 128 | 130 | 135 | 121 | 97 | 57 | 52 | 47 | 50 | 46 | 52 | 53 | 61 | 68 | 73 | 74 | 82 | 80 | 79 |
| 5 | **105** | | 112 | 116 | 119 | 122 | 132 | 130 | 120 | 96 | 73 | 57 | 47 | 47 | 49 | 44 | 50 | 53 | 61 | 68 | 74 | 77 | 80 | 79 | 82 |
| 6 | **103** | | 109 | 115 | 123 | 124 | 129 | 110 | 97 | 68 | 50 | 52 | 46 | 46 | 46 | 42 | 49 | 57 | 66 | 70 | 73 | 77 | 80 | 79 | 86 |
| 7 | **101** | | 110 | 112 | 120 | 112 | 104 | 74 | 72 | 54 | 45 | 46 | 51 | 46 | 45 | 42 | 49 | 58 | 68 | 71 | 74 | 75 | 81 | 80 | 84 |
| 8 | **95** | | 106 | 93 | 99 | 85 | 71 | 51 | 51 | 47 | 47 | 45 | 52 | 47 | 46 | 43 | 46 | 54 | 65 | 70 | 75 | 75 | 81 | 81 | 81 |
| 9 | **77** | | 85 | 65 | 70 | 63 | 51 | 54 | 44 | 44 | 44 | 47 | 44 | 44 | 44 | 42 | 45 | 54 | 65 | 68 | 74 | 73 | 79 | 81 | 81 |
| 10 | **54** | | 59 | 51 | 51 | 49 | 46 | 54 | 46 | 43 | 43 | 47 | 40 | 43 | 42 | 39 | 44 | 58 | 70 | 71 | 75 | 73 | 78 | 79 | 82 |
| 11 | **44** | | 46 | 51 | 46 | 44 | 44 | 43 | 50 | 45 | 47 | 47 | 46 | 46 | 44 | 39 | 44 | 61 | 73 | 73 | 78 | 78 | 78 | 74 | 80 |
| 12 | **45** | | 44 | 49 | 46 | 43 | 43 | 45 | 45 | 45 | 44 | 42 | 47 | 39 | 44 | 44 | 50 | 60 | 70 | 72 | 73 | 78 | 79 | 74 | 79 |
| 13 | **44** | | 44 | 46 | 49 | 43 | 44 | 45 | 42 | 46 | 44 | 45 | 47 | 40 | 46 | 50 | 54 | 63 | 71 | 72 | 72 | 77 | 78 | 82 | 77 |
| 14 | **42** | | 43 | 43 | 46 | 43 | 42 | 43 | 40 | 46 | 46 | 47 | 47 | 40 | 46 | 54 | 59 | 65 | 73 | 75 | 74 | 78 | 78 | 85 | 84 |
| 15 | **40** | | 44 | 42 | 40 | 43 | 43 | 42 | 43 | 45 | 47 | 45 | 45 | 38 | 49 | 59 | 64 | 67 | 75 | 77 | 74 | 78 | 79 | 79 | 85 |
| 16 | **43** | | 45 | 44 | 39 | 42 | 44 | 44 | 45 | 44 | 45 | 42 | 43 | 42 | 53 | 65 | 67 | 69 | 76 | 76 | 72 | 74 | 80 | 75 | 75 |
| 17 | **45** | | 44 | 45 | 42 | 42 | 42 | 43 | 42 | 43 | 40 | 43 | 46 | 53 | 59 | 67 | 70 | 71 | 76 | 76 | 73 | 74 | 77 | 78 | 75 |
| 18 | **44** | | 44 | 44 | 41 | 41 | 39 | 42 | 43 | 45 | 44 | 47 | 52 | 61 | 64 | 68 | 70 | 72 | 77 | 78 | 76 | 76 | 74 | 81 | 82 |
| 19 | **43** | | 47 | 45 | 40 | 40 | 45 | 46 | 50 | 49 | 51 | 50 | 54 | 61 | 70 | 71 | 72 | 74 | 78 | 78 | 76 | 74 | 75 | 82 | 82 |
| 20 | **43** | | 46 | 47 | 40 | 44 | 44 | 47 | 52 | 53 | 53 | 57 | 61 | 60 | 69 | 74 | 72 | 74 | 78 | 75 | 75 | 78 | 78 | 80 | 82 |

**Table 5 - Pixel values around centre - sliced in y direction going down**

| user y | | | | | | | | | 1[284] | | | | 2[288] | | | | | | | | | | | | 3[299] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | -23 | -22 | -21 | -20 | -19 | -18 | -17 | -16 | -15 | -14 | -13 | -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | | **0** |
| -20 | 82 | 79 | 81 | 82 | 78 | 78 | 75 | 68 | 64 | 64 | 60 | 56 | 61 | 56 | 59 | 52 | 50 | 47 | 53 | 53 | 44 | 46 | 51 | | **54** |
| -19 | 77 | 78 | 79 | 84 | 75 | 73 | 71 | 66 | 63 | 61 | 60 | 53 | 59 | 53 | 57 | 51 | 49 | 47 | 52 | 52 | 45 | 45 | 49 | | **47** |
| -18 | 78 | 80 | 77 | 80 | 75 | 70 | 67 | 65 | 59 | 58 | 57 | 51 | 49 | 46 | 50 | 49 | 47 | 47 | 47 | 46 | 44 | 43 | 46 | | **44** |
| -17 | 82 | 82 | 78 | 77 | 75 | 70 | 66 | 63 | 58 | 53 | 51 | 49 | 47 | 45 | 47 | 47 | 45 | 44 | 40 | 38 | 43 | 42 | 45 | | **44** |
| -16 | 84 | 80 | 80 | 77 | 74 | 72 | 66 | 61 | 57 | 51 | 47 | 51 | 50 | 46 | 45 | 44 | 44 | 45 | 42 | 42 | 44 | 43 | 44 | | **44** |
| -15 | 82 | 76 | 78 | 75 | 71 | 72 | 66 | 60 | 57 | 50 | 47 | 52 | 46 | 43 | 43 | 42 | 45 | 47 | 46 | 46 | 43 | 43 | 45 | | **46** |
| -14 | 82 | 77 | 77 | 76 | 72 | 71 | 65 | 59 | 54 | 49 | 47 | 50 | 47 | 45 | 45 | 42 | 47 | 46 | 44 | 43 | 44 | 44 | 46 | | **46** |
| -13 | 80 | 80 | 78 | 78 | 73 | 71 | 65 | 58 | 52 | 49 | 46 | 46 | 52 | 48 | 48 | 43 | 48 | 46 | 45 | 45 | 48 | 45 | 47 | | **43** |
| -12 | 85 | 81 | 84 | 80 | 73 | 71 | 65 | 60 | 53 | 50 | 45 | 45 | 49 | 49 | 46 | 51 | 51 | 46 | 49 | 49 | 51 | 44 | 38 | | **47** |
| -11 | 82 | 85 | 84 | 75 | 75 | 70 | 64 | 60 | 49 | 46 | 46 | 49 | 50 | 44 | 45 | 45 | 45 | 46 | 45 | 51 | 52 | 59 | 45 | | **60** |
| -10 | 79 | 85 | 84 | 75 | 75 | 70 | 64 | 59 | 46 | 45 | 47 | 47 | 47 | 43 | 46 | 47 | 50 | 53 | 53 | 60 | 61 | 70 | 63 | | **67** |
| -9 | 82 | 84 | 79 | 78 | 75 | 72 | 66 | 59 | 50 | 50 | 47 | 44 | 46 | 46 | 45 | 53 | 59 | 60 | 70 | 73 | 71 | 86 | 82 | | **80** |
| -8 | 85 | 81 | 78 | 81 | 75 | 73 | 67 | 61 | 53 | 51 | 46 | 44 | 47 | 49 | 47 | 56 | 61 | 64 | 73 | 78 | 72 | 78 | 88 | | **78** |
| -7 | 85 | 82 | 79 | 81 | 75 | 72 | 66 | 63 | 53 | 47 | 46 | 51 | 49 | 49 | 58 | 64 | 67 | 68 | 65 | 68 | 66 | 70 | 67 | | **88** |
| -6 | 87 | 85 | 80 | 75 | 73 | 67 | 61 | 61 | 53 | 47 | 49 | 54 | 53 | 55 | 64 | 70 | 73 | 71 | 65 | 66 | 72 | 136 | 110 | | **158** |
| -5 | 86 | 86 | 82 | 73 | 71 | 63 | 59 | 60 | 52 | 50 | 51 | 53 | 63 | 63 | 60 | 66 | 69 | 69 | 74 | 77 | 98 | 218 | 237 | | **227** |
| -4 | 88 | 84 | 82 | 77 | 71 | 71 | 64 | 58 | 53 | 50 | 46 | 50 | 62 | 66 | 65 | 70 | 72 | 72 | 78 | 74 | 135 | 212 | 255 | | **214** |
| -3 | 85 | 84 | 80 | 79 | 74 | 71 | 65 | 60 | 52 | 50 | 47 | 53 | 62 | 65 | 66 | 70 | 72 | 74 | 77 | 75 | 103 | 142 | 206 | | **138** |
| -2 | 82 | 82 | 77 | 80 | 77 | 70 | 65 | 60 | 51 | 49 | 47 | 54 | 62 | 64 | 67 | 69 | 72 | 77 | 77 | 78 | 95 | 104 | 118 | | **108** |
| -1 | 82 | 81 | 75 | 77 | 79 | 67 | 61 | 58 | 50 | 47 | 47 | 53 | 62 | 62 | 69 | 69 | 72 | 79 | 77 | 80 | 95 | 100 | 94 | | **104** |
| | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0** | **82** | **78** | **78** | **73** | **79** | **67** | **60** | **58** | **50** | **49** | **47** | **51** | **60** | **62** | **69** | **69** | **73** | **80** | **79** | **81** | **85** | **95** | **104** | | **101** |
| | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 81 | 77 | 80 | 75 | 77 | 70 | 61 | 59 | 53 | 49 | 47 | 50 | 59 | 62 | 67 | 70 | 74 | 80 | 81 | 83 | 88 | 93 | 94 | | **105** |
| 2 | 84 | 80 | 80 | 79 | 72 | 71 | 63 | 61 | 53 | 47 | 46 | 47 | 58 | 63 | 66 | 71 | 76 | 80 | 85 | 84 | 100 | 91 | 97 | | **99** |
| 3 | 87 | 84 | 78 | 81 | 66 | 71 | 63 | 61 | 53 | 44 | 45 | 46 | 58 | 64 | 65 | 72 | 77 | 79 | 86 | 84 | 97 | 88 | 94 | | **102** |
| 4 | 89 | 84 | 79 | 79 | 74 | 74 | 65 | 61 | 46 | 46 | 40 | 44 | 54 | 59 | 68 | 70 | 78 | 80 | 87 | 87 | 87 | 89 | 96 | | **99** |
| 5 | 89 | 85 | 79 | 75 | 75 | 75 | 65 | 61 | 50 | 50 | 42 | 43 | 52 | 58 | 68 | 70 | 75 | 77 | 84 | 85 | 85 | 91 | 99 | | **105** |
| 6 | 87 | 81 | 79 | 78 | 74 | 73 | 63 | 60 | 53 | 51 | 42 | 40 | 49 | 54 | 66 | 68 | 77 | 78 | 85 | 86 | 94 | 98 | 101 | | **103** |
| 7 | 86 | 80 | 81 | 79 | 73 | 73 | 64 | 63 | 57 | 53 | 43 | 42 | 42 | 45 | 54 | 60 | 71 | 75 | 82 | 84 | 93 | 95 | 102 | | **101** |
| 8 | 88 | 84 | 82 | 74 | 74 | 74 | 68 | 66 | 60 | 56 | 46 | 45 | 40 | 42 | 47 | 52 | 63 | 66 | 74 | 75 | 92 | 91 | 102 | | **95** |
| 9 | 86 | 82 | 80 | 74 | 73 | 73 | 71 | 68 | 59 | 54 | 49 | 46 | 44 | 44 | 47 | 47 | 53 | 53 | 60 | 64 | 81 | 77 | 87 | | **77** |
| 10 | 86 | 82 | 77 | 79 | 72 | 70 | 68 | 66 | 57 | 52 | 49 | 44 | 44 | 42 | 44 | 43 | 46 | 44 | 49 | 51 | 54 | 54 | 60 | | **54** |
| 11 | 91 | 88 | 77 | 79 | 73 | 68 | 68 | 66 | 57 | 53 | 50 | 44 | 45 | 40 | 42 | 42 | 46 | 44 | 46 | 46 | 46 | 50 | 49 | | **44** |
| 12 | 94 | 86 | 79 | 79 | 74 | 73 | 71 | 70 | 61 | 58 | 52 | 49 | 44 | 47 | 40 | 47 | 44 | 47 | 44 | 45 | 47 | 45 | 47 | | **45** |
| 13 | 91 | 93 | 84 | 79 | 74 | 74 | 71 | 68 | 64 | 60 | 53 | 51 | 46 | 48 | 43 | 46 | 43 | 45 | 44 | 45 | 44 | 39 | 46 | | **44** |
| 14 | 87 | 94 | 84 | 78 | 74 | 75 | 71 | 70 | 66 | 63 | 56 | 54 | 45 | 47 | 44 | 45 | 43 | 43 | 40 | 40 | 47 | 40 | 45 | | **42** |
| 15 | 88 | 87 | 79 | 76 | 74 | 75 | 73 | 72 | 67 | 64 | 58 | 58 | 45 | 45 | 43 | 44 | 44 | 44 | 39 | 37 | 46 | 44 | 43 | | **40** |
| 16 | 92 | 81 | 78 | 77 | 74 | 75 | 75 | 75 | 67 | 64 | 61 | 60 | 48 | 46 | 43 | 43 | 44 | 46 | 43 | 40 | 38 | 41 | 39 | | **43** |
| 17 | 94 | 82 | 82 | 79 | 75 | 77 | 77 | 77 | 67 | 65 | 63 | 61 | 48 | 47 | 44 | 44 | 45 | 48 | 43 | 42 | 38 | 43 | 40 | | **45** |
| 18 | 93 | 86 | 86 | 80 | 78 | 78 | 75 | 75 | 70 | 67 | 63 | 60 | 46 | 48 | 46 | 46 | 43 | 48 | 40 | 41 | 40 | 47 | 43 | | **44** |
| 19 | 89 | 87 | 85 | 80 | 79 | 79 | 74 | 73 | 72 | 70 | 61 | 59 | 47 | 51 | 47 | 46 | 39 | 46 | 40 | 44 | 37 | 47 | 44 | | **43** |
| 20 | 91 | 88 | 86 | 79 | 78 | 78 | 74 | 75 | 70 | 73 | 70 | 67 | 60 | 57 | 51 | 49 | 46 | 49 | 47 | 49 | 45 | 45 | 40 | | **43** |

**Table 6 - Pixel values around centre - sliced in y direction going up**

As expected the IR image has a limited range of values. It is not worth histogram equalising these as we are concerned with the actual data values and not how it looks on screen. It is always better to work with the raw data than with modified data.

The retinal reflection effect is also readily apparent by looking at the data around slices -4 to -6 at offset 0 in table 5. This will need careful thought to take it into account as the checks will start from the retinal reflection.

In this case to find the pupil/iris boundary a simple threshold value could be applied and if it is less than say 50 it is the boundary. This is rejected as it implies a priori knowledge and if the conditions alter then it will be ineffective.

What is required is an algorithm that uses as few fixed values (if any) and makes as few as possible assumptions. Where assumptions are made (as some will have to be) then these are all located in one place and held in variables such that future enhancements to the final program can be easily made (it could for instance be made self learning).

### 3.4.2 The difference data

Values either side of a pixel location can be used to ascertain how fast the change from one level to another is being made.

Due to noise and other concerns it is not acceptable to just look whether the current value is less than the previous value and more than the next value (as the pupil value is not constant, it also changes). What is required is to look at the previous two values and next two values and use the underlying data to ascertain which the best fit for the boundaries is.

The csv data was imported into a spreadsheet and a new sheet created such that the current value was made up of the summation of the previous two samples subtracted from the summation of the next two samples as in tables 8 to 11.

Notice that the two values immediately after the centre are not calculated as they will have no previous data in the final routines.

The bold figures have been decided by finding the largest positive difference and the first large negative difference (these will be tightened up soon) and as can be seen these are roughly eye shaped boundaries with the extremities roughly matching those set by the user.

The bold data is summarised in table 7 for use in comparisons later.

| | | Offsets from centre | | | |
|---|---|---|---|---|---|
| Slice number | | Sclera/Iris | Iris/Pupil | Pupil/Iris | Iris/Sclera |
| -13 | | n/a | n/a | 3 | 20 |
| -12 | | n/a | n/a | 4 | 20 |
| -11 | | 18 | 6 | 6 | 19 |
| -10 | | 18 | 6 | 6 | 19 |
| -9 | | 18 | 7 | 7 | 20 |
| -8 | | 19 | 8 | 7 | 20 |
| -7 | | 19 | 9 | 8 | 20 |
| -6 | | 19 | 9 | 8 | 22 |
| -5 | | 18 | 10 | 9 | 20 |
| -4 | | 18 | 10 | 9 | 19 |
| -3 | | 20 | 10 | 9 | 20 |
| -2 | | 20 | 11 | 9 | 22 |
| -1 | | 19 | 10 | 9 | 21 |
| 0 | | 19 | 11 | 9 | 21 |
| 1 | | 19 | 10 | 9 | 21 |
| 2 | | 19 | 10 | 8 | 21 |
| 3 | | 19 | 10 | 9 | 29 |
| 4 | | 20 | 10 | 8 | 20 |
| 5 | | 19 | 9 | 7 | 20 |
| 6 | | 19 | 8 | 7 | 20 |
| 7 | | 18 | 7 | 6 | 19 |
| 8 | | 17 | 6 | 5 | 19 |
| 9 | | 16 | 5 | 4 | 18 |
| 10 | | 17 | 4 | 3 | 18 |
| 11 | | 16 | 4 | n/a | n/a |

**Table 7 - Boundaries from differences as chosen by human input**

| user x | 3[460] | | | | | | | | | 4[469] | | | | | | | | | | | | 5[481] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| -20 | 73 | | | 0 | 7 | 3 | 1 | 5 | 4 | -5 | -10 | -5 | 1 | 4 | 5 | 1 | -5 | -7 | -3 | -1 | -2 | -3 | -6 | -11 | -14 | -10 | 1 | 5 | | |
| -19 | 79 | | | 0 | -11 | -7 | 2 | 2 | 2 | 2 | 0 | -3 | -3 | -1 | 0 | -1 | -5 | -9 | -8 | -6 | -6 | -9 | -9 | 0 | 11 | 14 | 6 | -6 | | |
| -18 | 67 | | | -8 | -6 | -1 | 2 | 1 | 3 | 9 | 6 | -4 | -11 | -9 | -3 | -3 | -5 | -7 | -5 | 1 | 2 | -3 | -5 | -3 | -4 | -6 | -5 | 0 | | |
| -17 | 60 | | | -6 | -2 | 1 | -2 | -11 | -12 | -4 | 0 | -3 | -6 | -5 | -6 | -10 | -8 | -1 | 4 | 1 | -8 | -11 | -2 | 6 | 1 | -10 | -8 | 6 | | |
| -16 | 58 | | | -2 | 1 | -1 | -7 | -13 | -11 | -3 | -2 | -4 | -6 | -6 | -9 | -14 | -10 | -1 | 4 | -4 | -14 | -11 | -1 | 4 | 2 | 1 | 3 | 5 | | |
| -15 | 50 | | | 10 | 3 | -11 | -21 | -16 | -6 | 3 | 1 | -9 | -16 | -15 | -9 | -4 | -4 | -8 | -8 | -8 | -13 | -16 | -6 | 9 | 10 | 1 | -4 | -2 | | |
| -14 | 49 | | | 0 | -10 | -14 | -13 | -9 | -2 | 4 | -1 | -12 | -18 | -16 | -10 | -6 | -5 | -9 | -14 | -14 | -11 | -7 | -2 | 0 | -3 | -5 | -2 | 0 | | |
| -13 | 47 | | | **11** | 7 | 0 | -7 | -11 | -13 | -10 | -7 | -7 | -10 | -12 | -14 | -15 | -12 | -5 | -7 | -15 | **-19** | -13 | -1 | 2 | 0 | -8 | -14 | -11 | | |
| -12 | 51 | | | 10 | **10** | 8 | 0 | -8 | -8 | -1 | -2 | -10 | -17 | -19 | -18 | -16 | -9 | -1 | -5 | -13 | **-16** | -12 | -6 | -6 | -6 | -8 | -12 | -11 | | |
| -11 | 60 | | | 11 | 15 | 21 | **24** | 17 | 3 | -7 | -5 | -2 | -2 | -1 | -3 | -5 | -4 | 0 | -11 | **-25** | -30 | -27 | -18 | -10 | 1 | 6 | -8 | -18 | | |
| -10 | 62 | | | 8 | 15 | 24 | **30** | 26 | 13 | 5 | -1 | -9 | -13 | -5 | 4 | 2 | -3 | -8 | -15 | **-20** | -20 | -19 | -18 | -14 | -4 | 3 | -6 | -16 | | |
| -9 | 69 | | | -3 | -3 | 13 | 35 | **40** | 29 | 15 | 12 | 8 | -1 | -5 | -3 | 0 | -3 | -7 | -11 | -15 | **-17** | -17 | -19 | -19 | -10 | -1 | -10 | -24 | | |
| -8 | 69 | | | 1 | 5 | 14 | 28 | **39** | 38 | 27 | 10 | -3 | -6 | 0 | 6 | 4 | -1 | -5 | -8 | -11 | **-14** | -17 | -21 | -22 | -13 | -5 | -6 | -13 | | |
| -7 | 73 | | | -3 | 1 | 7 | 19 | 36 | **49** | 42 | 26 | 12 | 6 | 3 | -1 | -3 | -1 | 6 | 4 | -7 | **-18** | -22 | -21 | -20 | -13 | -7 | -8 | -10 | | |
| -6 | 80 | | | 3 | 4 | 6 | 16 | 36 | **56** | 53 | 28 | 5 | 0 | 4 | 2 | -6 | -9 | -4 | -1 | -2 | -7 | -14 | **-18** | -18 | -12 | -8 | -1 | 5 | | |
| -5 | 79 | | | -5 | 2 | 6 | 15 | 35 | 58 | **61** | 44 | 21 | 11 | 11 | 6 | -2 | -4 | 2 | -1 | -11 | **-17** | -17 | -16 | -16 | -11 | -6 | -2 | 0 | | |
| -4 | 81 | | | -5 | -3 | 1 | 13 | 32 | 55 | **62** | 48 | 25 | 12 | 14 | 13 | 3 | -6 | -8 | -11 | **-13** | -12 | -10 | -13 | -15 | -9 | -4 | 3 | 4 | | |
| -3 | 85 | | | 16 | 18 | -3 | -13 | 6 | 52 | **85** | 80 | 44 | 9 | 0 | 7 | 15 | 15 | 7 | -6 | -17 | **-25** | -21 | -8 | -3 | -3 | -6 | -7 | -7 | | |
| -2 | 95 | | | 4 | -10 | -16 | -6 | 21 | 62 | **82** | 73 | 47 | 25 | 11 | -1 | -5 | -6 | -9 | -8 | -4 | -7 | -14 | **-18** | -15 | -8 | 2 | 5 | -2 | | |
| -1 | 104 | | | -4 | -9 | -13 | -9 | 11 | 56 | **95** | 93 | 54 | 18 | 8 | 11 | 12 | 5 | -5 | -5 | -1 | -8 | **-24** | -31 | -21 | -9 | -3 | 3 | 4 | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 101 | | | 0 | -7 | -3 | 8 | 36 | 73 | **98** | 83 | 42 | 12 | 7 | 6 | -2 | -7 | -4 | 2 | 2 | -9 | **-24** | -30 | -22 | -13 | -12 | -5 | 7 | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 100 | | | -12 | -4 | 0 | 6 | 30 | 75 | **111** | 101 | 56 | 18 | 3 | -1 | -2 | -1 | -2 | -4 | -3 | -4 | **-14** | -25 | -25 | -19 | -14 | -6 | 7 | | |
| 2 | 106 | | | -10 | 1 | 6 | 28 | 73 | **111** | 103 | 58 | 21 | 13 | 15 | 9 | 0 | -4 | -3 | -3 | -7 | -13 | **-18** | -19 | -14 | -7 | -4 | -3 | 0 | | |
| 3 | 111 | | | -13 | -5 | -3 | 19 | 73 | 122 | **122** | 77 | 26 | 2 | 6 | 16 | 14 | 3 | -2 | 1 | -4 | **-19** | -30 | -30 | -21 | -15 | -13 | -5 | 7 | | |
| 4 | 112 | | | -12 | -11 | 5 | 47 | 98 | **124** | 104 | 61 | 26 | 7 | 1 | 1 | 2 | 3 | 4 | 0 | -11 | **-21** | -22 | -18 | -16 | -13 | -6 | -1 | 2 | | |
| 5 | 118 | | | -12 | -14 | 16 | 85 | **139** | 136 | 85 | 35 | 10 | 4 | 2 | 0 | 0 | 4 | 3 | -3 | -14 | **-21** | -18 | -11 | -4 | -8 | -13 | -11 | -4 | | |
| 6 | 120 | | | -21 | 6 | 72 | 135 | **135** | 76 | 28 | 17 | 18 | 9 | 3 | 4 | 1 | 0 | -2 | -6 | -14 | **-20** | -21 | -19 | -15 | -10 | -1 | 6 | 7 | | |
| 7 | 123 | | | 7 | 54 | 102 | **132** | 122 | 79 | 27 | 0 | 3 | 14 | 12 | -1 | -5 | -2 | -5 | -15 | **-23** | -21 | -16 | -16 | -16 | -13 | -1 | 9 | 11 | | |
| 8 | 131 | | | 55 | 110 | **136** | 116 | 73 | 35 | 13 | 1 | -4 | -4 | -1 | 2 | 5 | 3 | -5 | -15 | **-24** | -26 | -20 | -12 | -8 | -9 | -6 | -1 | 5 | | |
| 9 | 120 | | | 86 | **127** | 124 | 78 | 32 | 8 | 2 | 0 | -1 | 0 | -2 | -4 | 5 | 9 | -3 | **-19** | -28 | -28 | -24 | -15 | -4 | -1 | -2 | -4 | -4 | | |
| 10 | 95 | | | **79** | 61 | 35 | 23 | 17 | 4 | -3 | 3 | 7 | 2 | -3 | 0 | 7 | 2 | -10 | **-22** | -28 | -28 | -20 | -8 | -3 | -4 | -5 | -4 | -7 | | |
| 11 | 68 | | | 24 | 15 | 6 | -4 | -3 | 13 | 17 | 3 | -10 | -8 | 1 | 6 | 5 | -6 | **-18** | -26 | -28 | -25 | -16 | -5 | -3 | -5 | -4 | -4 | -9 | | |
| 12 | 54 | | | 2 | 0 | 1 | 1 | 2 | 5 | 4 | 6 | 8 | 9 | 7 | -1 | -15 | -36 | -42 | -26 | -12 | -14 | -20 | -16 | -6 | 0 | 6 | -1 | -14 | | |
| 13 | 49 | | | -4 | -5 | 4 | 8 | 5 | 5 | 5 | 2 | -4 | -6 | -7 | -12 | -20 | -28 | -32 | -27 | -18 | -9 | -4 | -3 | -1 | -4 | -8 | -6 | -3 | | |
| 14 | 39 | | | -1 | 4 | 6 | 5 | 1 | 3 | 7 | 2 | -13 | -26 | -30 | -30 | -28 | -22 | -15 | -13 | -16 | -11 | 0 | 6 | 4 | -6 | -13 | -9 | -3 | | |
| 15 | 43 | | | -14 | -6 | 3 | 7 | 8 | 9 | 7 | -3 | -15 | -25 | -29 | -27 | -24 | -17 | -11 | -7 | -6 | -5 | -4 | -8 | -12 | -14 | -12 | 2 | 10 | | |
| 16 | 52 | | | -5 | -8 | -9 | -6 | 2 | 3 | -7 | -13 | -11 | -9 | -11 | -13 | -12 | -11 | -10 | -8 | -6 | -3 | -4 | -10 | -12 | -9 | -2 | 2 | 2 | | |
| 17 | 63 | | | 6 | 0 | -10 | -11 | -3 | -1 | -10 | -13 | -6 | -1 | -5 | -10 | -8 | -5 | -4 | -4 | -2 | 5 | 9 | 3 | -6 | -12 | -11 | -6 | 0 | | |
| 18 | 68 | | | -3 | -5 | -6 | -5 | 0 | 2 | -3 | -7 | -5 | -3 | -7 | -9 | -5 | -2 | -3 | -1 | 2 | 4 | -1 | -8 | -8 | -3 | 7 | 10 | 5 | | |
| 19 | 77 | | | -4 | -6 | -2 | -2 | -2 | -2 | -4 | -2 | 4 | 7 | 3 | -1 | 1 | -1 | -6 | -6 | -1 | 4 | 1 | -7 | -11 | -9 | -3 | -1 | 0 | | |
| 20 | 77 | | | 6 | -3 | -5 | 1 | 6 | 6 | -1 | -8 | -9 | -4 | 0 | 3 | 7 | 6 | 2 | -4 | -6 | -3 | -1 | -2 | -4 | -1 | 6 | 1 | -7 | | |

**Table 8 - Pixel difference values around centre - sliced in x direction going right**

| user x | | | | | | | | | 1[440] | | | | | | | | | 2[449] | | | | | | | | | | | 3[460] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | -28 | -27 | -26 | -25 | -24 | -23 | -22 | -21 | -20 | -19 | -18 | -17 | -16 | -15 | -14 | -13 | -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | **0** |
| -20 | | | -7 | -10 | -7 | -1 | 3 | 6 | 3 | -6 | -12 | -12 | -6 | 0 | 6 | 4 | -4 | -5 | 4 | 12 | 3 | -11 | -12 | 0 | 11 | 7 | | | **73** |
| -19 | | | -14 | -8 | -7 | -6 | -3 | -7 | -13 | -10 | -4 | -5 | -7 | -4 | 1 | 5 | 5 | 4 | 2 | 0 | -2 | -6 | -8 | -3 | 7 | 14 | | | **79** |
| -18 | | | -2 | -7 | -11 | -11 | -8 | -10 | -16 | -17 | -9 | 1 | 3 | 0 | -2 | -2 | -2 | 0 | 4 | 5 | -3 | -15 | -11 | 3 | 11 | 3 | | | **67** |
| -17 | | | -11 | -16 | -12 | -6 | -6 | -11 | -17 | -19 | -14 | -6 | -2 | -2 | -2 | -2 | -2 | 0 | 4 | 5 | -3 | -13 | -10 | 2 | 10 | 3 | | | **60** |
| -16 | | | -22 | -23 | -13 | -3 | -7 | -15 | -16 | -13 | -10 | -10 | -9 | -7 | -3 | 1 | 2 | 0 | 0 | 5 | 6 | 1 | -6 | -6 | -1 | 0 | | | **58** |
| -15 | | | -10 | -11 | -11 | -10 | -14 | -16 | -16 | -15 | -12 | -8 | -6 | -9 | -9 | -9 | -11 | -9 | 1 | 11 | 10 | 2 | -1 | -1 | -2 | -4 | | | **50** |
| -14 | | | -7 | -8 | -15 | -20 | -14 | -6 | -5 | -14 | -21 | -18 | -12 | -6 | -2 | -3 | -7 | -4 | 5 | 7 | -1 | -4 | 2 | 6 | 2 | -4 | | | **49** |
| -13 | | | -12 | -17 | -21 | -20 | -10 | 0 | 1 | -9 | -22 | -23 | -14 | -5 | -3 | -3 | 0 | 3 | 2 | -1 | 1 | 7 | 5 | -2 | -6 | -3 | | | **47** |
| -12 | | | -10 | -17 | -19 | -16 | -9 | -7 | -11 | -13 | -9 | -1 | 2 | -4 | -12 | -8 | 0 | 0 | -6 | -1 | 14 | 19 | 8 | -2 | 0 | 4 | | | **51** |
| -11 | | | 1 | -8 | -11 | -10 | -10 | -11 | -22 | -31 | **-23** | -12 | -3 | 2 | 5 | 6 | -2 | -8 | -6 | -1 | 9 | 23 | **29** | 22 | 8 | -1 | | | **60** |
| -10 | | | -1 | 7 | 2 | -17 | -32 | -31 | -28 | -26 | **-18** | -10 | -3 | 2 | 8 | 5 | -6 | -8 | 2 | 11 | 15 | 23 | **31** | 27 | 11 | -3 | | | **62** |
| -9 | | | -4 | 3 | 0 | -15 | -28 | -27 | -24 | -25 | **-24** | -19 | -9 | 1 | 6 | 3 | -2 | -3 | 1 | 14 | 31 | **32** | 20 | 9 | 9 | 11 | | | **69** |
| -8 | | | -9 | -9 | -8 | -12 | -30 | -41 | -36 | **-22** | -12 | -14 | -12 | -6 | 8 | 12 | 7 | 6 | 13 | 28 | **34** | 27 | 16 | 6 | 3 | 2 | | | **69** |
| -7 | | | -16 | -11 | -5 | -10 | -33 | -48 | -41 | **-23** | -10 | -7 | 0 | 6 | 10 | 4 | 0 | 10 | 24 | **33** | 31 | 22 | 13 | 4 | 2 | 3 | | | **73** |
| -6 | | | -16 | -21 | -15 | -10 | -30 | -50 | -44 | **-22** | -5 | -2 | 2 | 3 | 0 | -1 | 7 | 21 | 32 | **33** | 26 | 16 | 9 | 7 | 11 | 15 | | | **80** |
| -5 | | | -12 | -23 | -20 | -16 | -27 | -38 | -35 | -29 | **-23** | -11 | 3 | 7 | 6 | 4 | 9 | 29 | **49** | 40 | 7 | -4 | 14 | 25 | 15 | 2 | | | **79** |
| -4 | | | -11 | -6 | -8 | -26 | -44 | -44 | -29 | -25 | **-25** | -10 | 5 | 8 | 6 | 11 | 23 | 39 | **51** | 35 | 1 | -8 | 5 | 15 | 10 | 7 | | | **81** |
| -3 | | | -1 | -5 | -13 | -31 | -56 | -60 | **-40** | -16 | -2 | -2 | 0 | 5 | 12 | 16 | 21 | 33 | **40** | 25 | 6 | 27 | 95 | 100 | 28 | -43 | | | **85** |
| -2 | | | -11 | -15 | -28 | -41 | -48 | -43 | **-31** | -17 | -6 | -2 | 2 | 4 | 3 | 16 | 40 | **67** | 61 | 19 | 50 | 190 | 282 | 148 | -108 | -226 | | | **95** |
| -1 | | | -5 | -3 | -11 | -22 | -39 | -48 | -43 | **-27** | -11 | -6 | 0 | 4 | -4 | -8 | 15 | 60 | **87** | 47 | 32 | 177 | 337 | 284 | -23 | -280 | | | **104** |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0** | | | -21 | -23 | -22 | -29 | -35 | -36 | -30 | **-22** | -13 | -1 | 4 | 1 | 0 | 15 | 38 | **57** | 51 | 39 | 99 | 227 | 275 | 106 | -139 | -229 | | | **101** |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | -6 | -10 | -12 | -21 | -40 | -50 | -42 | **-26** | -15 | -11 | -5 | 0 | 1 | 5 | 28 | 64 | **86** | 70 | 26 | -1 | 14 | 24 | 11 | -2 | | | **100** |
| 2 | | | 2 | -6 | -20 | -29 | -31 | -31 | -30 | **-26** | -18 | -12 | -7 | -4 | -7 | 3 | 39 | 80 | **87** | 50 | 14 | 9 | 13 | 10 | 4 | 12 | | | **106** |
| 3 | | | 0 | -9 | -23 | -32 | -40 | -40 | -31 | **-18** | -12 | -17 | -15 | -5 | 2 | 4 | 19 | 51 | **73** | 64 | 47 | 41 | 30 | 12 | 5 | 16 | | | **111** |
| 4 | | | -6 | -14 | -18 | -25 | -41 | -42 | **-28** | -15 | -11 | -16 | -12 | -1 | 2 | -1 | 19 | 63 | **86** | 65 | 36 | 26 | 25 | 16 | 11 | 10 | | | **112** |
| 5 | | | 5 | -6 | -22 | -35 | -43 | -42 | -31 | **-20** | -14 | -11 | -9 | -7 | -12 | -13 | -3 | 21 | 54 | **71** | 66 | 50 | 36 | 29 | 27 | 27 | | | **118** |
| 6 | | | -2 | -3 | -13 | -25 | -37 | -43 | -38 | **-26** | -13 | -6 | -5 | -6 | -3 | 0 | 2 | 11 | 40 | 73 | **81** | 62 | 38 | 29 | 28 | 26 | | | **120** |
| 7 | | | -4 | 0 | -1 | -7 | -29 | -49 | -48 | -32 | **-18** | -14 | -10 | -5 | -2 | -1 | -2 | -1 | 12 | 37 | 68 | **86** | 87 | 63 | 37 | 27 | | | **123** |
| 8 | | | -11 | -14 | -14 | -10 | -23 | -42 | -44 | -31 | -21 | **-18** | -11 | 0 | 6 | 5 | -1 | -6 | 4 | 27 | 58 | 79 | **96** | 92 | 68 | 37 | | | **131** |
| 9 | | | -2 | -10 | -16 | -17 | -17 | -22 | -26 | -30 | -32 | -26 | **-18** | -11 | -2 | 6 | 3 | -8 | -10 | 4 | 24 | 44 | 73 | **95** | 94 | 64 | | | **120** |
| 10 | | | 5 | -1 | -10 | -18 | -21 | -24 | -28 | -32 | -30 | **-19** | -13 | -13 | -7 | 5 | 7 | -5 | -13 | -8 | 3 | 22 | 61 | 89 | **85** | 53 | | | **95** |
| 11 | | | 4 | 2 | -4 | -11 | -13 | -14 | -19 | -27 | -31 | -26 | **-22** | -19 | -8 | 3 | 2 | -9 | -8 | 4 | 4 | -4 | 3 | 20 | **26** | 20 | | | **68** |
| 12 | | | -12 | -19 | -21 | -16 | -11 | -10 | -14 | -20 | -25 | -28 | -26 | -17 | -8 | -2 | 1 | -1 | -2 | -1 | 4 | 10 | 16 | 6 | -10 | -7 | | | **54** |
| 13 | | | 3 | -1 | -5 | -8 | -13 | -18 | -18 | -18 | -21 | -23 | -22 | -22 | -29 | -29 | -19 | -7 | -2 | -1 | 8 | 14 | 8 | 1 | -1 | -3 | | | **49** |
| 14 | | | 0 | -1 | -1 | 0 | -6 | -13 | -15 | -18 | -25 | -31 | -26 | -16 | -17 | -20 | -19 | -16 | -14 | -8 | 8 | 18 | 6 | -16 | -20 | -9 | | | **39** |
| 15 | | | -4 | 0 | 6 | 9 | -2 | -15 | -17 | -14 | -15 | -20 | -17 | -11 | -13 | -17 | -19 | -21 | -21 | -13 | 5 | 16 | 1 | -21 | -28 | -20 | | | **43** |
| 16 | | | -1 | -1 | 3 | 7 | 0 | -11 | -16 | -13 | -10 | -11 | -11 | -13 | -18 | -18 | -13 | -12 | -15 | -14 | -2 | 6 | 1 | -9 | -16 | -17 | | | **52** |
| 17 | | | 2 | 0 | -1 | -2 | -2 | -2 | -4 | -9 | -12 | -8 | -3 | -4 | -10 | -11 | -6 | -6 | -12 | -14 | -7 | -1 | -3 | -13 | -17 | -11 | | | **63** |
| 18 | | | 5 | -1 | -3 | -4 | -8 | -9 | -4 | -1 | -4 | -8 | -5 | 0 | -5 | -9 | -8 | -8 | -12 | -14 | -5 | 2 | -3 | -13 | -12 | -4 | | | **68** |
| 19 | | | 5 | 0 | 0 | 0 | -4 | -5 | -1 | 1 | -3 | -9 | -9 | -6 | -5 | -4 | -2 | -6 | -12 | -11 | -1 | 7 | 0 | -6 | -5 | -5 | | | **77** |
| 20 | | | 3 | 2 | 6 | 8 | 1 | -6 | -8 | -7 | -4 | -2 | 1 | 0 | -8 | -12 | -9 | -6 | -7 | -7 | 0 | 5 | 1 | -8 | -11 | -8 | | | **77** |

**Table 9 - Pixel difference values around centre - sliced in x direction going left**

| user y | 3[299] | | | | | | | | | | | | 4[309] | | | | | 5[314] | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| -20 | **54** | | | | -3 | -8 | -13 | -8 | -11 | -21 | -23 | -12 | -5 | -12 | -20 | -21 | -18 | -16 | -11 | -9 | -13 | -14 | -9 | 93 | 192 | 193 | 99 | 0 | 0 | | |
| -19 | **47** | | | | -12 | -5 | -2 | -2 | -11 | -17 | -17 | -12 | -12 | -22 | -29 | -27 | -19 | -16 | -14 | -12 | -12 | -11 | -8 | 93 | 193 | 194 | 99 | 0 | 0 | | |
| -18 | **44** | | | | -13 | -6 | -2 | -2 | -7 | -11 | -11 | -12 | -17 | -26 | -31 | -28 | -20 | -19 | -20 | -17 | -13 | -11 | -10 | 92 | 195 | 196 | 99 | 0 | 0 | | |
| -17 | **44** | | | | -11 | -6 | -1 | 1 | -1 | -5 | -9 | -12 | -19 | -26 | -32 | -33 | -28 | -25 | -21 | -18 | -16 | -17 | -15 | 90 | 196 | 198 | 99 | 0 | 0 | | |
| -16 | **44** | | | | -12 | -10 | -6 | 3 | 5 | -2 | -9 | -9 | -14 | -23 | -33 | -36 | -33 | -31 | -22 | -18 | -19 | -22 | -17 | 91 | 196 | 197 | 98 | 0 | 0 | | |
| -15 | **46** | | | | -9 | -13 | -11 | -2 | 4 | -1 | -6 | -5 | -9 | -16 | -30 | -39 | -41 | -39 | -26 | -19 | -15 | -16 | -10 | 93 | 191 | 191 | 94 | 0 | 0 | | |
| -14 | **46** | | | | -6 | -7 | -4 | 1 | 3 | 1 | 0 | 0 | -10 | -23 | -38 | -42 | -41 | -40 | -29 | -22 | -13 | -9 | -3 | 93 | 183 | 183 | 91 | 0 | 0 | | |
| -13 | **43** | | | | -13 | -5 | 2 | 0 | -1 | 5 | 9 | 4 | -8 | -20 | -36 | -42 | -41 | -39 | -32 | -25 | -10 | -2 | 1 | 90 | 175 | 176 | 88 | 0 | 0 | | |
| -12 | **47** | | | | -4 | -8 | -10 | -13 | -9 | -4 | -5 | 2 | 13 | 12 | -13 | -39 | -51 | -46 | -37 | -27 | -14 | -13 | -14 | 80 | 179 | 185 | 93 | 0 | 0 | | |
| -11 | **60** | | | | 11 | 8 | -1 | -6 | -8 | -8 | -5 | 3 | 10 | 6 | -17 | -38 | -51 | -50 | -40 | -27 | -14 | -11 | -8 | 87 | 181 | 182 | 91 | 0 | 0 | | |
| -10 | **67** | | | | 21 | **39** | 37 | 26 | 7 | 4 | 2 | 0 | 0 | -2 | -16 | -32 | -50 | -53 | -43 | -28 | -17 | -18 | -12 | 88 | 185 | 184 | 90 | 0 | 0 | | |
| -9 | **80** | | | | 22 | 47 | **59** | 52 | 29 | 20 | 9 | 2 | 0 | 3 | -6 | -24 | -47 | -55 | -45 | -30 | -24 | -27 | -23 | 82 | 185 | 187 | 94 | 0 | 0 | | |
| -8 | **78** | | | | 18 | 19 | 37 | 53 | **54** | 43 | 18 | 7 | 4 | 10 | 1 | -21 | -44 | -52 | -44 | -32 | -28 | -30 | -24 | 81 | 184 | 186 | 94 | 0 | 0 | | |
| -7 | **88** | | | | -2 | -2 | 18 | 39 | 58 | **67** | 47 | 24 | 4 | -1 | -10 | -24 | -38 | -44 | -41 | -33 | -29 | -31 | -22 | 86 | 189 | 188 | 93 | 0 | 0 | | |
| -6 | **158** | | | | -17 | 2 | 12 | 15 | 43 | 79 | **83** | 52 | 11 | -7 | -17 | -23 | -34 | -39 | -38 | -33 | -33 | -32 | -22 | 88 | 193 | 191 | 96 | 0 | 0 | | |
| -5 | **227** | | | | -11 | 2 | 3 | 1 | 29 | 66 | **91** | 74 | 35 | 6 | -16 | -20 | -30 | -35 | -38 | -36 | -33 | -32 | -25 | 84 | 192 | 194 | 98 | 0 | 0 | | |
| -4 | **214** | | | | -7 | -16 | -18 | -31 | -12 | 35 | 93 | **107** | 83 | 53 | 19 | -3 | -36 | -48 | -42 | -32 | -30 | -26 | -13 | 85 | 175 | 173 | 87 | 0 | 0 | | |
| -3 | **138** | | | | -23 | -20 | -16 | -25 | -16 | 19 | 83 | **130** | 109 | 63 | 17 | -7 | -29 | -46 | -48 | -42 | -26 | -9 | -9 | 70 | 160 | 170 | 89 | 0 | 0 | | |
| -2 | **108** | | | | -28 | -22 | -18 | -26 | -19 | 12 | 85 | **142** | 127 | 68 | 14 | -3 | -23 | -47 | -55 | -44 | -18 | -3 | -6 | 70 | 156 | 167 | 86 | 0 | 0 | | |
| -1 | **104** | | | | -17 | -17 | -24 | -31 | -12 | 25 | 90 | **127** | 118 | 76 | 31 | 7 | -27 | -50 | -53 | -39 | -23 | -16 | -11 | 77 | 163 | 166 | 84 | 0 | 0 | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0** | **101** | | | | -24 | -21 | -20 | -24 | -13 | 28 | 91 | **129** | 112 | 75 | 40 | 8 | -27 | -49 | -50 | -39 | -30 | -20 | -12 | 76 | 165 | 166 | 85 | 0 | 0 | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | **105** | | | | -29 | -25 | -21 | -24 | -22 | 19 | 97 | **148** | 128 | 72 | 28 | 3 | -23 | -44 | -45 | -37 | -26 | -18 | -14 | 71 | 159 | 167 | 86 | 0 | 0 | | |
| 2 | **99** | | | | -18 | -14 | -23 | -30 | -19 | 33 | 121 | **161** | 130 | 60 | 14 | -1 | -22 | -40 | -42 | -34 | -25 | -26 | -22 | 70 | 161 | 169 | 85 | 0 | 0 | | |
| 3 | **102** | | | | -15 | -10 | -22 | -33 | -12 | 64 | 146 | **162** | 107 | 38 | 4 | -11 | -27 | -37 | -37 | -29 | -24 | -26 | -24 | 69 | 165 | 169 | 85 | 0 | 0 | | |
| 4 | **99** | | | | -23 | -22 | -24 | -6 | 40 | 111 | **147** | 119 | 57 | 13 | 1 | -8 | -18 | -31 | -36 | -33 | -27 | -21 | -12 | 77 | 162 | 159 | 79 | 0 | 0 | | |
| 5 | **105** | | | | -26 | -27 | -9 | 38 | 93 | **120** | 112 | 75 | 34 | 11 | 0 | -7 | -21 | -35 | -39 | -37 | -28 | -17 | -10 | 75 | 159 | 161 | 82 | 0 | 0 | | |
| 6 | **103** | | | | -29 | -1 | 40 | 88 | **121** | 105 | 67 | 26 | 10 | 10 | 1 | -14 | -35 | -45 | -37 | -27 | -21 | -16 | -15 | 71 | 159 | 165 | 86 | 0 | 0 | | |
| 7 | **101** | | | | 6 | 54 | 86 | **90** | 79 | 55 | 29 | 2 | 0 | 10 | 6 | -16 | -39 | -48 | -38 | -23 | -17 | -16 | -15 | 72 | 161 | 164 | 84 | 0 | 0 | | |
| 8 | **95** | | | | 43 | 70 | **82** | 58 | 28 | 10 | 1 | -5 | -1 | 8 | 10 | -7 | -30 | -46 | -45 | -31 | -21 | -17 | -12 | 75 | 162 | 162 | 81 | 0 | 0 | | |
| 9 | **77** | | | | 36 | 30 | **35** | 26 | 17 | 7 | -3 | 0 | 3 | 5 | 1 | -11 | -33 | -46 | -43 | -28 | -19 | -18 | -15 | 71 | 160 | 162 | 81 | 0 | 0 | | |
| 10 | **54** | | | | **15** | 2 | 0 | 6 | 14 | 10 | 2 | 3 | 5 | 6 | 0 | -17 | -47 | -58 | -44 | -20 | -10 | -11 | -13 | 69 | 157 | 161 | 82 | 0 | 0 | | |
| 11 | **44** | | | | 9 | 10 | -3 | -7 | -5 | -1 | 2 | 0 | 4 | 10 | 9 | -15 | -51 | -63 | -46 | -22 | -10 | -1 | 2 | 76 | 152 | 154 | 80 | 0 | 0 | | |
| 12 | **45** | | | | 7 | 7 | -1 | -4 | -1 | 4 | 1 | 3 | 3 | 1 | -8 | -27 | -42 | -48 | -35 | -21 | -15 | -8 | -2 | 78 | 153 | 153 | 79 | 0 | 0 | | |
| 13 | **44** | | | | 3 | 6 | 5 | -1 | -1 | -2 | -4 | 3 | 3 | -4 | -17 | -31 | -38 | -39 | -27 | -15 | -12 | -16 | -10 | 78 | 160 | 159 | 77 | 0 | 0 | | |
| 14 | **42** | | | | 1 | 4 | 6 | -1 | -7 | -10 | -8 | 5 | 7 | -6 | -26 | -38 | -38 | -35 | -25 | -14 | -8 | -14 | -17 | 72 | 163 | 169 | 84 | 0 | 0 | | |
| 15 | **40** | | | | 0 | -3 | -2 | -2 | -7 | -7 | -2 | 9 | 5 | -18 | -40 | -44 | -34 | -29 | -20 | -10 | -5 | -7 | -12 | 72 | 158 | 164 | 85 | 0 | 0 | | |
| 16 | **43** | | | | 3 | -5 | -8 | -3 | -1 | 2 | 4 | 4 | -8 | -33 | -47 | -41 | -27 | -20 | -12 | -1 | -2 | -7 | -4 | 79 | 155 | 150 | 75 | 0 | 0 | | |
| 17 | **45** | | | | 5 | 2 | -1 | -1 | 2 | 2 | -4 | -16 | -29 | -37 | -38 | -29 | -21 | -15 | -8 | 0 | 1 | -6 | -6 | 76 | 155 | 153 | 75 | 0 | 0 | | |
| 18 | **44** | | | | 8 | 4 | -3 | -8 | -8 | -6 | -11 | -24 | -34 | -33 | -25 | -17 | -17 | -17 | -12 | -3 | 5 | -1 | -11 | 68 | 155 | 163 | 82 | 0 | 0 | | |
| 19 | **43** | | | | 7 | -6 | -16 | -14 | -9 | -5 | -5 | -15 | -30 | -37 | -28 | -15 | -11 | -13 | -8 | 2 | 7 | -3 | -14 | 67 | 157 | 164 | 82 | 0 | 0 | | |
| 20 | **43** | | | | 5 | -4 | -15 | -17 | -15 | -11 | -13 | -15 | -19 | -25 | -25 | -17 | -9 | -7 | -4 | -1 | -3 | -8 | -9 | 74 | 158 | 162 | 82 | 0 | 0 | | |

**Table 10 - Pixel difference values around centre - sliced in y direction going down**

| user y | | | | | | | | | 1[284] | | | | 2[288] | | | | | | | | | | | | 3[299] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | -23 | -22 | -21 | -20 | -19 | -18 | -17 | -16 | -15 | -14 | -13 | -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | | **0** |
| -20 | | | -1 | -4 | -10 | -17 | -24 | -25 | -19 | -16 | -11 | -7 | -1 | -6 | -15 | -18 | -11 | 4 | 0 | -10 | -9 | | | | **54** |
| -19 | | | 4 | -9 | -19 | -22 | -19 | -20 | -16 | -16 | -12 | -9 | -3 | -4 | -12 | -14 | -9 | 4 | 1 | -9 | -10 | | | | **47** |
| -18 | | | -3 | -12 | -20 | -23 | -21 | -20 | -17 | -16 | -17 | -20 | -12 | -1 | 1 | -2 | -5 | -3 | -4 | -7 | -4 | | | | **44** |
| -17 | | | -12 | -15 | -19 | -23 | -24 | -25 | -25 | -21 | -15 | -12 | -8 | -2 | 0 | -3 | -10 | -14 | -8 | 1 | 9 | | | | **44** |
| -16 | | | -13 | -14 | -19 | -24 | -28 | -30 | -29 | -20 | -7 | -2 | -7 | -12 | -8 | -2 | -2 | -4 | -3 | 0 | 3 | | | | **44** |
| -15 | | | -12 | -11 | -15 | -20 | -26 | -31 | -29 | -18 | -9 | -8 | -13 | -13 | -2 | 6 | 8 | 5 | -3 | -7 | -4 | | | | **46** |
| -14 | | | -11 | -11 | -17 | -24 | -30 | -33 | -28 | -16 | -6 | -4 | -7 | -10 | -3 | 3 | 3 | -2 | -6 | -2 | 3 | | | | **46** |
| -13 | | | -9 | -14 | -20 | -28 | -34 | -35 | -28 | -18 | -3 | 5 | 4 | -7 | -9 | -2 | 0 | -1 | -1 | 2 | 2 | | | | **43** |
| -12 | | | -13 | -21 | -28 | -28 | -31 | -33 | -30 | -23 | -9 | 3 | 5 | 3 | 4 | 2 | -2 | -4 | 3 | 0 | -16 | | | | **47** |
| -11 | | | -17 | -24 | -25 | -26 | -36 | -39 | -32 | -14 | 4 | 2 | -6 | -9 | -4 | 2 | 1 | 6 | 12 | **20** | 8 | | | | **60** |
| -10 | | | -14 | -24 | -25 | -27 | -40 | -43 | -31 | -11 | 3 | -2 | -5 | -1 | 7 | 14 | 13 | 16 | 18 | **25** | 20 | | | | **67** |
| -9 | | | -13 | -16 | -19 | -28 | -38 | -38 | -28 | -18 | -10 | -5 | 0 | 8 | 20 | 28 | **32** | 31 | 25 | 27 | 25 | | | | **80** |
| -8 | | | -10 | -11 | -19 | -28 | -34 | -36 | -31 | -24 | -13 | -1 | 6 | 12 | 21 | **29** | 34 | 34 | 25 | 13 | 15 | | | | **78** |
| -7 | | | -11 | -14 | -22 | -27 | -31 | -38 | -36 | -19 | 0 | 5 | 10 | 22 | **33** | 28 | 11 | 2 | -1 | 3 | 4 | | | | **88** |
| -6 | | | -24 | -25 | -27 | -26 | -26 | -28 | -26 | -11 | 7 | 12 | 16 | **27** | 35 | 25 | 2 | -12 | -6 | 72 | 115 | | | | **158** |
| -5 | | | -28 | -34 | -33 | -25 | -22 | -20 | -18 | -8 | 14 | **25** | 19 | 10 | 9 | 15 | 17 | 16 | 37 | 173 | 304 | | | | **227** |
| -4 | | | -24 | -24 | -24 | -26 | -31 | -32 | -26 | -15 | 9 | 32 | **35** | 23 | 14 | 13 | 15 | 10 | 65 | 197 | 315 | | | | **214** |
| -3 | | | -16 | -19 | -23 | -28 | -33 | -34 | -28 | -12 | 13 | 30 | **31** | 21 | 15 | 15 | 15 | 10 | 32 | 94 | 196 | | | | **138** |
| -2 | | | -7 | -12 | -22 | -32 | -36 | -35 | -29 | -10 | 16 | 30 | **30** | 20 | 15 | 18 | 18 | 14 | 24 | 45 | 67 | | | | **108** |
| -1 | | | -7 | -10 | -24 | -37 | -38 | -31 | -25 | -8 | 18 | 30 | **31** | 23 | 17 | 20 | 18 | 16 | 24 | 39 | 37 | | | | **104** |
| | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0** | | | -8 | -10 | -24 | -34 | -38 | -28 | -22 | -10 | 12 | 26 | **33** | 27 | 20 | 22 | 21 | 18 | 13 | 21 | 39 | | | | **101** |
| | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | -6 | -10 | -24 | -32 | -35 | -29 | -24 | -15 | 7 | 25 | **32** | 28 | 23 | 25 | 24 | 20 | 17 | 20 | 23 | | | | **105** |
| 2 | | | -13 | -17 | -25 | -27 | -29 | -34 | -31 | -21 | 5 | 28 | **36** | 32 | 26 | 27 | 28 | 22 | 28 | 26 | 19 | | | | **99** |
| 3 | | | -24 | -25 | -25 | -23 | -23 | -37 | -35 | -23 | 7 | 33 | **38** | 33 | 27 | 27 | 28 | 21 | 25 | 20 | 12 | | | | **102** |
| 4 | | | -20 | -15 | -19 | -27 | -41 | -47 | -40 | -23 | 6 | 27 | **43** | 40 | 35 | 31 | 29 | 26 | 16 | 9 | 11 | | | | **99** |
| 5 | | | -24 | -14 | -14 | -24 | -39 | -40 | -34 | -26 | -5 | 18 | 41 | **43** | 35 | 26 | 23 | 24 | 18 | 15 | 21 | | | | **105** |
| 6 | | | -16 | -13 | -21 | -29 | -34 | -32 | -30 | -31 | -15 | 10 | 38 | **45** | 42 | 35 | 29 | 26 | 25 | 29 | 28 | | | | **103** |
| 7 | | | -14 | -15 | -23 | -25 | -26 | -27 | -31 | -35 | -26 | -9 | 14 | 30 | **44** | 47 | 43 | 35 | 31 | 31 | 31 | | | | **101** |
| 8 | | | -24 | -18 | -14 | -14 | -22 | -26 | -32 | -35 | -31 | -20 | -2 | 14 | 33 | **40** | 41 | 34 | 38 | 43 | 44 | | | | **95** |
| 9 | | | -21 | -16 | -10 | -8 | -19 | -31 | -36 | -32 | -23 | -15 | -4 | 4 | 12 | 15 | 19 | **24** | 39 | 45 | 40 | | | | **77** |
| 10 | | | -17 | -17 | -18 | -17 | -19 | -29 | -33 | -30 | -21 | -15 | -7 | -1 | 3 | 4 | 6 | 11 | **15** | 15 | 14 | | | | **54** |
| 11 | | | -27 | -24 | -20 | -18 | -18 | -26 | -31 | -29 | -21 | -18 | -12 | -5 | 3 | 8 | 6 | 4 | 2 | 6 | 7 | | | | **44** |
| 12 | | | -27 | -18 | -14 | -12 | -16 | -25 | -31 | -30 | -26 | -19 | -14 | -6 | 0 | 4 | 4 | -2 | 1 | 1 | 3 | | | | **45** |
| 13 | | | -31 | -29 | -18 | -14 | -16 | -21 | -26 | -28 | -27 | -19 | -13 | -8 | -5 | -3 | 0 | 0 | 1 | -6 | -4 | | | | **44** |
| 14 | | | -29 | -29 | -16 | -11 | -13 | -17 | -22 | -26 | -30 | -27 | -19 | -10 | -4 | -5 | -6 | -8 | 1 | 4 | 5 | | | | **42** |
| 15 | | | -25 | -17 | -7 | -5 | -10 | -17 | -23 | -23 | -28 | -32 | -28 | -16 | -2 | 0 | -4 | -12 | -5 | 7 | 11 | | | | **40** |
| 16 | | | -22 | -10 | -5 | -1 | -7 | -19 | -25 | -21 | -23 | -31 | -32 | -22 | -7 | 1 | 3 | -4 | -12 | -10 | -3 | | | | **43** |
| 17 | | | -22 | -12 | -7 | 0 | -8 | -22 | -26 | -20 | -23 | -33 | -33 | -21 | -6 | 2 | 3 | -4 | -13 | -10 | -2 | | | | **45** |
| 18 | | | -21 | -16 | -13 | -8 | -11 | -16 | -20 | -22 | -31 | -36 | -29 | -14 | -5 | -3 | -4 | -8 | -10 | -1 | 9 | | | | **44** |
| 19 | | | -17 | -14 | -12 | -12 | -13 | -11 | -16 | -25 | -36 | -33 | -22 | -13 | -13 | -13 | -7 | -1 | -4 | -2 | 7 | | | | **43** |
| 20 | | | -22 | -18 | -13 | -8 | -11 | -9 | -6 | -8 | -16 | -26 | -29 | -27 | -22 | -13 | -4 | 1 | -1 | -6 | -11 | | | | **43** |

**Table 11 - Pixel difference values around centre - sliced in y direction going up**

From looking at the pixel difference data in tables 8 to11 it can be seen that, in general, the difference is increasing and is largest at the pupil/iris boundary. After this the difference tends to decrease and swings negative and becomes a large negative value. A typical plot of this is shown in figure 11. Notice that this was carefully chosen (right hand side at offset −6) as most data has some retinal reflection effects that need to be processed as will be seen later.

From the figure it is clear that the difference data can be used as a start in both finding the boundaries and in verifying that the slice is actually "eye like".

For the pupil/iris boundary the maximum difference (which must be above the pupil/iris threshold value – see later description) before the difference starts a downward trend is used.

For the iris/sclera the first negative difference below the iris/sclera threshold is used. Note that the minima is not used as it is uncertain how the eye slice will respond after the iris/sclera boundary and experience of the data shows that this method will produce better results.

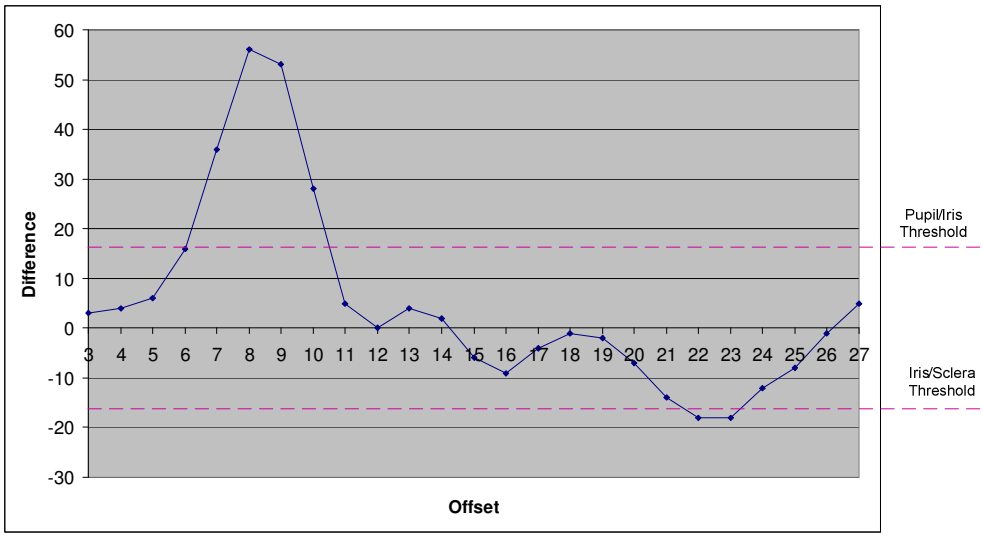



**Figure 11 - Differences as offset moves out from pupil to sclera**

## *3.5 Proposed slice verifier*

The slice verifier will be concerned only with points from the pupil outwards and it is left to other higher level routines, which are developed later, to spilt the data (as shown in figure 12) and place it into an array.

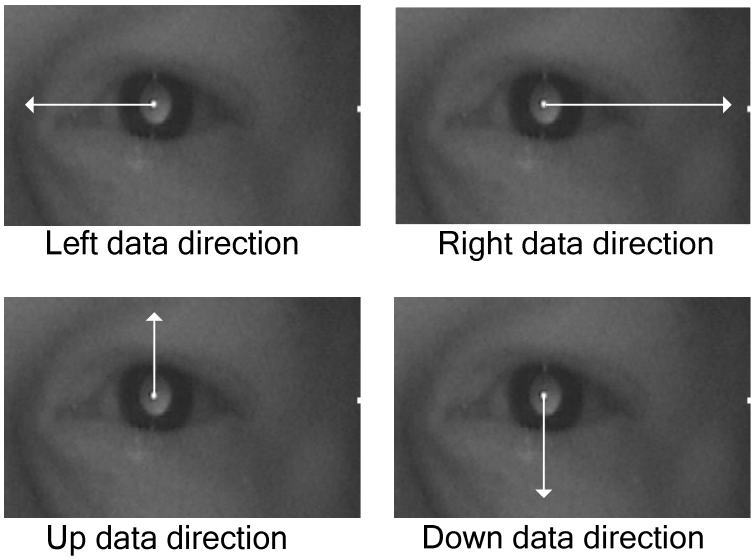



**Figure 12 - Slicing the eye**

This slice verifier routine is responsible for finding the boundaries in relation to the array data. If no boundaries (within its guideline parameters) are found then false is returned to indicate that the given slice does not follow the characteristics of our eye detection. If boundaries are found then these are returned to the caller.

Other routines will be responsible for correctly formatting the data and interpreting the results.

As a start point for the routine all of the IR front facing images were checked using difference data as above and analysed, leading to the following observations.

- The high reflectivity from the pupil leads to values that can cause problems where initial large negative values and large positive values occur, all of which are not really representative of the true data. This is even more so as the final

algorithm will start on the retinal reflection hot spot as this will be the bright value that is used to start a test for an eye.

One method to deal with this would be to average out the "hot spot" (retinal reflection threshold) values in both x & y directions but this leads to some loss of data.

Instead the image array can be parsed to find the position where there is a value less than the hot spot threshold that is immediately followed by another value that is less than the hot spot threshold. These two values are then added and divided by two (thus taking the average) and replace the values before these positions with this average instead. Hence, *on a slice by slice basis only,* the hot spot value has been averaged out, where there is no hot spot effect the underlying data is left alone.

- The iris pixel value will be less than the pupil value by some factor and it has been empirically chosen that it must be at least 15% less than the initial start value (which is in the pupil). In the software this is actually doubled as two values either side of a point are being used and it is quicker to do this than divide by two every time. Note also that this threshold will be positive due to the order in which the additions and subtractions are performed.

  This, and all other empirical values, is held in a variable and all located and set together in a constructor to allow easy change at any stage.

- Similarly the sclera value is chosen such that it has to be at least 15% greater than the *pixel* value (not difference) that was found when getting the pupil/iris boundary. This threshold will be negative due to the additions and subtractions performed.

- Whilst looking for the pupil/iris boundary only 6 negative values are allowed, if this is exceeded then the routine reports failure to indicate that this slice is not, by our definitions, an eye slice.

- Once the best pupil/iris boundary is found only negative values (or small positive ones) should be found, if not return false is returned. When it is under the desired negative threshold then the pupil/iris and iris/sclera boundaries have been found for that data slice.

The actual flowcharts used are shown in figures 13 to 16.

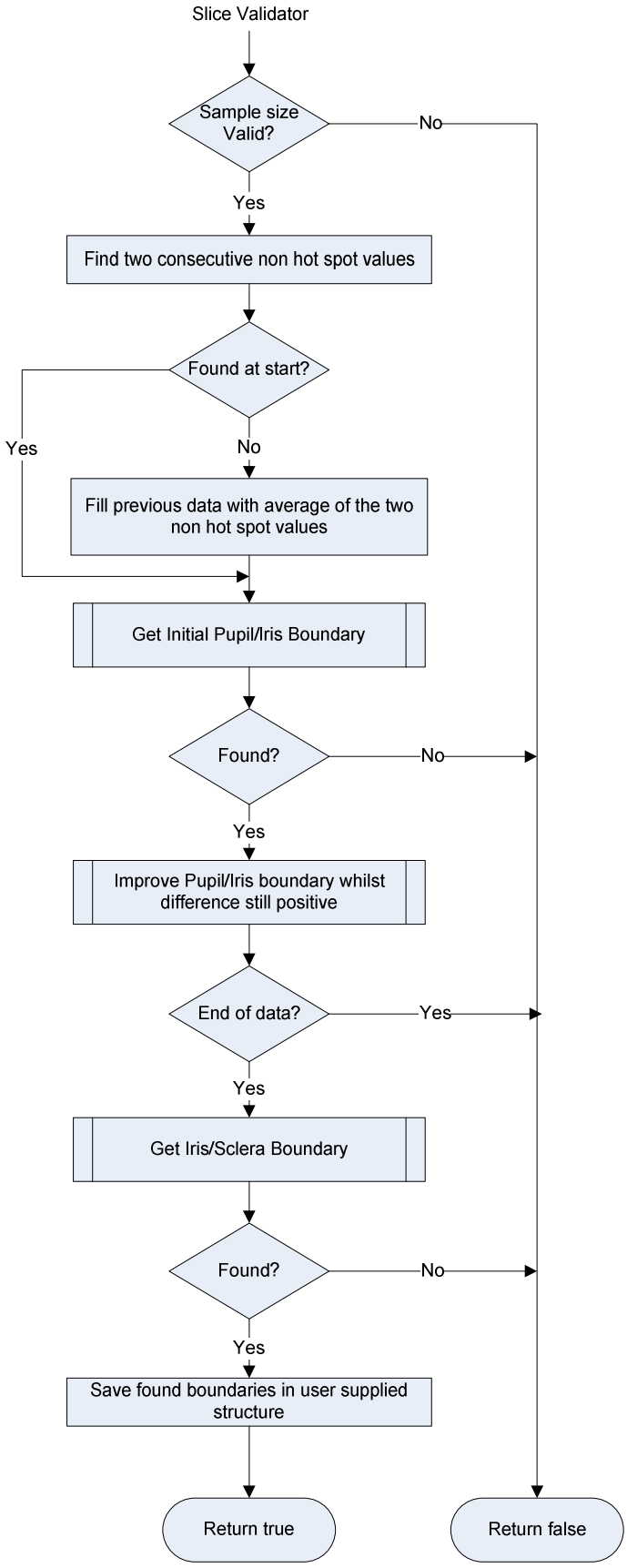



**Figure 13 - Flowchart of Eye Slice validator routine**

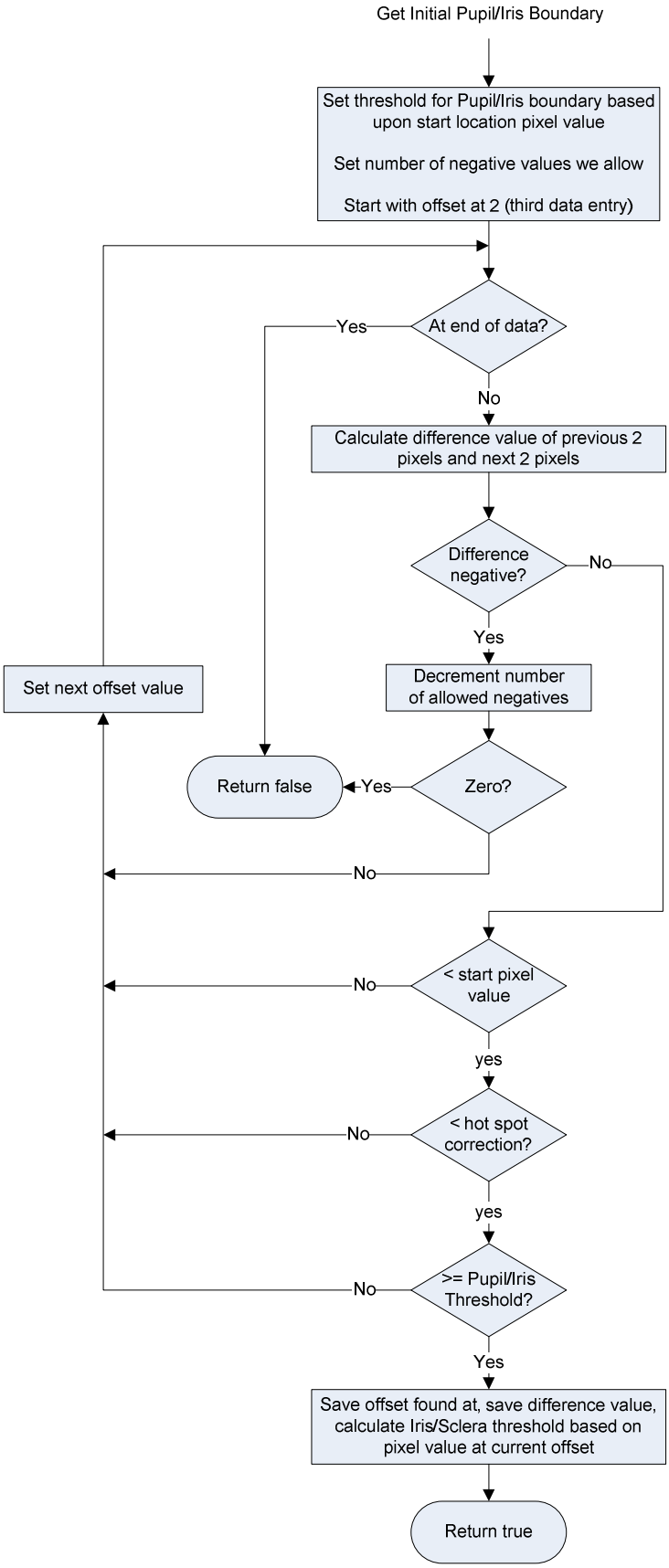



**Figure 14 - Flowchart of initial Pupil/Iris boundary routine**

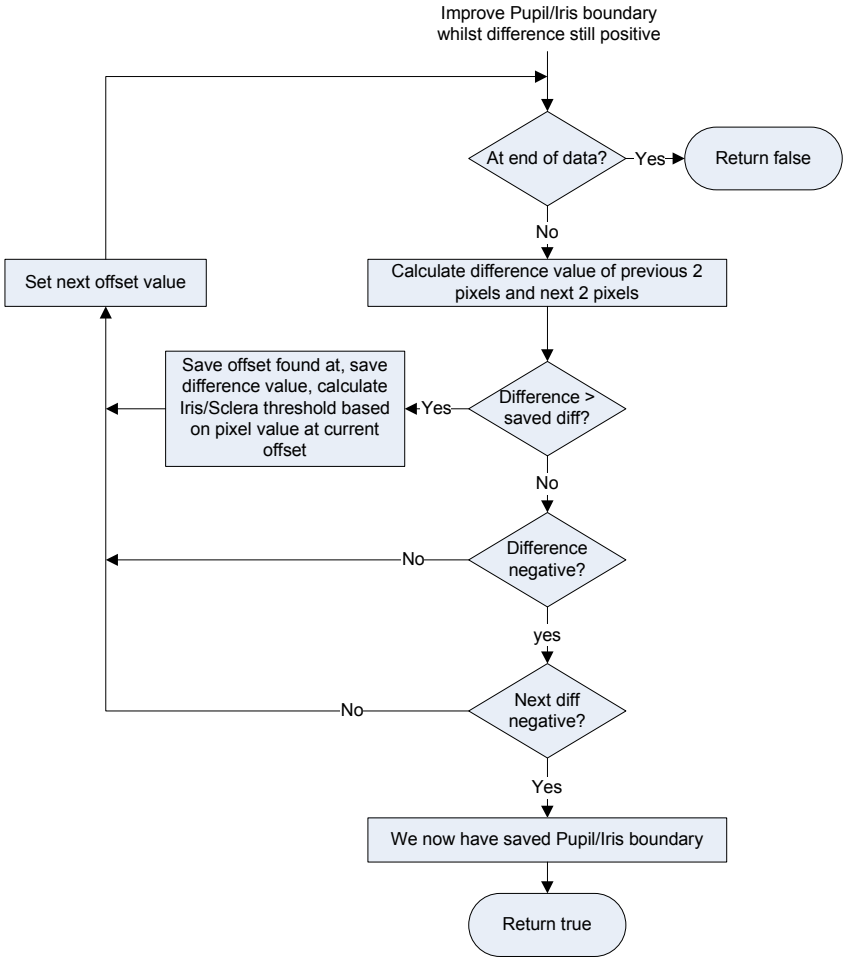



**Figure 15 - Flowchart of improving Pupil/Iris boundary routine**

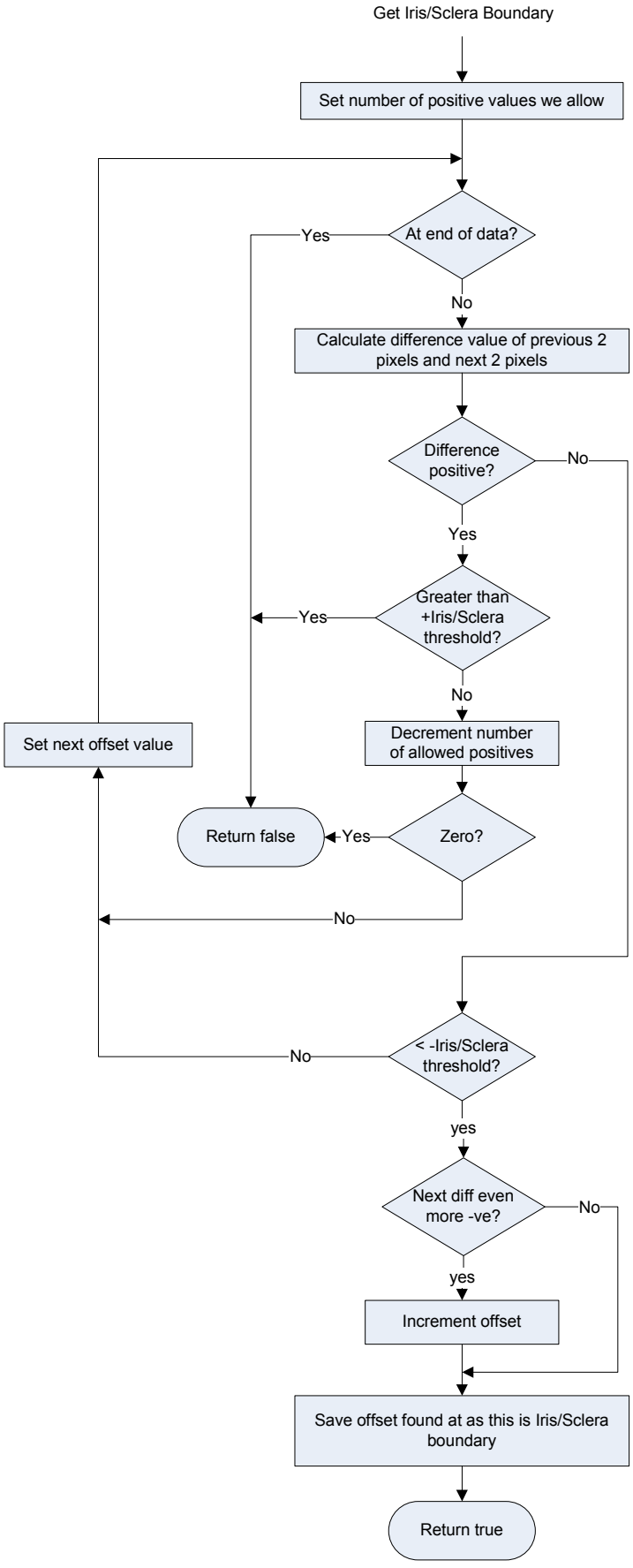



**Figure 16 - Flowchart of Iris/Sclera boundary routine**

## *3.6 Summary*

The actual image data was analysed and a method found to reliably determine the iris/pupil and pupil/iris boundaries using the difference values between the two pixels on one side compared with the two pixels on the other side.

A flowchart was then developed that will form the basis of the eye slice verifier.

# Chapter 4
# Implementation and results

## 4.1 Introduction

This chapter concentrates on how other routines are built on top of the slice verifier of the previous chapter.

The routines are then tested against where the user would place the data and using this information the routines are to be refined to improve detection accuracy and eliminating reflections from glasses being treated as eyes.

Finally the whole eye detector class will be put to test on a live image to obtain eye motion vectors.

## 4.2 Program implementation

A test program was written[6] that took the user data centred at $x_c$ ,$y_c$ and generated slices to feed to the slice validator function both in x slices and y slices through the eye for +/- 50 lines from the centre.

A routine was provided that would take an incoming array of size 2*iSize+1 (where iSize is the user supplied size of samples to take either side of centre values and is typically set to 50 for this application) and copy it into arrays of right hand (or bottom) values from iStart to 2*iStart and an array from iStart to 0 (thus turning what would have been left/up going operations into the same order as the right/bottom side which the slice validator expects).

The slice validator is then called for both these new arrays and false returned if either data set fails (as the whole slice must be valid to be a potential eye). If true is

[6] Appendix 5

returned for both arrays then the routine updates the boundary values that it has found into structures that are passed (as references) into this routine.

For slicing in the x direction, image pixels from $\{x_c - size , yp\}$ to $\{x_c + size, yp\}$ are copied from the image into an array that can be passed to the above routine. This is repeated as yp is varied from $y_c$ - size to $y_c$ + size to provide the output (minus the slices, for clarity, not found above and below consecutively) as summarised in tables 12 & 13. Note that the program "not found" is across all its columns as slices in both directions must be valid for the combined slice to be considered valid.

| | | Program decided Offsets from centre | | | |
|---|---|---|---|---|---|
| Slice number | | Sclera/Iris | Iris/Pupil | Pupil/Iris | Iris/Sclera |
| | | | | | |
| -10 | | 16 | 4 | 4 | 15 |
| -9 | | 15 | 7 | 5 | 15 |
| -8 | | 15 | 7 | 7 | 16 |
| -7 | | 16 | 9 | 8 | 15 |
| -6 | | not found | not found | not found | not found |
| -5 | | 19 | 4 | 9 | 16 |
| -4 | | not found | not found | not found | not found |
| -3 | | not found | not found | not found | not found |
| -2 | | not found | not found | not found | not found |
| -1 | | not found | not found | not found | not found |
| 0 | | not found | not found | not found | not found |
| 1 | | not found | not found | not found | not found |
| 2 | | not found | not found | not found | not found |
| 3 | | not found | not found | not found | not found |
| 4 | | 16 | 11 | 9 | 17 |
| 5 | | 15 | 10 | 8 | 17 |
| 6 | | 14 | 10 | 7 | 16 |
| 7 | | 14 | 8 | 6 | 16 |
| 8 | | 14 | 3 | 5 | 16 |
| 9 | | 14 | 4 | 3 | 16 |

**Table 12 - Program points for y axis sliced data**

| | User decided Offsets from centre | | | | Program decided Offsets from centre | | | |
|---|---|---|---|---|---|---|---|---|
| Slice number | Sclera/ Iris | Iris/ Pupil | Pupil/ Iris | Iris/ Sclera | Sclera/ Iris | Iris/ Pupil | Pupil/ Iris | Iris/ Sclera |
| -13 | n/a | n/a | 3 | 20 | | | | |
| -12 | n/a | n/a | 4 | 20 | not found | not found | not found | not found |
| -11 | 18 | 6 | 6 | 19 | 19 | 6 | 6 | 20 |
| -10 | 18 | 6 | 6 | 19 | 19 | 6 | 6 | 19 |
| -9 | 18 | 7 | 7 | 20 | 18 | 7 | 7 | 20 |
| -8 | 19 | 8 | 7 | 20 | 20 | 8 | 7 | 23 |
| -7 | 19 | 9 | 8 | 20 | 20 | 9 | 8 | 21 |
| -6 | 19 | 9 | 8 | 22 | not found | not found | not found | not found |
| -5 | 18 | 10 | 9 | 20 | 19 | 10 | 9 | 49 |
| -4 | 18 | 10 | 9 | 19 | 18 | 10 | 9 | 49 |
| -3 | 20 | 10 | 9 | 20 | 21 | 5 | 9 | 20 |
| -2 | 20 | 11 | 9 | 22 | not found | not found | not found | not found |
| -1 | 19 | 10 | 9 | 21 | 20 | 10 | 9 | 22 |
| 0 | 19 | 11 | 9 | 21 | 20 | 11 | 9 | 22 |
| 1 | 19 | 10 | 9 | 21 | 20 | 10 | 11 | 22 |
| 2 | 19 | 10 | 8 | 21 | not found | not found | not found | not found |
| 3 | 19 | 10 | 9 | 29 | 21 | 10 | 9 | 21 |
| 4 | 20 | 10 | 8 | 20 | 21 | 10 | 8 | 49 |
| 5 | 19 | 9 | 7 | 20 | not found | not found | not found | not found |
| 6 | 19 | 8 | 7 | 20 | not found | not found | not found | not found |
| 7 | 18 | 7 | 6 | 19 | not found | not found | not found | not found |
| 8 | 17 | 6 | 5 | 19 | not found | not found | not found | not found |
| 9 | 16 | 5 | 4 | 18 | not found | not found | not found | not found |
| 10 | 17 | 4 | 3 | 18 | 18 | 5 | 4 | 19 |
| 11 | 16 | 4 | n/a | n/a | not found | not found | not found | not found |

**Table 13 - Program vs User points for x axis sliced data**

## *4.3 Comparing user data with program obtained data*

Table 13 also compares where it was decided that a human would place the boundaries given the same data and there is a high degree of correspondence in the pupil/iris boundary.

Notice that the program iris/sclera boundary occurs mainly one position later than the human placing (looking at the raw data and not on screen image) and this is to be expected as the routine peeks at the next difference along and sees if it is greater (in a

negative sense) than the current one and uses that instead as tests show that this produces results that are more in line with the user (on screen) entered positions.

## *4.4 Programmatically obtaining multiple slices*

The previous manual program was altered to put together a higher level routine that, from a given $x_c$,$y_c$ position in the image, takes multiple slices and forms a queue of valid (at slice level) slices that we will be able to further validate.

These multiple slices are taken as the eye is considered to be made up of multiple scans. The results can then be combined and validated statistically thus preventing rogue data or possible no detection due to an unfortunate start point.

Two routines are provided, one which will generate the necessary x slices and one which will generate y slices. As these vary only in how the data from the image is placed into the array it will only be necessary to discuss one of them here (x slicing being chosen).

The scan starts initially from the user specified $y_c$ coordinate to $y_c$ –size (also user specified). A slice in the x direction is taken at the current y position from $x_c$ to $x_c$ – size (left hand data side). This is then validated by the slice validator and if boundaries are found a further slice is taken from $x_c$ to $x_c$ + size (right hand data side). If both slices are valid then it is deemed to be a valid eye slice and the found boundaries are entered into a deque, pushing to the front to preserve order. If the slices are not valid then a failure count is decremented and if there are more than three failures (or the y limit $y_c$ –size is met) then we must have moved out of the pupil and we move to the next stage.

The failure count is reseeded and the x slice tests above are performed as y is varied from $y_c$+1 to $y_c$ + size. If the slice is valid then the found boundaries are pushed to the back of the deque (as the order of slices needs preserving).

There is now a deque containing the found boundaries and the offset (from $y_c$ that they were found at) as shown in table 14[7]. Notice that failures do not get queued (no offsets -6, -2, 2) and slices outside the eye (the three failure count) do not pollute the data.

| | | Offset from user centre | | | |
|---|---|---|---|---|---|
| slice | | Sclera/Iris | Iris/Pupil | Pupil/Iris | Iris/Sclera |
| -11 | | 19 | 6 | 6 | 20 |
| -10 | | 19 | 6 | 6 | 19 |
| -9 | | 18 | 7 | 7 | 20 |
| -8 | | 20 | 8 | 7 | 23 |
| -7 | | 20 | 9 | 8 | 21 |
| -5 | | 19 | 10 | 9 | 49 |
| -4 | | 18 | 10 | 9 | 49 |
| -3 | | 21 | 5 | 9 | 20 |
| -1 | | 20 | 10 | 9 | 22 |
| 0 | | 20 | 11 | 9 | 22 |
| 1 | | 20 | 10 | 11 | 22 |
| 3 | | 21 | 10 | 9 | 21 |
| 4 | | 21 | 10 | 8 | 49 |

**Table 14 - Algorithm detected slice boundaries**

## *4.5 Obtaining centre values with confidence factor*

It is possible to attach a confidence level to the deque queued sliced data that can be used to gauge whether the point selected by the user really is an eye. If there are no valid slices then a confidence level of NONE is returned, otherwise extra tests are performed as outlined below.

If there are less than 3 valid slices in the deque then it is pointless processing further so a confidence level of L1 is set, the offset from start position calculated as per chapter 2.5 and the program returns.

Checking with several images it was noticed that some of the boundaries could be far too early or late to be actually valid. Having obtained the sliced boundaries some basic statistics can be used on the data set.

[7] Appendix 6

The pupil/iris boundaries are much more accurate than the iris/sclera boundaries (as it is known what happens between the pupil/iris but not for the rest of the image past the sclera) and that the "error" results are usually due to detecting earlier with hot spot interference (even though it has been much reduced in the algorithms).

The proposed method of further validating the sliced data is to first sort the data on the left hand iris/pupil in ascending order. Initially the median value was to be used and values 30% less than or 50% greater than this were to be removed, this was altered to use the value at (now sorted) deque position ¾ of deque size to give more emphasis to the larger values. Table 15 shows the data once sorted where the check value used is 10 giving a valid low of 7 and a valid high of 15.

| | | Offset from user centre | | | | |
|---|---|---|---|---|---|---|
| slice | | Sclera/Iris | *Iris/Pupil* | Pupil/Iris | Iris/Sclera | Comments |
| -3 | | 21 | 5 | 9 | 20 | Remove |
| -11 | | 19 | 6 | 6 | 20 | Remove |
| -10 | | 19 | 6 | 6 | 19 | Remove |
| -9 | | 18 | 7 | 7 | 20 | |
| -8 | | 20 | 8 | 7 | 23 | |
| -7 | | 20 | 9 | 8 | 21 | |
| -5 | | 19 | 10 | 9 | 49 | |
| -4 | | 18 | 10 | 9 | 49 | |
| -1 | | 20 | 10 | 9 | 22 | |
| 1 | | 20 | 10 | 11 | 22 | |
| 3 | | 21 | 10 | 9 | 21 | |
| 4 | | 21 | 10 | 8 | 49 | |
| 0 | | 20 | 11 | 9 | 22 | |

**Table 15 - Slices sorted Iris/Pupil boundary**

Once the values have been removed there is a new deque and if the size of this is less than three a confidence level of L2 is assigned, the deque sorted back to slice order, the offset from start position calculated and the routine returns.

The new deque is now sorted based on right hand pupil/iris, again in ascending order. The value at the ¾ deque size is found and values 30% less or 50% greater than this are removed to provide a new deque. Table 16 shows the new data and as can be seen, in this case, there are no deletions required.

| | | Offset from user centre | | | |
|---|---|---|---|---|---|
| slice | | Sclera/Iris | Iris/Pupil | *Pupil/Iris* | Iris/Sclera |
| -9 | | 18 | 7 | 7 | 20 |
| -8 | | 20 | 8 | 7 | 23 |
| -7 | | 20 | 9 | 8 | 21 |
| 4 | | 21 | 10 | 8 | 49 |
| -5 | | 19 | 10 | 9 | 49 |
| -4 | | 18 | 10 | 9 | 49 |
| -1 | | 20 | 10 | 9 | 22 |
| 3 | | 21 | 10 | 9 | 21 |
| 0 | | 20 | 11 | 9 | 22 |
| 1 | | 20 | 10 | 11 | 22 |

**Table 16 - Slices sorted Pupil/Iris boundary**

If the deque size if less than three then a confidence level of L3 is assigned otherwise the highest confidence level of L4 is assigned and in both cases the deque is re-sorted in slice order as shown in table 17.

| | | Offset from user centre | | | |
|---|---|---|---|---|---|
| slice | | Sclera/Iris | Iris/Pupil | Pupil/Iris | Iris/Sclera |
| -9 | | 18 | 7 | 7 | 20 |
| -8 | | 20 | 8 | 7 | 23 |
| -7 | | 20 | 9 | 8 | 21 |
| -5 | | 19 | 10 | 9 | 49 |
| -4 | | 18 | 10 | 9 | 49 |
| -1 | | 20 | 10 | 9 | 22 |
| 0 | | 20 | 11 | 9 | 22 |
| 1 | | 20 | 10 | 11 | 22 |
| 3 | | 21 | 10 | 9 | 21 |
| 4 | | 21 | 10 | 8 | 49 |

**Table 17 - Final validated slices**

The offset from the start position is then simply given by

$$C = \frac{\sum_{all\ slices} PI - \sum_{all\ slices} IP}{2 \times number\ of\ slices}$$

$$= \frac{(7+7+8+9+9+9+9+11+9+8) - (7+8+9+10+10+10+11+10+10+10)}{2 \times 10}$$

$$= \frac{86-95}{20} = -0.45\ pixels$$

A program was written[8] that placed all the eye detection code in a fully standalone class CEyeDetector. Using this an image could be passed and x,y positions specified to start the detection from. The software allows the user to move around the screen and, by pressing key 'c', have the detector check whether the surrounding location is an eye and if so what confidence it had in the result and where the centres of the pupil were calculated to be.

Using this software enabled the routines to be easily checked and gauged for accuracy. Figure 17 shows a test image with the second eye defined (and hence the calculation applied) that finds the correct eye centres. It also shows the calculation applied to the point 241, 301 which is the start of the retinal reflection hot spot (value of 199) and the eye is correctly identified and the centres calculated accordingly.

Note that for a point to be treated as an eye it must have confidence levels of L4 and at least three slices in the deques in both x and y directions.

[8] Appendix 7

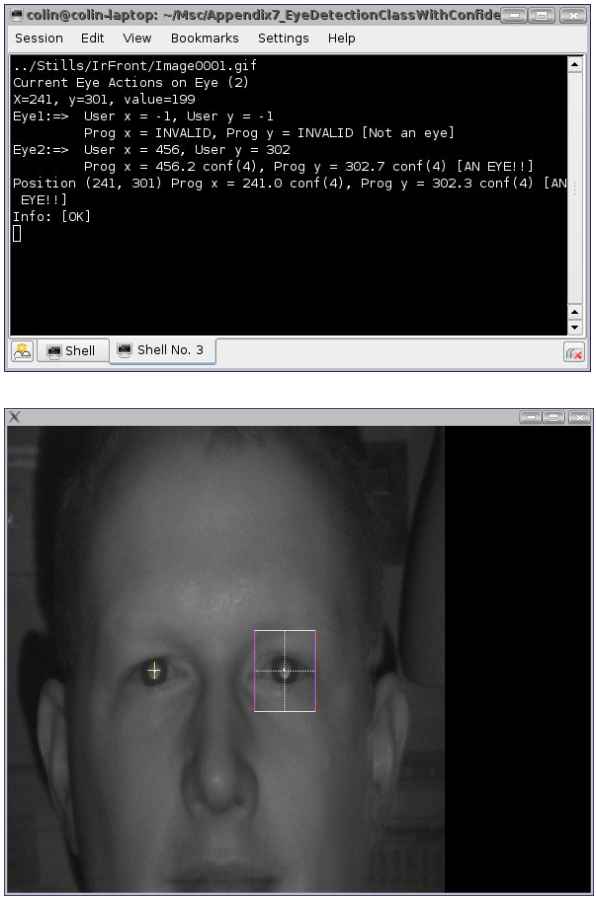



**Figure 17 - Screenshot showing centre calculation in action**

## 4.6 Increasing the detection success

The following description is for x sliced data but applies equally to the y direction by swapping references to x & y.

Using the test program it was found that confidence levels of L4 could be reliably obtained for the x sliced data when on an eye but it did depend somewhat upon where the start position was. Moving just a few pixels left or right could, in some cases, change the level from L2 to L4.

Note that for a fixed x, moving in the y direction does not significantly affect the results and this is to be expected as the routines go up to +/-50 slices from the given y when gathering x slices anyway.

As the start point will be the hot spot then the approach taken must be altered to ensure that a possible eye is not missed.

An array of adjustments from the start location and number of slices making up that adjustment is reset for each possible confidence level. When the user is on the spot to be tested for an eye (ideally the hot spot) rather than just calculating at the x start position the routine was altered to calculate from x start – 5 pixels to x start + 5 pixels.

The confidence level returned is used to update the adjustment and number of slices (remembering to add back in the offset such that all are deemed to start at the same point). After all x, y coordinates above have been checked the best confidence level is found and from all its saved data the average offset is calculated and how many total slices it had to make up the adjustment.

Empirically, for the algorithm[9] to treat the point as an eye confidence levels of L4 in both x & y sliced directions are required and there must now be at least nine valid calculations at this level in both x & y directions (as many more are used in the calculation now).

## *4.7 Automating the detection*

So far it has only been detected whether there is an eye or not when manually inputting points to check. What is required is a way to automate this process and the main reason for choosing IR illumination was to get retinal reflections from the eyes.

If the image has been correctly framed, the eyes will lie within ¼ - ¾ of the screen height and hence the full image does not require testing (creating a welcome boost in speed).

A search is made through the image starting from ¼ screen height and for each line searching for pixel values greater than the deemed retinal reflection (taken initially as

[9] Appendix 8

170). This location is then checked to see if there already is an eye here and if so then the x location is moved well past it.

If not a check is made to see if it is an already actioned hot spot (in a vector of hot spot x,y locations) and if not it is added to the hot spot locations (remembering to set the width/height of it as the initial point found will be the top left hand corner and there will be more of it below and to the left and/or right). The point is then sent to the eye detector which will inform if it is indeed an eye and if so it is added to a list of detected eyes.

This is repeated until ¾ screen height after which the checks are completed. Note that it would be possible to terminate after finding two eyes but it is inevitable that there will be some false detection and by testing and keeping a vector of all found eyes it will be able to deal with this at a later stage.

A test program was written[10] which allowed the auto detection when the image is loaded and displays the data as shown in figure 18.

[10] Appendix 9

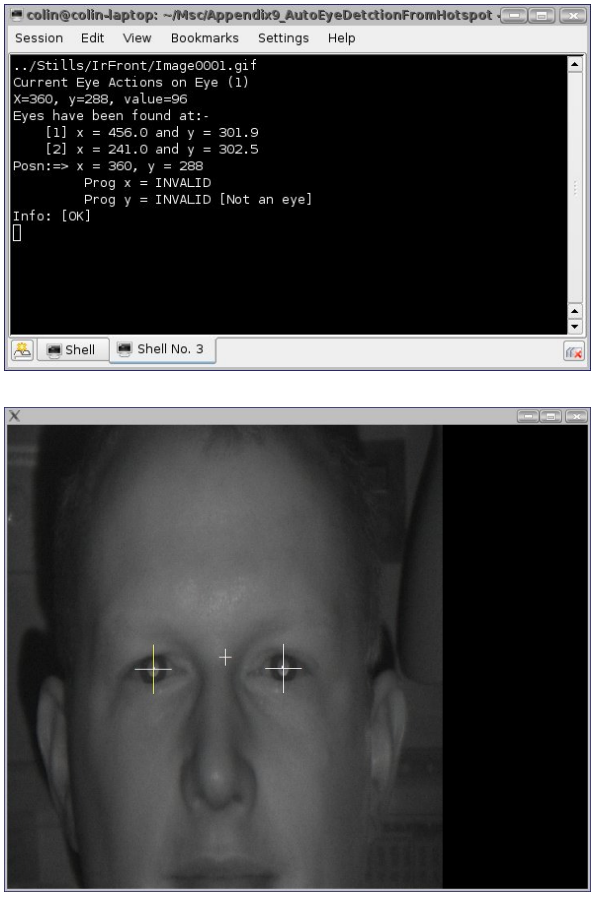



**Figure 18 - Detected Eye Locations**

## *4.8 Testing on images with glasses*

Even though the algorithm was designed to work on a frontal facial image it was decided to test the glass side views (as there are no images with glasses for front on but there are for side on) as the algorithm must work on people with glasses and the glass reflection effect needs to be negated.

### 4.8.1 Eliminating glass reflections from being detected as eyes

The glass reflections are a high value and so tend to be treated as a seed for the eye detector software. In the majority of cases the software rejected these points as not being eye shaped but under some circumstances they were being interpreted as eyes as shown in figure 19.

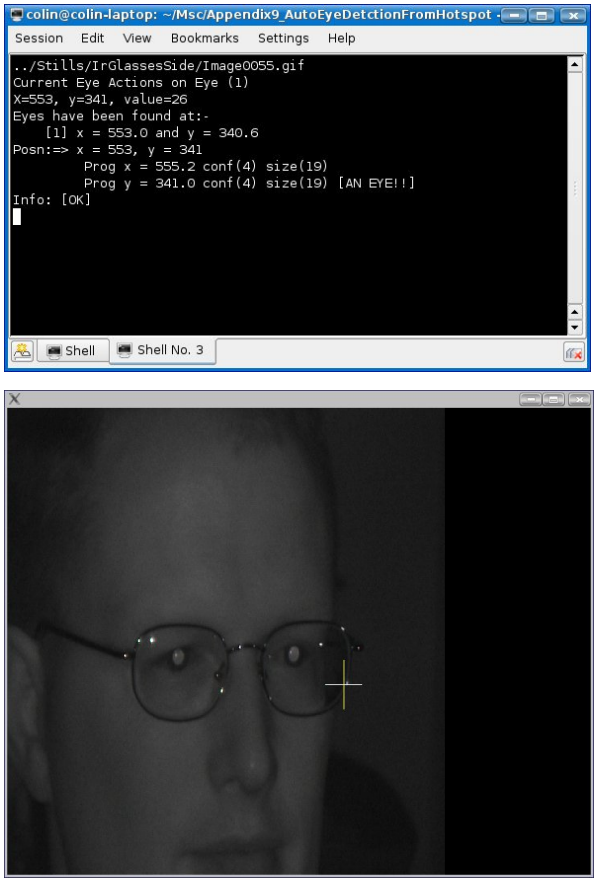



**Figure 19 - False eye detected from glass reflection**

The associated data showed that it was not the retinal reflection point causing the problem but slices further along (in y direction) that were fooling the algorithm with a low initial pixel value.

The slice routine was altered[11] to ensure that any start value of less than 35 returned as an invalid slice and this cured the problem as shown in figure 20 where there are no detected eyes and manually testing at the "found eye" location of figure 19 no longer produces an eye output.

Further analysis also led to the discovery that, in other false detection cases, there were a lot of high negative differences before the pupil/iris boundary was found. Initially six of these were allowed (as the hot spot effect also caused them) but it has now been reduced to three to ensure that the glass reflections do not count as eyes.

[11] Appendix 10

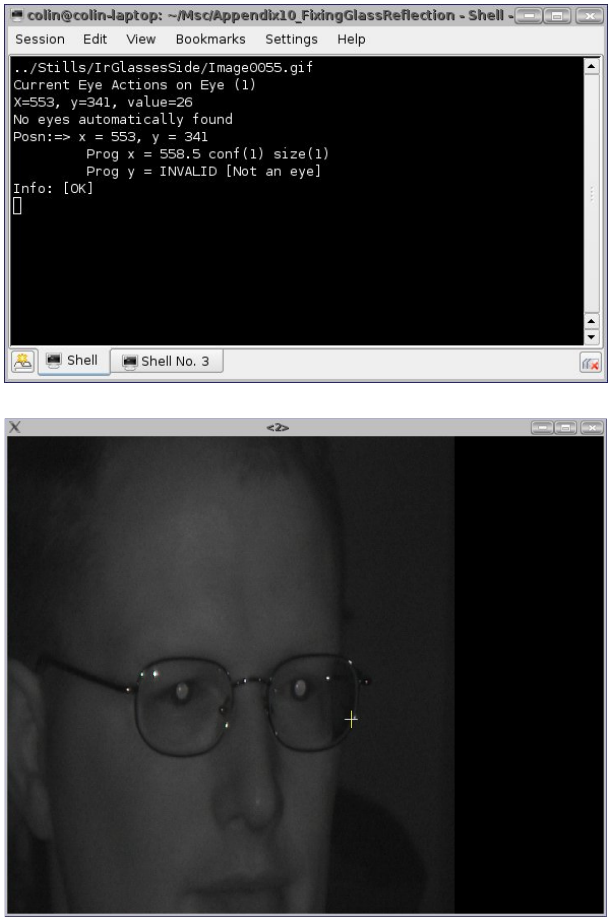



**Figure 20 - False eye detected from glass reflection corrected**

### 4.8.2 Increasing detection rate after modifications for glasses

Tightening of the parameters to remove glass reflections now causes some eyes to be missed in the IR frontal image (due to hot spot creating large negative numbers initially) as can be seen from figure 21.

We could increase the number allowed back up to six but this would be at the expense of glass reflections causing some false eye detection and for our purposes it would be better not to detect at all as the missing value can be inferred from the next and previous video frame values if necessary.

Experimentation showed that the eyes could be found with the tighter parameters if the test point was moved diagonally off the retinal reflection hot spot as can be seen in figure 21 (with a manually started algorithm calculation performed at x=242, y=343 the eye was found even though it could not by using the hot spot).

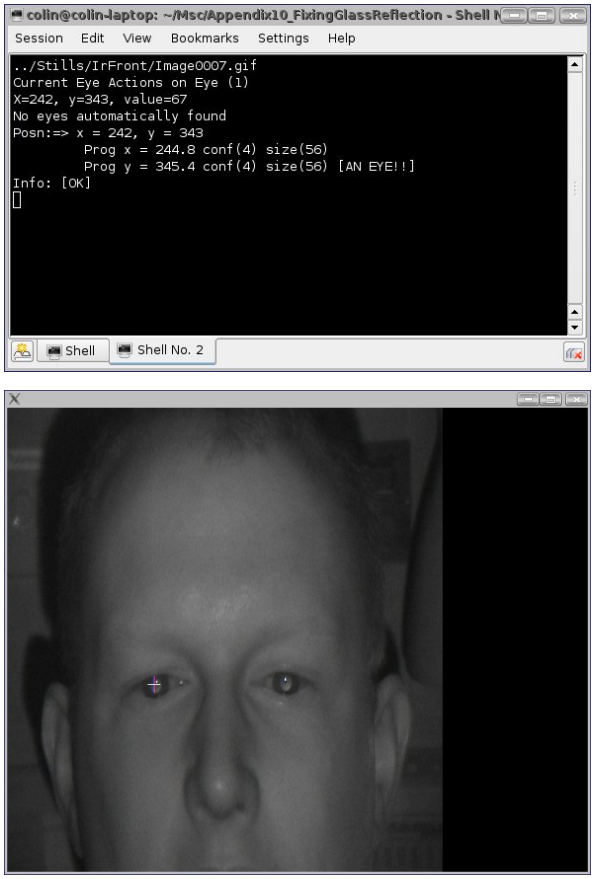



**Figure 21 - Missing eye syndrome**

As figure 22 shows, the hot spot is not always in a central position (it can be located anywhere within the pupil) and basing the detection only on this point can lead to non detection in some images.

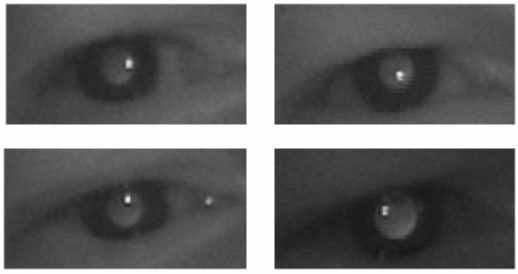


**Figure 22 - Off centre retinal reflections**

To improve the detection, albeit at a small speed penalty, the routine was changed such that instead of performing the checks at just the hot spot position the tests were actually performed at five positions.

- At {hot spot left, hot spot top}
- At {hot spot left-3, hot spot top-3}
- At {hot spot left-3, hot spot bottom+3}
- At {hot spot right+3, hot spot bottom+3}
- At {hot spot right+3, hot spot top -3}

At each position the confidence level, detected adjustments and number of slices used are saved. The highest confidence data is then used to form a new centre offset adjustment value.

$$adj = \frac{\sum_{\text{data at highest confidence level}} number\ of\ slices \times adj\ for\ those\ slices}{total\ number\ of\ slices\ at\ highest\ confidence\ level}$$

To be treated as an eye, there must be L4 confidence level and the total number of slices at L4 confidence must be greater than 20 (as many more slices in total) in both x and y directions.

Running the new SuperCalcCentres[12] routine led to both eyes being successfully found.

As the routines are now very robust the retinal reflection threshold was reduced to 130 to enable an ever greater success rate with even poorer image input.

[12] Appendix 11

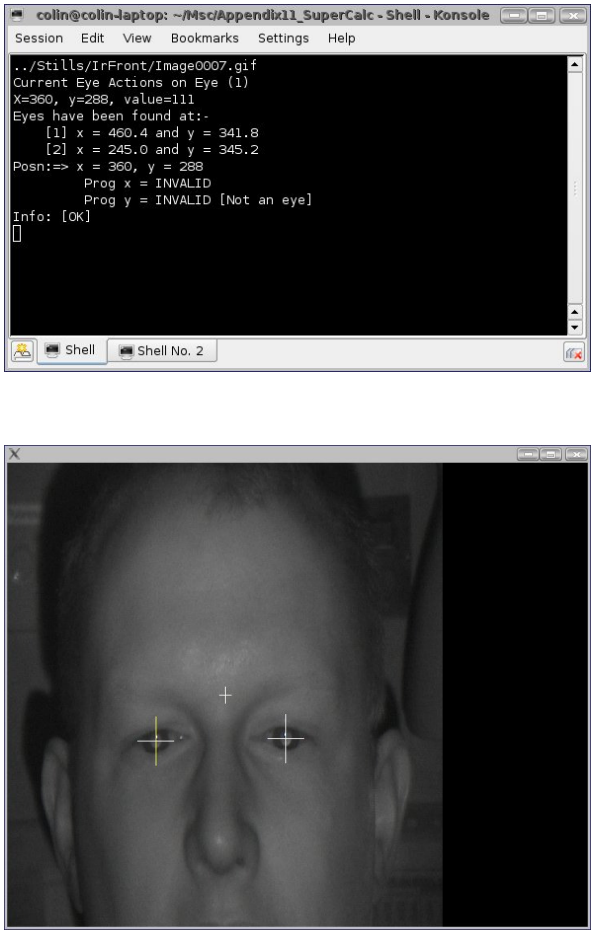



**Figure 23 - The missing eyes return**

The algorithm is now a fully stand alone class (CEyeDetector) that requires only a pointer to a source image and a call to the public routine FindEyes. It was also enhanced to check that there were only two eyes and that they were on the correct side of the image. The data was then ordered on left eye, right eye basis.

A test program[13] was written to draw the detected eye centres as the image is loaded and the image is optionally (command line parameter) saved as a jpeg file when closed. These are shown in appendix 1 and a sample in figure 24 below.

[13] Appendix 12

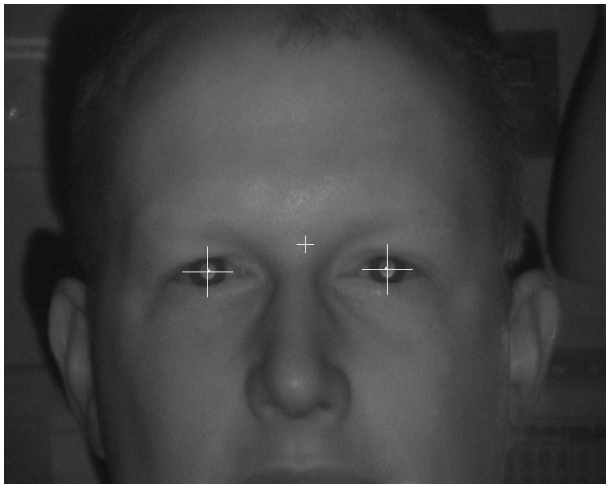


**Figure 24 - Typical jpeg output with eye locations automatically added**

## *4.9 Live video motion*

As the eye detector is now a single class it is relatively easy to write a program[14] that calls it on live video.

This program also saved the eye centres out to disk to enable the motion vectors to be obtained as in figures 25 to 28.

Note that the sudden jumps in pixel value are solely due to the fact the subject's head moves in the image and as of yet there is no feature used to fix where the general face is. Watching the live video does indeed show smooth tracking of the eyes.

Even with the jump in head positions the tracking of the left and right eye x position closely correlate as do the left and right eye y positions and it can be inferred that the patient was tracking the same direction with each eye.

To fully track the eyes would need a constant point to also be monitored and this could be the nose with some IR reflective target.

[14] Appendix 13


**Figure 25 - Left eye x motion**

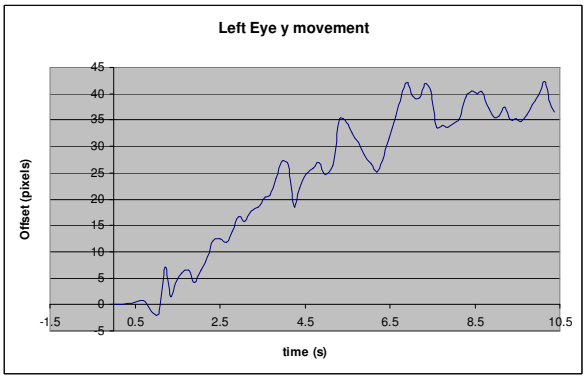



**Figure 26 - Left eye y motion**

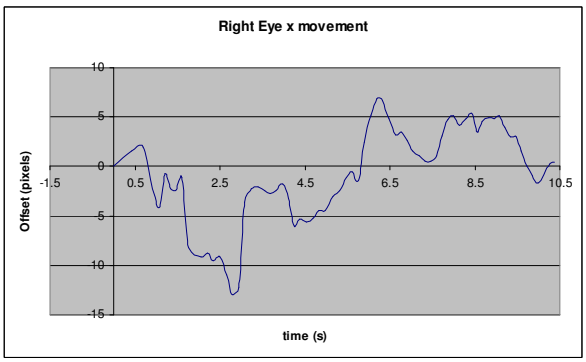



**Figure 27 - Right eye x motion**

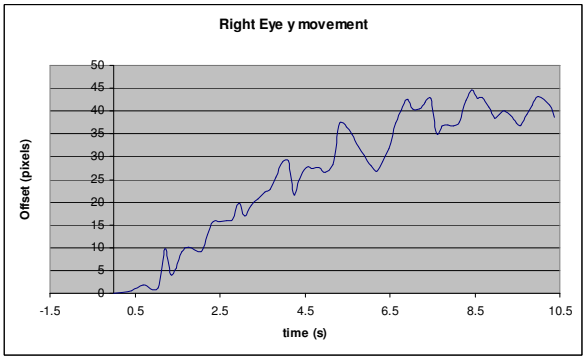



**Figure 28 - Right eye y motion**

## 4.10 Summary

This chapter showed how the basic eye slice detector could be used on multiple slices to provide the centre positions of the eyes. The data gathered was compared against that of a human operator and found to be both robust and reliable.

The routine was enhanced to utilise a concept of confidence factor and this enables reflections from non eye pixels to be rejected whilst still maintaining high detection on full facial infrared images.

The capability of the eye detector class was shown using the camcorder recording to place the detected eye centres live onto the video stream. This program also output the eye motion vectors (albeit with no fixed facial reference) and from these the eye motion could readily be observed.

# Chapter 5
# Conclusions and areas for further research

## *5.1 Conclusions*

The objective was to produce an algorithm that could automatically find the pupil centres such that eye movement would be able to be deduced by other software and this was achieved.

The end user environment was frontal images of patients with or without glasses and the algorithm worked under both sets of circumstances.

The algorithm worked on both still images and live images and showed that it is possible to track the eye centres on live video footage.

The approach taken was both novel and provides extremely good results for little processing overhead.

## *5.2 Further research areas*

The following are areas that others could research to take the thesis further.

- The actual boundaries found are saved and kept in order. This could be used to check that the found boundaries were indeed spherical and eye shaped.

- The algorithm was designed for IR images but may be useful in normal lighting conditions as the pupil is darker than the iris and the iris is darker than the sclera. The bright sclera could also be used to seed the new algorithm.

- The last detected vector of eyes is kept and as the patient is supposed to be tracking with eyes then the positions should not have changed much from the

last frame of data. This could be used to do a quick seed of the expected eye positions rather than checking over half the image pixels looking for hot spots.

- The eye centres may change due to the patient moving their head rather than their eyes (as evidenced in the motion vectors of chapter 4.9). What is needed is to find some object on the face that only moves when the head move, the nose for instance, and factor this in as well as the where the pupil centres currently are. For IR it should be possible to make a reflective target that can easily be attached to the nose and the routines altered to find this, indeed this could be found first (as it would be in a known smaller area) and the face statistics used to quickly locate possible eye positions.

- When eyes are not found for a frame or two (such as eye lid closed) then it should be possible to extrapolate where the data would have been based upon the last known positions and the currently detected position (but the output stream should indicate that it is only an estimate).

- The code could be further optimised especially the copying of data from the image into the memory arrays.

# References

[1] http://www.shu.ac.uk/mmvl/research/mimas

[2] http://www.gnu.org/software/ncurses/ncurses.html

[3] http://www.openerg.com/dse/notebooks.html

[4] A. H. Clarke, W. Teiwes and H. Scherer. Video oculography – an alternative method for measurement of three dimensional eye movements. Oculomotor Control and Cognitive processes, New York Elsevier Science Publishers, 1991.

[5] L. Ma, T. Tan, Y. Wang and D. Zhang. Efficient iris recognition by characterizing key local variations. IEEE transactions on image processing, volume 13, number 6, 2004.

[6] A. K. Jain, Y. Zhong and M. P. Dubuisson-Jolly. Deformable template models. Signal processing number 71, 1998.

[7] K. Lam and H. Yan. Locating and extracting the eye in human face images. Pattern recognition volume 29 number 5, 1996.

[8] A. Yule, P. Haalinan and C. S. Cohen. Feature extraction from faces using deformable templates. CVPR - Proceedings IEEE Computer Vision and Pattern Recognition, 1989.

[9] J. Y. Deng anf F. Lai. Region-based template deformation and masking for eye-feature extraction and description. Pattern recognition, volume 30, number 3, 1997.

[10] J. G. Daugman. Biometric personal identification system based on iris analysis. Canberra Patent Office, 1994.

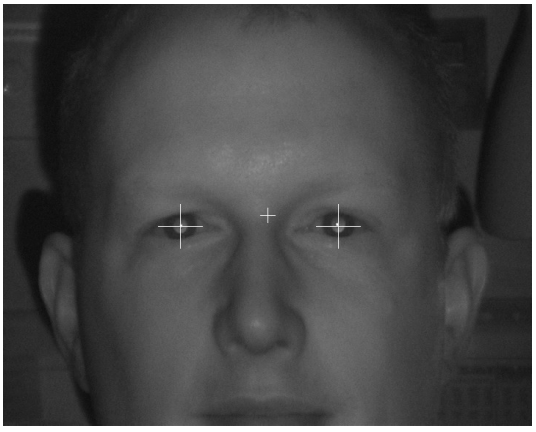
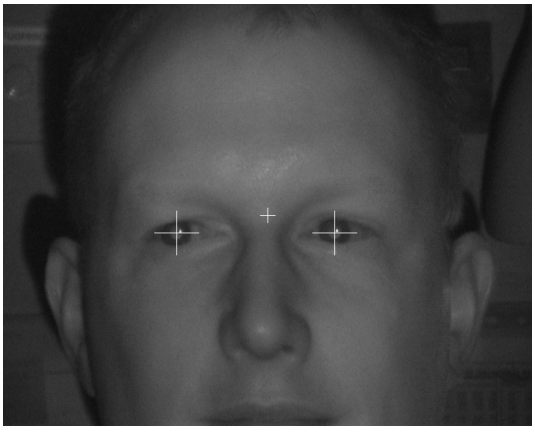
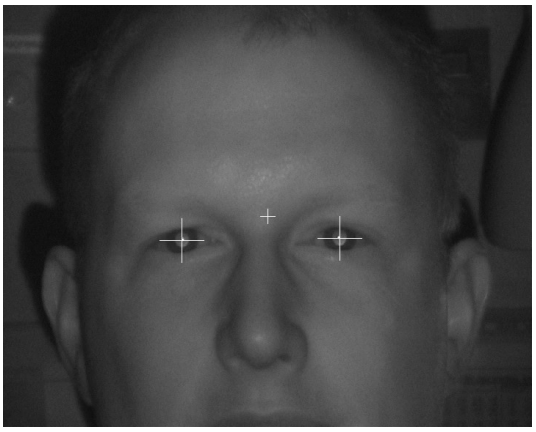
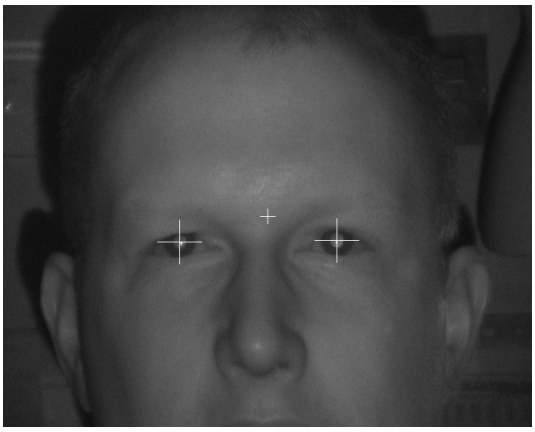
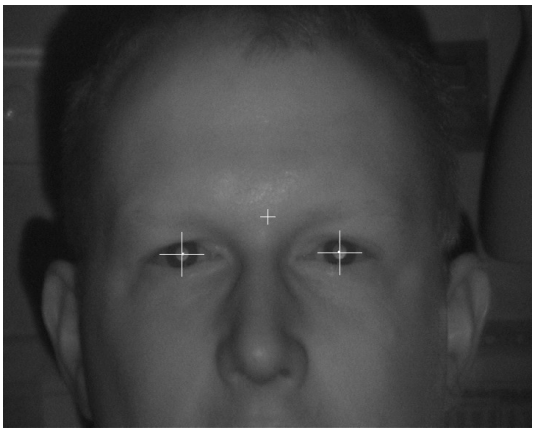
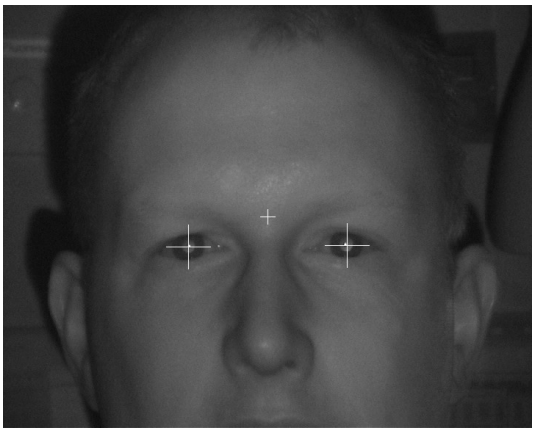
[11] C. Kimme, D. Ballard and J. Slansky. Finding circles by an array of accumulators. Communications of the ACM, volume 18, 1975.

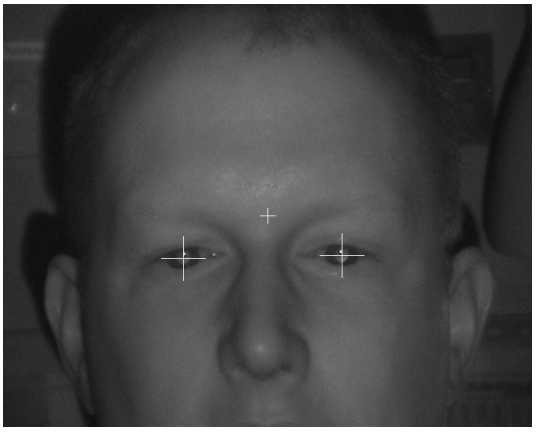
[12] K. Toennies, F. Behrens and M. Aurnhammer. Feasibility of Hough-transform-based iris localisation for real-time-application. 16th International Conference on Pattern Recognition, Volume 2, 2002.

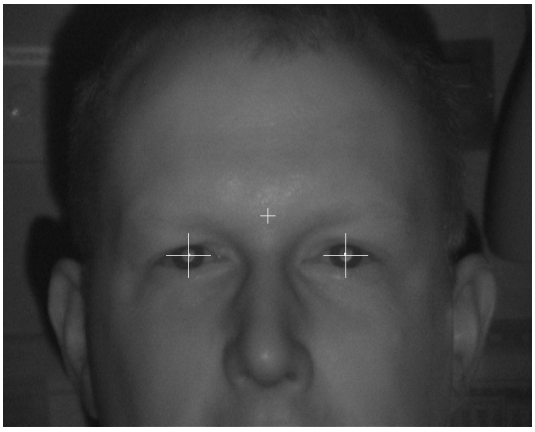
[13] N. Ritter and J.R. Cooper. Locating the iris: a first step to registration and identification. Proceeding 399, Signal and image processing 2003, 399-177.

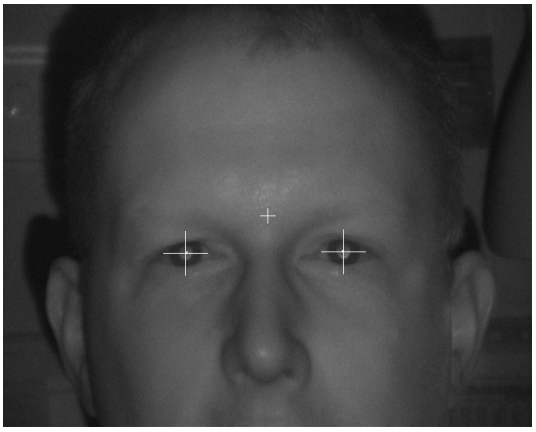
[14] A. Poursaberi and B. N. Araabi. A novel iris recognition system using morphological edge detecetor and wavelet phase features. Journal of Graphics, Vision and Image Processing, number 6, 2005.

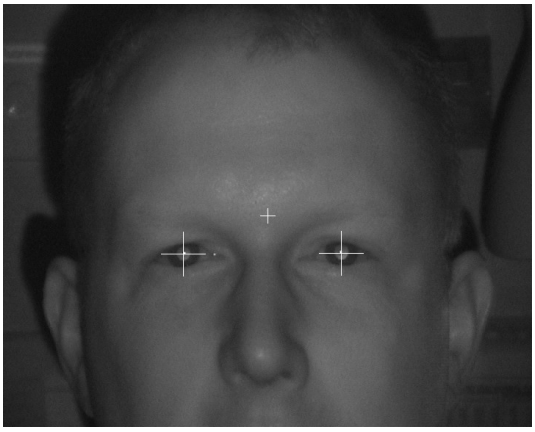
[15] http://www.stlukeseye.com/Anatomy.asp

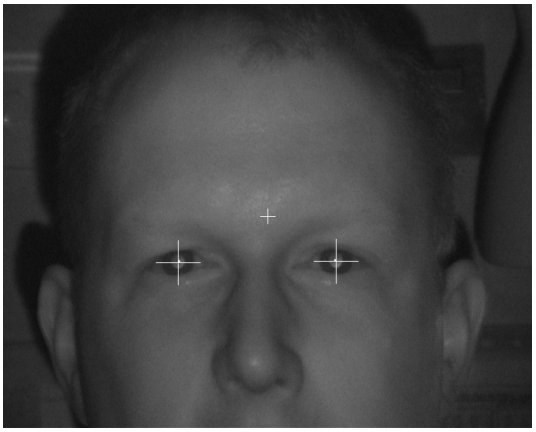

















main.cc

```
// Split the original .dv file in to separate images

#include "image_xineinput.h"
#include <GL/glut.h>

#include "image_filesinput.h"
#include "image_fileoutput.h"

using namespace std;
using namespace mimas;

int main( int argc, char *argv[] )
{
  int iImageNumber = 1;
  char ac[300];

  try{
    // Initialise GLUT.
    glutInit( &argc, argv );

    // Open X11 display.
    //x11_display display;

    // Allocate images
    image<unsigned char> imgRaw; // for raw data

    //open movie
    image_xineinput<unsigned char> input("../electrophys.dv",
                                         "auto", "auto");

     // Loop for images.
    do{
      try{
        input >> imgRaw;

        // get filename
        sprintf(ac, "Image%04d.gif", iImageNumber++);

        // get image and save it
        input >> imgRaw;
        ofstream ofile(ac, ios::binary);
        ofile << setformat( "GIF" ) << imgRaw;
        ofile.close();

        // not too many images!
        mimas::delay(100);

        static int iCount = 0;
        if(++iCount == 100)
          return 1;


      }
      catch (exception &e){
        cerr << "Warning: " << e.what() << endl;
        mimas::delay(100);
      };

    }
    while (input);
  }
  catch(exception &e){
    cerr << "Error: " << e.what() << endl;
  };

 return 0;
}
```

main.cc

```
// ------------------------------------------------------------------------
// Main code that reads in images and provides calls to the routines that
// are specific to the test we are carrying out and will be held in
// specific.cc
// ------------------------------------------------------------------------

/*
 Read in stills from command line parameter 1.  This is added to the end of
 /../Stills/ to form where the directory is.  This allows the programs to s
 save any data under a sub directory so we can get to them at a later date.
 The images inside the directory in parameter 1 are of the form Image????.gif
 where ???? is a number i.e. Image0001.gif although it does not have to start
 at 1 (must be < 1000 though!).


 ========================================================
 As shell for other routines also added global variables
 ========================================================
  iXpos that is incremented using cursor right, dec using cursor left
  iYpos that is incremented using cursor down, dec using cursor up

 The min value of the variables is 1 and the max is used from other
 globals iXposMax and iYposMax (usually set to screen limits but
 initialised to 10 to ensure valid).


 ==========
 Keypresses
 ==========
     n => next image
     p => previous image.
     cursor keys adjust values in iXpos (left/right) and iYpos (up/down)
     speed of cursor is set using
          1 => pixel jump of 1
          2 => pixel jump of 5
          3 => pixel jump of 10
          4 => pixel jump of 20


 ======================================================================
 As a shell we expect the user specific routines to define and use
 ======================================================================
 ------------------------------------------------------------------------
 User must provide the routines that actually do the work and this is
 done on user key press, we also send a reference to a copy of the
 current raw image and the key that is pressed.  On return we will
 display the copy image to the screen.
 ------------------------------------------------------------------------
 void DoUpdate(const int iKey, image<unsigned char>& imgMod);

 ------------------------------------------------------------------------
 Called when the source image is being closed to allow the other
 routines to save any helpful data (such as where the user had
 previously placed the points on screen!).
 ------------------------------------------------------------------------
 void ImageClosing(image<unsigned char>& imgRaw, const char* directory,
                      int iImageNumber);

 ------------------------------------------------------------------------
 Called when the source image has been changed (new fileloaded)
 ------------------------------------------------------------------------
 void ImageLoaded(image<unsigned char>& imgMod, const char* directory,
                     int iFileNumber);

 ------------------------------------------------------------------------
 When we have an initial image on startup we call this to allow
 the specific module to initialise.
 ------------------------------------------------------------------------
 void InitSpecific(const char* directory, image<unsigned char>* pImg);

 ------------------------------------------------------------------------
```

main.cc

```
 The real routines will need to show user data on screen and this
 will allow them chance to do so.  Here we simply show the file
 and the global iXpos, iYpos
 ------------------------------------------------------------------------
 void ShowInfo(const char* file);

 ------------------------------------------------------------------------
 The real routines may need to tidy up
 ------------------------------------------------------------------------
 void TerminateSpecific();

 ========================================
 Helper routines for general use by others
 ========================================

 ------------------------------------------------------------------------
 Given directory name,ensure that it exists under us and if not create it
 ------------------------------------------------------------------------
 void EnsureDirectory(const char* directory);

 ------------------------------------------------------------------------
 Wait for a character and return it
 ------------------------------------------------------------------------
 char Getch();

 ------------------------------------------------------------------------
 Show the current global iXpos & iYpos location as a cursor on screen
 ------------------------------------------------------------------------
 void ShowCursorPosn(image<unsigned char>& imgMod, int iSize=20);

*/

#include "main.h"

// ------------------------------------------------------------------------
// Global variables across all modules
// ------------------------------------------------------------------------
int iXpos = 1, iYpos=1;
int iXposMax = 10, iYposMax = 10;
int iPixelValue;

// ------------------------------------------------------------------------
// Globals for us only
// ------------------------------------------------------------------------
static image<unsigned char> imgRaw; // to hold the raw image
static image<unsigned char> imgMod; // to hold the modified image
static int iStartImageNumber, iCurrentImageNumber;
static int iCursorJump = 1;         // number of pixels to jump

// ########################################################################
// The following are generic and should not need to be altered
// ########################################################################

// ------------------------------------------------------------------------
// This can be called by other routines and simply draws a cursor at
// iXpos, iYpos.
// ------------------------------------------------------------------------
void ShowCursorPosn(image<unsigned char>& imgMod, int iSize)
{
  for(int i=-iSize/2; i<=iSize/2; i++){
    imgMod.setPixel(iXpos+i, iYpos, 255);
    imgMod.setPixel(iXpos, iYpos+i, 255);
  }
}

// ------------------------------------------------------------------------
// as using immediate return we must provide a routine
// that actually waits for a key when necessary
// ------------------------------------------------------------------------
char Getch()
{
```

main.cc

```
  int iVal;

  while(true)
  {
    iVal = getch();
    if(iVal != ERR)
      break;
  }

  return static_cast<char>(iVal);
}

// ---------------------------------------------------------------------
// Given a directory name (and path if appropriate) check that the
// directory exists.  If not but it is a filename then delete it and
// create as directory otherwise can just create it
// ---------------------------------------------------------------------
void EnsureDirectory(const char* directory)
{
  // check if sub directory exists
  boost::filesystem::path dirSub(directory);
  if(!boost::filesystem::exists(dirSub)){
    // create it then
    boost::filesystem::create_directory(directory);
  }
  else if(!boost::filesystem::is_directory(dirSub)){
    // get rid of it as not a sub directory
    boost::filesystem::remove(dirSub);
    // now make it a sub directory
    boost::filesystem::create_directory(directory);
  }
}

// ---------------------------------------------------------------------
// Given a directory, read current image from it.  If fail then try to
// read the first image we found when initially run.  If fail that then
// inform caller by returning false
// ---------------------------------------------------------------------
bool ReadImage(const char* directory, string& str, int iOldImageNumber,
               bool fInformOthers)
{
  char ac[300];
  sprintf(ac, "../Stills/%s/Image%04d.gif",
          directory, iCurrentImageNumber);

  if(fInformOthers){
    ImageClosing(imgRaw, directory, iOldImageNumber);
  }

  ifstream input;
  input.open(ac, ios::binary);
  if(!input){
    // try to read initial image
    iCurrentImageNumber = iStartImageNumber;
    sprintf(ac, "../Stills/%s/Image%04d.gif", directory,
            iCurrentImageNumber);
    input.open(ac, ios::binary);
    if(!input){
      str = "Start Image used has been deleted so quiting";
      return false;
    }
  }
  // can now read it
  input >> imgRaw;
  imgMod = imgRaw;

  // and close image file
  input.close();

  return true;
}
```

main.cc

```
// ------------------------------------------------------------------------
// Main routine that will read in the still images and allow selection
// around them (using n & p).  Will also increment/decrement global
// iXpos,iYpos using cursor keys.  q will quit.
// ------------------------------------------------------------------------
int main( int argc, char *argv[] )
{
  string str;
  char ac[500];

  // initialise ncurses data
  initscr();
  keypad(stdscr, TRUE);
  noecho();
  halfdelay(1);

  try {
    // Initialise GLUT.
    glutInit( &argc, argv );

    MMERROR(argc == 2, mimasexception, ,
            "Syntax: " << argv[0] << " <media resource location>" );

    // Open X11 display.
    x11_display display;
    image_mesaoutput< unsigned char > window( &display );

    // try to read in first image
    ifstream input;
    for(iStartImageNumber=1; iStartImageNumber<1000; iStartImageNumber++){
      sprintf(ac, "../Stills/%s/Image%04d.gif", argv[1], iStartImageNumber);
      input.open(ac, ios::binary);
      if(input){
        input.close();// ensure tidy up
        break;
      }
    }

    if(iStartImageNumber >= 1000){
      str = "Image not found from base of ";
      str += argv[1];
      throw str;
    }

    // ensure we have the correct subdirectories
    EnsureDirectory(argv[1]);
    sprintf(ac, "%s/ProgData", argv[1]);
    EnsureDirectory(ac);
    sprintf(ac, "%s/Eye1Data", argv[1]);
    EnsureDirectory(ac);
    sprintf(ac, "%s/Eye2Data", argv[1]);
    EnsureDirectory(ac);

    // read in the source image
    iCurrentImageNumber = iStartImageNumber;
    ReadImage(argv[1], str, iCurrentImageNumber, false);

    // allow specific routine chance to initialise
    InitSpecific(argv[1], &imgRaw);

    // and let it action the new image
    ImageLoaded(imgMod, argv[1], iCurrentImageNumber);
    window << imgMod << mimas::delay(100);

    // and show user info
    sprintf(ac, "../Stills/%s/Image%04d.gif", argv[1], iStartImageNumber);
    iPixelValue = imgRaw.getPixel(iXpos, iYpos);
    ShowInfo(ac);

    bool fQuit = false;
```

main.cc

```
// wait for next choice
while(!fQuit){
  int iVal= tolower(getch());
  if(iVal == ERR){
    window << imgMod << mimas::delay(50);
  }
  else{
    bool fNewInput = false;
    int iOldImageNumber = iCurrentImageNumber; // save for later
    switch(iVal){
      case 'q':
        ImageClosing(imgRaw, argv[1], iCurrentImageNumber);
        fQuit = true;
        break;
      case 'n':
        iCurrentImageNumber++;
        fNewInput = true;
        break;
      case 'p':
        if(iCurrentImageNumber > iStartImageNumber){
          iCurrentImageNumber--;
          fNewInput = true;
        }
        break;
      case KEY_LEFT:
        if(iXpos > iCursorJump)
          iXpos -= iCursorJump;
        break;
      case KEY_RIGHT:
        if(iXpos < iXposMax - iCursorJump)
          iXpos += iCursorJump;
        break;
      case KEY_UP:
        if(iYpos > iCursorJump)
          iYpos -= iCursorJump;
        break;
      case KEY_DOWN:
        if(iYpos < iYposMax - iCursorJump)
          iYpos += iCursorJump;
        break;
      case '1':
        iCursorJump = 1;
        break;
      case '2':
        iCursorJump = 5;
        break;
      case '3':
        iCursorJump = 10;
        break;
      case '4':
        iCursorJump = 20;
        break;
    }

    // check if new image required
    if(fNewInput){
      if(!ReadImage(argv[1], str, iOldImageNumber, true)){
        throw str;
      }
      ImageLoaded(imgMod, argv[1], iCurrentImageNumber);
    }
    else{  // action key
      imgMod = imgRaw;
      DoUpdate(iVal, imgMod);
    }

    // show user info
    sprintf(ac, "../Stills/%s/Image%04d.gif", argv[1], iCurrentImageNumber);
    iPixelValue = imgRaw.getPixel(iXpos, iYpos);
    ShowInfo(ac);
```

main.cc

```
            // and finally show the *possibly* new image
            window << imgMod;
          }
        }
      }
      catch(string& str){
        printw("Error: %s\n", str.c_str());
        Getch();
      }
      catch(exception &e){
        printw("Error: %s\n", e.what());
        Getch();
      };

      // allow specific code to tidy up
      TerminateSpecific();

      // shutdown ncurses
      endwin();

      return 0;
}
```

main.h

```
#ifndef main_CW_h
#define main_CW_h

#include <GL/glut.h>
#include "image_mesaoutput.h"
#include "image_fileinput.h"
#include <curses.h>

#include <iostream>
#include <fstream>
#include "boost/filesystem/operations.hpp"

using namespace std;
using namespace mimas;

// ########################################################################
// These routines must be provided in specific.cc (which is specific
// to each test).
// ########################################################################

// ------------------------------------------------------------------------
// We must provide the routines that actually do the work and this is
// done on user key press, we also send a reference to a copy of the
// current raw image and the key that is pressed.  On return we will
// display the copy image to the screen.
// ------------------------------------------------------------------------
void DoUpdate(const int iKey, image<unsigned char>& imgMod);

// ------------------------------------------------------------------------
// Called when the source image is being closed to allow the other
// routines to save any helpful data (such as where the user had
// previously placed the points on screen!).
// ------------------------------------------------------------------------
void ImageClosing(image<unsigned char>& imgRaw, const char* directory,
                  int iImageNumber);

// ------------------------------------------------------------------------
// Called when the source image has been changed (new fileloaded)
// ------------------------------------------------------------------------
void ImageLoaded(image<unsigned char>& imgMod, const char* directory,
                 int iFileNumber);

// ------------------------------------------------------------------------
// When we have an initial image on startup we call this to allow
// the specific module to initialise.
// ------------------------------------------------------------------------
void InitSpecific(const char* directory, image<unsigned char>* pImg);

// ------------------------------------------------------------------------
// The real routines will need to show user data on screen and this
// will allow them chance to do so.  Here we simply show the file
// and the global iXpos, iYpos
// ------------------------------------------------------------------------
void ShowInfo(const char* file);

// ------------------------------------------------------------------------
// The real routines may need to tidy up
// ------------------------------------------------------------------------
void TerminateSpecific();

// ------------------------------------------------------------------------
// Global variables across all modules
// ------------------------------------------------------------------------
extern int iXpos, iYpos;
extern int iXposMax, iYposMax;
extern int iPixelValue;

// ########################################################################
// Provided by main for all to use where required
// ########################################################################
```

main.h

```
// ---------------------------------------------------------------------
// Given directory name,ensure that it exists under us and if not create it
// ---------------------------------------------------------------------
void EnsureDirectory(const char* directory);

// ---------------------------------------------------------------------
// Wait for a character and return it
// ---------------------------------------------------------------------
char Getch();

// ---------------------------------------------------------------------
// Show the current global iXpos & iYpos location as a cursor on screen
// ---------------------------------------------------------------------
void ShowCursorPosn(image<unsigned char>& imgMod, int iSize=20);

#endif
```

specific.cc

```
// ------------------------------------------------------------------------
// included files
// ------------------------------------------------------------------------
#include "main.h"
#include "boost/filesystem/operations.hpp"

// ------------------------------------------------------------------------
// global this file only
// ------------------------------------------------------------------------
static bool fInit=true;

// ------------------------------------------------------------------------
// We must provide the routines that actually do the work and this is
// done on user key press, we also send a reference to a copy of the
// current raw image and the key that is pressed.  On return we will
// display the copy image to the screen.
//
// The default one here simply shows a cursor on screen
// ------------------------------------------------------------------------
void DoUpdate(const int iKey, image<unsigned char>& imgMod)
{
  for(int i=-15; i<15; i++){
    imgMod.setPixel(iXpos, iYpos+i, 255);
    imgMod.setPixel(iXpos+i, iYpos, 255);
  }
}

// ------------------------------------------------------------------------
// Called when the source image is being closed to allow the other
// routines to save any helpful data (such as where the user had
// previously placed the points on screen!).
// ------------------------------------------------------------------------
void ImageClosing(image<unsigned char>& imgRaw, const char* directory,
                  int iImageNumber)
{
}

// ------------------------------------------------------------------------
// Called when the source image has been changed (new fileloaded).
// Here we simply get the size and set the iXposMax, iYposMax extremities.
// and set iXpos & iYpos to middle of screen if we have not done
// so previously
// ------------------------------------------------------------------------
void ImageLoaded(image<unsigned char>& imgMod, const char* directory,
                 int iFileNumber)
{
  if(fInit){
    fInit = false;
    iXposMax = imgMod.getWidth();
    iYposMax = imgMod.getHeight();
    iXpos = iXposMax/2;
    iYpos = iYposMax/2;
    DoUpdate(-1, imgMod);
  }
}

// ------------------------------------------------------------------------
// When we have an initial image on startup we call this to allow
// the specific module to initialise.
// ------------------------------------------------------------------------
void InitSpecific(const char* directory, image<unsigned char>* pImg)
{
}

// ------------------------------------------------------------------------
// The real routines will need to show user data on screen and this
// will allow them chance to do so.  Here we simply show the file
// and the global iXpos, iYpos
// ------------------------------------------------------------------------
void ShowInfo(const char* file)
{
```

specific.cc

```
  mvprintw(0,0,"%s", file);
  clrtoeol();
  mvprintw(1,0, "X=%i, y=%i", iXpos, iYpos);
  clrtoeol();
  move(2,0);
  refresh();
}

// ------------------------------------------------------------------------
// Do any tidy up work, here we have none to do
// ------------------------------------------------------------------------
void TerminateSpecific()
{
}
```

specific.cc

```
/*
 Allow boundaries for each eye that can be seen on screen.  A box is put
 around the eye (based on centre) and the user can save the eye data
 as csv files (will go in the argv[1] sub directory and have same
 name as image it was for with .eyeX.Q.csv appended (X=eye number) and
 will contain the data sliced in direction Q (x or y slice).  It will
 also use the defined boundaries to enable the user markers to appear
 in the file for analysis as well.

 The main reason for this is to allow the underlying statistics to be
 gathered and analysed.

  Additional Keys:==>
     F1 = set x - LHS Sclera/Iris boundary on active eye
     F2 = set x - LHS Iris/Pupil boundary on active eye
     F3 = set x - RHS Pupil/Iris boundary on active eye
     F4 = set x - RHS Iris/Sclera boundary on active eye

     F5 = set y - TOP Sclera/Iris boundary on active eye
     F6 = set y - TOP Iris/Pupil boundary on active eye
     F7 = set y - BOTTOM Pupil/Iris boundary on active eye
     F8 = set y - BOTTOM Iris/Sclera boundary on active eye

     F9  = set eye centre on active eye
     F11 = Make eye 1 active - NOTE: clears all eye 1 boundaries
     F12 = Make eye 2 active - NOTE: clears all eye 2 boundaries
*/

// ----------------------------------------------------------------------
// included files
// ----------------------------------------------------------------------
#include "main.h"
#include "Test.h"

// ----------------------------------------------------------------------
// global this file only
// ----------------------------------------------------------------------
static STestData testData;
static string strMessage;

// ----------------------------------------------------------------------
// routines this module only
// ----------------------------------------------------------------------
static void ShowUserEyeDataOnImage(const SEyeTestData& sEyeTestData,
                                   image<unsigned char>& imgMod);

static void WriteSliceData(image<unsigned char>& imgRaw, int iEyeNumber,
                           const char* directory, int iImageNumber);
static void WriteSliceHeader(ofstream& output, SEyeTestData* pEye,
                             bool fUseX, image<unsigned char>& imgRaw);

// ----------------------------------------------------------------------
// We must provide the routines that actually do the work and this is
// done on user key press, we also send a reference to a copy of the
// current raw image and the key that is pressed.  On return we will
// display the copy image to the screen.
// ----------------------------------------------------------------------
void DoUpdate(const int iKey, image<unsigned char>& imgMod)
{
  // where the F keys map into the x or y array
  static const int caiNORM_F_KEYS[8] = {0,1,3,4,0,1,3,4};

  // get a pointer to the eye we are working on
  SEyeTestData* pEye = (testData.fEye1Active) ?  &testData.testEye1 :
                                                 &testData.testEye2;

  strMessage = "OK"; // assume we will be ok
  // check initially for the F1..F8 keys

  // check if defining user points
  if((iKey >= KEY_F(1)) && iKey <= KEY_F(8)){
```

```
    // get pointer to data to check.alter
    int* pai;
    int iPosn;
    if(iKey < KEY_F(5)){
      pai = pEye->aiXposns;
      iPosn = iXpos;
    }
    else{
      pai = pEye->aiYposns;
      iPosn = iYpos;
    }
    // get offset in to array
    int iOffset = caiNORM_F_KEYS[iKey-KEY_F(1)];

    // the centre point must be defined before anything else
    if(pai[2] == -1)
      strMessage = "Must define centre point first";
    else{
      // must be within the user bounding box
      if((iPosn < (pai[2]-pEye->iSizeBox)) ||
              (iPosn > (pai[2]+pEye->iSizeBox)))
        strMessage = "Must be within box";
      else{
        // check if greater than previous one
        if(iOffset && (iPosn <= pai[iOffset-1]))
          strMessage = "must be greater than previous value";
        //check if less than next one
        else if((iOffset < SEyeTestData::eNumIntPosns-1) &&
                (iPosn >= pai[iOffset+1]) && (pai[iOffset+1] != -1))
          strMessage = "must be less than next value";
        else
          pai[iOffset] = iPosn;// success!
      }
    }
  }
  else{
    // can now check other keys
    switch(iKey){
      case KEY_F(9):// set centre on active eye
        // check that position is at least valid
        if((iXpos > pEye->iSizeBox) && (iYpos > pEye->iSizeBox)){
          pEye->NewValue(iXpos, iYpos);
        }
        else
          beep();
        break;
      case KEY_F(11):
        testData.fEye1Active = true;
        break;
      case KEY_F(12):
        testData.fEye1Active = false;
        break;
    }
  }

  // show user points on the image
  ShowUserEyeDataOnImage(testData.testEye1, imgMod);
  ShowUserEyeDataOnImage(testData.testEye2, imgMod);

  // and show data on command line
  ShowCursorPosn(imgMod);
}

// ---------------------------------------------------------------------
// Called when the source image is being closed to allow the other
// routines to save any helpful data (such as where the user had
// previously placed the points on screen!).
// ---------------------------------------------------------------------
void ImageClosing(image<unsigned char>& imgRaw, const char* directory,
                  int iImageNumber)
{  // save current x,y cursor position
```

```
  testData.iSavedXpos = iXpos;
  testData.iSavedYpos = iYpos;

  // simply write the test structure out as binary data
  ofstream output;
  char ac[800];
  sprintf(ac, "%s/ProgData/Image%04d.prgData", directory, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  if(output){
    output.write((char*)&testData, sizeof(testData));
    output.close();
  }

  // now write out the x & y user boxed sliced data for each eye
  WriteSliceData(imgRaw, 1, directory, iImageNumber);
  WriteSliceData(imgRaw, 2, directory, iImageNumber);
}

// ---------------------------------------------------------------------
// Called when the source image has been changed (new fileloaded).
// Here we simply get the size and set the iXposMax, iYposMax extremities.
// and set iXpos & iYpos to middle of screen if we have not done
// so previously
// ---------------------------------------------------------------------
void ImageLoaded(image<unsigned char>& imgMod, const char* directory,
                 int iFileNumber)
{
  // get screen limits
  iXposMax = imgMod.getWidth();
  iYposMax = imgMod.getHeight();

  // set deafult positions
  iXpos = iXposMax/2;
  iYpos = iYposMax/2;

  // clear old test data
  testData.Clear();

  // read in any associated test data (if any)
  ifstream input;
  char ac[800];
  sprintf(ac, "%s/ProgData/Image%04d.prgData", directory, iFileNumber);
  input.open(ac, ios::binary);
  if(input){
    // read in data
    input.read((char*)&testData, sizeof(testData));
    input.close();

    // set last x,y position as current
    iXpos = testData.iSavedXpos;
    iYpos = testData.iSavedYpos;
  }

  // actually draws our data
  DoUpdate(-1, imgMod);
}

// ---------------------------------------------------------------------
// When we have an initial image on startup we call this to allow
// the specific module to initialise.
// ---------------------------------------------------------------------
void InitSpecific(const char* directory, image<unsigned char>* pImg)
{
    // ensure that we have SliceData subdirectory under both eye data dirs.
  char ac[300];
  sprintf(ac,"%s/Eye1Data/SliceData", directory);
  EnsureDirectory(ac);
  sprintf(ac,"%s/Eye2Data/SliceData", directory);
  EnsureDirectory(ac);
}
```

specific.cc

```
// ---------------------------------------------------------------------
// The real routines will need to show user data on screen and this
// will allow them chance to do so.
// ---------------------------------------------------------------------
void ShowInfo(const char* file)
{
  clear();
  printw("%s\n", file);
  printw("Current Eye Actions on Eye (%d)\n", (testData.fEye1Active)? 1:2);
  printw("X=%i, y=%i, value=%d\n", iXpos, iYpos, iPixelValue);
  printw("Info: [%s]\n", strMessage.c_str());
  refresh();
}

// ---------------------------------------------------------------------
// Do any tidy up work, here we have none to do
// ---------------------------------------------------------------------
void TerminateSpecific()
{
}

// #####################################################################
// These are called only in this module and not directly from main.cc
// #####################################################################

// ---------------------------------------------------------------------
// Show the user test data on the image
// ---------------------------------------------------------------------
void ShowUserEyeDataOnImage(const SEyeTestData& sEyeTestData,
                            image<unsigned char>& imgMod)
{
  if((sEyeTestData.GetXcentre() > sEyeTestData.iSizeBox)
       && (sEyeTestData.GetYcentre() > sEyeTestData.iSizeBox)){
    // draw box around the centre point
    for(int i=-sEyeTestData.iSizeBox; i<=sEyeTestData.iSizeBox; i++){
      imgMod.setPixel(sEyeTestData.GetXcentre()+i,
                      sEyeTestData.GetYcentre()-sEyeTestData.iSizeBox,
                      255);
      imgMod.setPixel(sEyeTestData.GetXcentre()+i,
                      sEyeTestData.GetYcentre()+sEyeTestData.iSizeBox,
                      255);
      imgMod.setPixel(sEyeTestData.GetXcentre()-sEyeTestData.iSizeBox,
                      sEyeTestData.GetYcentre()+i,
                      255);
      imgMod.setPixel(sEyeTestData.GetXcentre()+sEyeTestData.iSizeBox,
                      sEyeTestData.GetYcentre()+i,
                      255);
      // and centre cursor!
      if(i & 1){
        imgMod.setPixel(sEyeTestData.GetXcentre()+i,
                        sEyeTestData.GetYcentre(),
                        255);
        imgMod.setPixel(sEyeTestData.GetXcentre(),
                        sEyeTestData.GetYcentre()+i,
                        255);
      }

      // check if need to position user x locators
      for(int i=0; i<2; i++){
        if(sEyeTestData.aiXposns[i] != -1){
          for(int j=0; j<sEyeTestData.iSizeBox; j++){
            imgMod.setPixel(sEyeTestData.aiXposns[i],
                            sEyeTestData.GetYcentre()+j, 255);
          }
        }
      }
      for(int i=3; i<5; i++){
        if(sEyeTestData.aiXposns[i] != -1){
          for(int j=0; j<sEyeTestData.iSizeBox; j++){
            imgMod.setPixel(sEyeTestData.aiXposns[i],
                            sEyeTestData.GetYcentre()-j, 255);
```

```
            }
          }
        }

        // check if need to position user y locators
        for(int i=0; i<2; i++){
          if(sEyeTestData.aiYposns[i] != -1){
            for(int j=0; j<sEyeTestData.iSizeBox; j++){
              imgMod.setPixel(sEyeTestData.GetXcentre()-j,
                              sEyeTestData.aiYposns[i], 255);
            }
          }
        }
        for(int i=3; i<5; i++){
          if(sEyeTestData.aiYposns[i] != -1){
            for(int j=0; j<sEyeTestData.iSizeBox; j++){
              imgMod.setPixel(sEyeTestData.GetXcentre()+j,
                              sEyeTestData.aiYposns[i], 255);
            }
          }
        }
      }
    }
}

// ---------------------------------------------------------------------------
// Write out the images user sliced (in x & y) data for given eye
// ---------------------------------------------------------------------------
void WriteSliceData(image<unsigned char>& imgRaw, int iEyeNumber,
                        const char* directory, int iImageNumber)
{
  ofstream output;

  // form filename and path, set default output
  char ac[500];
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
          iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
          iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  // get pointer to correct eye
  SEyeTestData* pEye = (iEyeNumber == 1) ? &testData.testEye1 :
                                           &testData.testEye2;

  // ensure that at least centre value is defined
  if((pEye->aiXposns[2] == -1) || (pEye->aiYposns[2] == -1))
    return;

  // write header for x direction
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
          iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  WriteSliceHeader(output, pEye, true, imgRaw);

  // now finished with file
  output.close();

  // write header for y direction
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
          iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  WriteSliceHeader(output, pEye, false, imgRaw);

  // now finished with file
```

```
  output.close();
}

void WriteSliceHeader(ofstream& output, SEyeTestData* pEye,
                      bool fUseX, image<unsigned char>& imgRaw)
{
  // save screen info
  output << "screen width(" << imgRaw.getWidth();
  output << ") screen height(" << imgRaw.getHeight();

  // output user entered data
  if(fUseX){
    output << ")\n\nuser x,";
  }
  else{
    output << ")\n\nuser y,";
  }

  // now ensure the user marker points are shown
  int* pai = (fUseX) ? pEye->aiXposns : pEye->aiYposns;
  int iCentre = pai[2];
  int iOffset = 0;
  for(int i=-pEye->iSizeBox; i<=pEye->iSizeBox; i++){
    if((iCentre+i) == pai[iOffset]){
      // user marker position
      if(i == 0){
        output << ',' << iOffset+1 << "[";
        output << iCentre+i << "],,";
      }
      else{
        output << iOffset+1 << "[" << iCentre+i << "],";
      }
      if(iOffset < SEyeTestData::eNumIntPosns-1)
        iOffset++;
    }
    else
      output << ',';
  }
  output << "\n,";

  // now the offset (from centre) information
  for(int i=-pEye->iSizeBox; i<=pEye->iSizeBox; i++){
    if(i)
      output << i << ',';
    else
      output << ',' << i << ",,";
  }
  output << endl;

  // the actual slices can now be output
  int x, y;
  for(int i=-pEye->iSizeBox; i<=pEye->iSizeBox; i++){
    // save slice number
    output << i << ',';

    // cycle slice data
    for(int j=-pEye->iSizeBox; j<=pEye->iSizeBox; j++){
      // calc x, y positions
      if(fUseX){
        x = pEye->GetXcentre() + j;
        y = pEye->GetYcentre() + i;
      }
      else{
        x = pEye->GetXcentre() + i;
        y = pEye->GetYcentre() + j;
      }

      // get pixel and save it
      int iVal = imgRaw.getPixel(x, y);
      if(j == 0)
        output << ',' << iVal << ",,";
```

specific.cc

```
        else
          output << iVal << ',';
      }
      output << endl;
    }
}
```

Test.h

```
#ifndef Test_CW_h
#define Test_CW_h

#include "Point.h"

// ---------------------------------------------------------------------
// We need to hold the user points in a structure for ease
// ---------------------------------------------------------------------
struct SEyeTestData
{
  // #####################################################################
  // functions
  // #####################################################################

  // -----------
  // constructor
  // -----------
  SEyeTestData()
  {
    Clear(50);
  }

  // -------------------
  // clear out user data
  // -------------------
  void Clear(int iSizeBoxIn)
  {
    for(int i=0; i<=eNumIntPosns; i++)
      aiXposns[i] = aiYposns[i] = -1;
    iSizeBox = iSizeBoxIn;
  }

  // -------------------------------------
  // recall the user x centre (-1 if none)
  // -------------------------------------
  int GetXcentre()const {return aiXposns[2];}

  // -------------------------------------
  // recall the user y centre (-1 if none)
  // -------------------------------------
  int GetYcentre()const {return aiYposns[2];}

  // ---------------------
  // set new centre values
  // ---------------------
  void NewValue(int xIn, int yIn)
  {
    Clear(iSizeBox);
    aiXposns[2]=xIn;
    aiYposns[2]=yIn;
  }

  // #####################################################################
  // constants
  // #####################################################################
  enum {eNumIntPosns=5}; // pos 0 = SI, 1=IP, 2 = centre, 3 = PI, 4 = IP

  // #####################################################################
  // variables
  // #####################################################################
  int aiXposns[eNumIntPosns], aiYposns[eNumIntPosns];
  int iSizeBox; // how large box (+/-) around eye is for grabbing samples
};

// ---------------------------------------------------------------------
// We need to hold the the test data
// ---------------------------------------------------------------------
struct STestData
{
  // #####################################################################
  // functions
```

Test.h

```
  // ######################################################################

  // -----------
  // constructor
  // -----------
  STestData() : fEye1Active(true) {}

  // ------------------------
  // clear both user eye data
  // ------------------------
  void Clear()
  {
    testEye1.Clear(testEye1.iSizeBox);
    testEye2.Clear(testEye1.iSizeBox);
  }

  // ######################################################################
  // variables
  // ######################################################################
  SEyeTestData testEye1, testEye2; // two eyes
  bool fEye1Active;                  // which one we are working on
  int iSavedXpos, iSavedYpos;        // last known cursor position
};

#endif
```

Point.h

```
#ifndef Point_CW_h
#define Point_CW_h

// ---------------------------------------------------------------------
// structure for working on a point - not made private as we do not
// need to do so here.
// ---------------------------------------------------------------------
struct SPoint
{
   // ###################################################################
   // functions
   // ###################################################################

   // -----------
   // constructor
   // -----------
   SPoint(int xIn=-1, int yIn=-1) : x(xIn), y(yIn) {}

   // ###################################################################
   // variables
   // ###################################################################
   int x, y;
};

#endif
```

Specific.cc changes from Appendix 4

- New Description

```
/*
 As well as allowing the user to set data on eye centres and boundaries
 we also allow the algorithm to calculate (and output) what it thinks
 the boundaries are (both in x & y directions)

 The algorithm to be used for getting the pupil/iris & iris/sclera
 boundaries is given by looking at the previous two values added
 together and subtracting from the next two values added together which
 will find the biggest difference.

 The high reflectivity from the pupil leads to values that will cause
 the algorithm problems and hence we must deal with this.  We do not
 simply average the whole "hot spot" as some values in it are valid
 and we are better using those values rather than the averaged values.
 The method used here is to see if the current value and the next value
 are less than the "hot spot" value and if not we increment our position
 to the next value and recheck.  Once we have two values at position p,
 we make the previous p-1 values = 50% value of p + 50% value of p+1

 The iris will be less than the pupil by some factor and here I have used
 15% of the start value (p,p+1 value used above).  As we are using two
 values for the difference then we actually use twice this value and it
 is held in ciPI_PC_DIFF.

 We then scan looking for a difference greater than this (if we find more than
 ciPI_NEG_ALLOWED (=6) negative values before finding it,then we reject that
 slice as a possible eye).  Once we find it we save the position and
 ciIS_PC_DIFF * the value at that position (as iThreshDiffIS) and move on to
 the next section.

 We are now expecting to get a better positive threshold (in which case we
 save the new position and recalculate iThreshDiffIS). Once we have started
 to get negative differences (on way back up!) then we have found the possible
 pupil/iris boundary.  If we find, at any time here, a value less than
 -iThreshDiffIS then we reject that slice.

 we are now just looking for a value less than -iThreshDiffIS to find the
 iris/sclera boundary.  NOTE that if we get more than ciIS_POS_ALLOWED (5)
 values or if the value is > 70% of iThreshDiffIS then the slice is
 rejected.  Once we have a valid difference < -iThreshDiffIS we peek at the
 next difference and if it is less than us we use that as the boundary
 location instead.

 The save routine saves the calculated slice data as .slice.csv
*/
```

- Added include file

```
#include "EyeDetector.h"
```

- Changed InitSpecific routine

```
void InitSpecific(const char* directory, image<unsigned char>* pImg)
{
  // ensure that we have SliceData subdirectory under both eye data dirs.
  char ac[300];
  sprintf(ac,"%s/Eye1Data/SliceData", directory);
  EnsureDirectory(ac);
  sprintf(ac,"%s/Eye2Data/SliceData", directory);
  EnsureDirectory(ac);
  sprintf(ac,"%s/Eye1Data/AlgData", directory);
  EnsureDirectory(ac);
  sprintf(ac,"%s/Eye2Data/AlgData", directory);
  EnsureDirectory(ac);

  // make an eye detector
  pEyeDetector = new CEyeDetector(pImg);
}
```

Specific.cc changes from Appendix 4

- Changed TerminateSpecifc routine

```
void TerminateSpecific()
{
  delete pEyeDetector;
}
```

- Changed WriteSliceData routine

```
void WriteSliceData(image<unsigned char>& imgRaw, int iEyeNumber,
                    const char* directory, int iImageNumber)
{
  ofstream output;

  // form filename and path, set default output
  char ac[500];
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  // get pointer to correct eye
  SEyeTestData* pEye = (iEyeNumber == 1) ? &testData.testEye1 :
                                           &testData.testEye2;

  // ensure that at least centre value is defined
  if((pEye->aiXposns[2] == -1) || (pEye->aiYposns[2] == -1))
    return;

  // write header for x direction
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  WriteSliceHeader(output, pEye, true, imgRaw);

  // now finished with file
  output.close();

  // write header for y direction
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  WriteSliceHeader(output, pEye, false, imgRaw);

  // now finished with file
  output.close();

  // write out the algorithm x sliced data
  // 1st header
  sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
```

```
    output << "slice, xc, si, ip, pi, is" << endl;
    // now get the actual data for x slices
    SEyeBoundaries sEyeBoundaries;
    for(int i=-pEye->iSizeBox; i<=pEye->iSizeBox; i++){
      int y = pEye->GetYcentre() + i;
      if(pEyeDetector->GetEyeFromXslice(y, pEye->GetXcentre(),
                                        pEye->iSizeBox, sEyeBoundaries)){
          output << i << ',' << pEye->GetXcentre() << ',';
          output << sEyeBoundaries.sLeftOrTop.iIS << ',';
          output << sEyeBoundaries.sLeftOrTop.iPI << ',';
          output << sEyeBoundaries.sRightOrBottom.iPI << ',';
          output << sEyeBoundaries.sRightOrBottom.iIS << endl;
        }
      else{
        output << i << ", not found" << endl;
      }
    }

    // now finished with file
    output.close();

    // write out the algorithm y sliced data
    // 1st header
    sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.y.csv", directory,
           iEyeNumber, iImageNumber);
    output.open(ac, ios::binary | ios::trunc | ios::out);
    output << "slice, xc, si, ip, pi, is" << endl;
    // now get the actual data for y slices
    for(int i=-pEye->iSizeBox; i<=pEye->iSizeBox; i++){
      int x = pEye->GetXcentre() + i;
      if(pEyeDetector->GetEyeFromYslice(x, pEye->GetYcentre(),
                                        pEye->iSizeBox, sEyeBoundaries)){
           output << i << ',' << pEye->GetYcentre() << ',';
           output << sEyeBoundaries.sLeftOrTop.iIS << ',';
           output << sEyeBoundaries.sLeftOrTop.iPI << ',';
           output << sEyeBoundaries.sRightOrBottom.iPI << ',';
           output << sEyeBoundaries.sRightOrBottom.iIS << endl;
         }
        else{
          output << i << ", not found" << endl;
        }
    }

    // now finished with file
    output.close();
}
```

EyeDetector.cc

```
// ------------------------------------------------------------------------
// Included files
// ------------------------------------------------------------------------
#include "EyeDetector.h"

// ------------------------------------------------------------------------
// main constructor
// ------------------------------------------------------------------------
CEyeDetector::CEyeDetector(image<unsigned char>* pImgIn)
              : pImg(pImgIn)
{
  // simply set up the deafult values
  ciPI_NEG_ALLOWED = 6;
  ciIS_POS_ALLOWED = 5;
  cfPI_PC_DIFF = 2 * 0.15f;
  cfIS_PC_DIFF = 2 * 0.15f;
  ciHOT_SPOT_VALUE = 170;
  ciDIFF_IGNORE = 75;
}

// ------------------------------------------------------------------------
// given a data set check whether it is a possible eye.
// return false if not, otherwise true and sBoundaries will contain where
// the locations are.
// ------------------------------------------------------------------------
bool CEyeDetector::CheckSlice(int* paiData, const int iSizeData,
                                    SBoundaries& sBoundaries)
{
  // check that the size is sensible
  if(iSizeData < 10)
    return false;

  // check where the reflectivity ends
  int iPosn;
  for(iPosn=0; iPosn<iSizeData-1; iPosn++){
    if((paiData[iPosn]<ciHOT_SPOT_VALUE) &&
           (paiData[iPosn+1]<ciHOT_SPOT_VALUE))
      break;
  }

  // check we found position
  if(iPosn >= iSizeData-1)
    return false;

  // average the two pixels past the hot spot
  int iAverageVal = static_cast<int>(0.50f*paiData[iPosn] +
                                          0.50f*paiData[iPosn+1]);

  // for hot spot values make equal to average
  if(iPosn){
    for(int i=0; i<iPosn; i++)
      paiData[i] = iAverageVal;
  }

  int iStartValue = paiData[0];
  int iPIdiff = static_cast<int>(cfPI_PC_DIFF * iStartValue + 0.5f);
  int iISdiff = 0; // 0 to stop compiler warning
  int iNegCount = ciPI_NEG_ALLOWED;// only allowed so many negative
  int iFoundPosn = -1;  // not found yet
  int iFoundDiff = 0;   // 0 to stop compiler warning
  for(iPosn = 3; iPosn<iSizeData; iPosn++){
    int iDiff = paiData[iPosn-2] + paiData[iPosn-1]
                -paiData[iPosn+2] - paiData[iPosn+1];

    // check not too many negative differences
    if(iDiff < 0){
      if(iNegCount-- == 0)
        return false;
    }

    // if the difference is bigger than start value
```

```
    // ignore as hotspot effect or a value > say 80
    // set to variable ciDIFF_IGNORE
    if((iDiff < iStartValue) && (iDiff < ciDIFF_IGNORE)){
      // have we got value above threshold?
      if(iDiff > iPIdiff){
        iFoundPosn = iPosn;
        iFoundDiff = iDiff;
        iISdiff = static_cast<int>(cfIS_PC_DIFF * paiData[iPosn] + 0.5f);
        break;
      }
    }
  }

  // did we find it?
  if(iFoundPosn == -1)
    return false;    // no

  // we are now looking either for a negative difference
  //(end of high threshold) or for a better value for
  // the found threshold difference (in which case we
  // use that position and reseed iISdiff)
  for(++iPosn; iPosn<iSizeData; iPosn++){
    int iDiff = paiData[iPosn-2] + paiData[iPosn-1]
                -paiData[iPosn+2] - paiData[iPosn+1];
    // check if we have a better boundary
    if(iDiff >= iFoundDiff){
      iFoundPosn = iPosn;
      iFoundDiff = iDiff;
      iISdiff = static_cast<int>(cfIS_PC_DIFF * paiData[iPosn] + 0.5f);
    }
    // check if end of better PI fit
    else if(iDiff < 0){
      // need next diff -ve as well
      if((paiData[iPosn-1] + paiData[iPosn]
            -paiData[iPosn+2] - paiData[iPosn+3]) < 0)
        break;
    }
  }

  // did we find it?
  if(iPosn >= iSizeData)
      return false;    // no

  // save the pupil iris boundary
  sBoundaries.iPI = iFoundPosn;

  // finally looking for negative threshold
  int iPosCount = ciIS_POS_ALLOWED;// only allowed so many +ves
  for(++iPosn; iPosn<iSizeData; iPosn++){
    int iDiff = paiData[iPosn-2] + paiData[iPosn-1]
                -paiData[iPosn+2] - paiData[iPosn+1];
    // we do not allow a positive value greater than iISdiff
    if(iDiff > iISdiff)
      return false;

    // is it a lower positive diff?
    if(iDiff > 0){
      // too many?
      if(iPosCount-- == 0)
        return false;
    }

    // does it exceed negative threshold?
    if(iDiff < -iISdiff){
      sBoundaries.iIS = iPosn++;
      // is next one a better match?
      int iDiff2 = paiData[iPosn-2] + paiData[iPosn-1]
                   -paiData[iPosn+2] - paiData[iPosn+1];
      if(iDiff2 < iDiff)
        sBoundaries.iIS++; // use it
```

```
      return true;  // success!
    }
  }

  // not found
  return false;
}

// ------------------------------------------------------------------------
// given a start position in x,y, get the associated x slice and then get
// PI & IS boundaries for both directions (left and right)
// ------------------------------------------------------------------------
bool CEyeDetector::GetEyeFromXslice(const int iYpos, const int iXcentre,
                                    const int iSize, SEyeBoundaries&
sEyeBoundaries)
{
  // NOTE: we swing +/ iSize around centre

  // ensure we have correctly sized array
  int pai[2*iSize+1];

  // first read the slice in to array
  for(int i=-iSize; i<=iSize; i++)
    pai[i+iSize] = pImg->getPixel(iXcentre+i, iYpos);

  // we can now call common routine to get the data
  return GetEyeFromSliceData(pai, iSize, sEyeBoundaries);
}

// ------------------------------------------------------------------------
// given a start position in x,y, get the associated y slice and then get
// PI & IS boundaries for both directions (top and bottom)
// ------------------------------------------------------------------------
bool CEyeDetector::GetEyeFromYslice(const int iXpos, const int iYcentre,
                                    const int iSize, SEyeBoundaries&
sEyeBoundaries)
{
  // NOTE: we swing +/ iSize around centre

  // ensure we have correctly sized array
  int pai[2*iSize+1];

  // first read the slice in to array
  for(int i=-iSize; i<=iSize; i++)
    pai[i+iSize] = pImg->getPixel(iXpos, iYcentre+i);

  // we can now call common routine to get the data
  return GetEyeFromSliceData(pai, iSize, sEyeBoundaries);
}

// ------------------------------------------------------------------------
// given a slice of data,split it in to left(or top) and right (or bottom)
// and feed it to the CheckSlice routine.  The incoming data has a length
// of 2*iSize+1 where the notional centre split (offset 0) is at
// paiData[iSize]
// ------------------------------------------------------------------------
bool CEyeDetector::GetEyeFromSliceData(const int* pai, const int iSize,
                                       SEyeBoundaries& sEyeBoundaries)
{
  int iStart = iSize;

  // we can now deal with the lhs of the data
  int paiData[2*iStart+1];  // big enough for all
  int iOffset = 0;
  for(int i=iStart; i>=0; i--)
    paiData[iOffset++] = pai[i];

  // can now see if we can get boundaries
  if(!CheckSlice(paiData, iStart+1, sEyeBoundaries.sLeftOrTop))
    return false;
```

```
  // now set up data for rhs
  iOffset = 0;
  for(int i=iStart; i<=2*iSize; i++)
    paiData[iOffset++] = pai[i];

  // can now see if we can get boundaries
  if(!CheckSlice(paiData, 2*iSize-iStart+1, sEyeBoundaries.sRightOrBottom))
    return false;

  // we now need to adjust the points based upon the true start
  // and not our adjusted one
  sEyeBoundaries.sLeftOrTop.iPI += (iStart-iSize);
  sEyeBoundaries.sLeftOrTop.iIS += (iStart-iSize);
  sEyeBoundaries.sRightOrBottom.iPI += (iStart-iSize);
  sEyeBoundaries.sRightOrBottom.iIS += (iStart-iSize);
  return true;
}
```

EyeDetector.h

```
#ifndef EyeDetector_CW_H
#define EyeDetector_CW_H

#include "image_mesaoutput.h"
using namespace mimas;

// ---------------------------------------------------------------------
// structure to hold found pupil/iris and iris/sclera offsets
// ---------------------------------------------------------------------
struct SBoundaries
{
  // #####################################################################
  // functions
  // #####################################################################

  // -----------
  // constructor
  // -----------
  SBoundaries() : iPI(-1), iIS(-1) {}

  // #####################################################################
  // variables
  // #####################################################################
  int iPI;  // pupil/iris
  int iIS;  // iris/sclera
};

// ---------------------------------------------------------------------
// structure to hold found pupil/iris and iris/sclera offsets for both
// lhs (or top) and rhs (or bottom) of an eye
// ---------------------------------------------------------------------
struct SEyeBoundaries
{
  // #####################################################################
  // variables
  // #####################################################################
  SBoundaries sLeftOrTop;
  SBoundaries sRightOrBottom;
};

// ---------------------------------------------------------------------
// The actual detector class - routines left public for now for
// ease of testing
// ---------------------------------------------------------------------
class CEyeDetector
{
  public:
    // ###################################################################
    // functions
    // ###################################################################

    // -----------
    // constructor
    // -----------
    CEyeDetector(image<unsigned char>* pImgIn);

    // -----------------------------------------------
    // given a start position in x,y, get the associated
    // x slice and then get PI & IS boundaries for
    // both directions (left and right)
    // -----------------------------------------------
    bool GetEyeFromXslice(const int iYpos, const int iXcentre,
                          const int iSize, SEyeBoundaries& sEyeBoundaries);

    // -----------------------------------------------
    // given a start position in x,y, get the associated
    // y slice and then get PI & IS boundaries for
    // both directions (top and bottom)
    // -----------------------------------------------
    bool GetEyeFromYslice(const int iXpos, const int iYcentre,
                          const int iSize, SEyeBoundaries& sEyeBoundaries);
```

```
    // ---------------------------------------------------------
    // given a slice of data,split it in to left(or top) and
    // right (or bottom) and feed it to the CheckSlice routine.
    // ---------------------------------------------------------
    bool GetEyeFromSliceData(const int* pai, const int iSize,
                             SEyeBoundaries& sEyeBoundaries);

    // -------------------------------------------------
    // given a data set check whether it is a possible eye
    // return false if not, otherwise true and sBoundaries
    // will contain where the locations are.  NOTE:  the
    // input data will be altered.
    // -------------------------------------------------
    bool CheckSlice(int* paiData, const int iSizeData,
                    SBoundaries& sBoundaries);

    // ########################################################################
    // variables
    // ########################################################################

    // -------------------------------------------------------
    // constants used in program - actually set as variables
    // so at a later date we will allow the user/program
    // to be able to dynamically alter them
    // -------------------------------------------------------

    // -------------------------------------------------------
    // number of negative differences allowed whilst looking
    // for pupil/iris boundary
    // -------------------------------------------------------
    int ciPI_NEG_ALLOWED;

    // -------------------------------------------------------
    // number of positive differences allowed whilst looking
    // for iris/sclera boundary
    // -------------------------------------------------------
    int ciIS_POS_ALLOWED;

    // -------------------------------------------------------
    // the percentage differences (remember x2) for threshold
    // of pupil/iris
    // -------------------------------------------------------
    float cfPI_PC_DIFF;

    // -------------------------------------------------------
    // the percentage differences (remember x2) for threshold
    // of iris/sclera
    // -------------------------------------------------------
    float cfIS_PC_DIFF;

    // ------------------------------
    // the level of the reflectivity
    // ------------------------------
    int ciHOT_SPOT_VALUE;

    // -------------------------------------------------
    // when starting search ignore if difference is above
    // this value (hot spot effects)
    // -------------------------------------------------
    int ciDIFF_IGNORE;

  private:
    // ----------------------------------
    // pointer to where raw image data is
    // ----------------------------------
    image<unsigned char>* pImg;
};

#endif
```

Specific.cc changes from Appendix 5

- New Description

```
/*
 Given an x,y coordinate the CEyeDetector class has been written to
 have a Validate(SEyeData& sEyeData) function.  The sEyeData must have
 set up variables iXstart (xC), iYstart (yC), iSize before calling it.

 This function gathers data slices in x starting (from xC-iSze to xC+iSze)
 from y=yC+0 .. y=yC-size and passes to the GetEyeFromXslice slice routine
 to validate if eye slice.  If so the PI,IS is pushed to the front of dequeX
 of SEyeBoundaries (which has a iSliceNum for higher level uses).  If we
 have three failures then we stop the scan as we presumably have moved out
 of the eye.

 We then perform the above from y=xC+1 .. y=xC+iSze but this time
 push to the back of the deque (as this will preserve the order if
 we want to perform later statistics).

 The above is repeated for yslices from x=xC+0.. x=xC-iSze and for
 x=xC+1.. xC+iSze.

 Here we do not do any checking but simply allow the specific routine
 to print out the relevant data
*/
```

- Changed WriteSliceData routine

```
void WriteSliceData(image<unsigned char>& imgRaw, int iEyeNumber,
                    const char* directory, int iImageNumber)
{
  ofstream output;

  // form filename and path, set default output
  char ac[500];
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  // get pointer to correct eye
  SEyeTestData* pEye = (iEyeNumber == 1) ? &testData.testEye1 :
                                           &testData.testEye2;

  // ensure that at least centre value is defined
  if((pEye->aiXposns[2] == -1) || (pEye->aiYposns[2] == -1))
    return;

  // write header for x direction
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  WriteSliceHeader(output, pEye, true, imgRaw);
```

Specific.cc changes from Appendix 5

```
// now finished with file
output.close();

// write header for y direction
sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
       iEyeNumber, iImageNumber);
output.open(ac, ios::binary | ios::trunc | ios::out);
WriteSliceHeader(output, pEye, false, imgRaw);

// now finished with file
output.close();

// set up data and call validate
SEyeData sEyeData;
sEyeData.iXstart =  pEye->GetXcentre();
sEyeData.iYstart =  pEye->GetYcentre();
sEyeData.iSize =  pEye->iSizeBox;
bool fGood = pEyeDetector->Validate(sEyeData);

// write out the algorithm x sliced data
// 1st header
sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.x.csv", directory,
       iEyeNumber, iImageNumber);
output.open(ac, ios::binary | ios::trunc | ios::out);
output << "slice, xc, si, ip, pi, is" << endl;
int iCentre = pEye->aiXposns[2];
output << "user," << iCentre << ',';
output << iCentre - pEye->aiXposns[0] << ',';
output << iCentre - pEye->aiXposns[1] << ',';
output << pEye->aiXposns[3] - iCentre << ',';
output << pEye->aiXposns[4] - iCentre << endl;
// now cycle the found data
std::deque<SEyeBoundaries>::iterator it;
if(fGood){
  for(it=sEyeData.dqX.begin(); it!=sEyeData.dqX.end(); it++){
       output << it->iSliceNum << ',' << pEye->GetXcentre() << ',';
       output << it->sLeftOrTop.iIS << ',';
       output << it->sLeftOrTop.iPI << ',';
       output << it->sRightOrBottom.iPI << ',';
       output << it->sRightOrBottom.iIS << endl;
    }
}
else{
  output << "Invalid x,y locations" << endl;
}
// now finished with file
output.close();

// write out the algorithm y sliced data
// 1st header
sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.y.csv", directory,
       iEyeNumber, iImageNumber);
output.open(ac, ios::binary | ios::trunc | ios::out);
output << "slice, xc, si, ip, pi, is" << endl;
iCentre = pEye->aiYposns[2];
output << "user," << iCentre << ',';
output << iCentre - pEye->aiYposns[0] << ',';
output << iCentre - pEye->aiYposns[1] << ',';
output << pEye->aiYposns[3] - iCentre << ',';
output << pEye->aiYposns[4] - iCentre << endl;
// now cycle the found data
if(fGood){
  for(it=sEyeData.dqY.begin(); it!=sEyeData.dqY.end(); it++){
    output << it->iSliceNum << ',' << pEye->GetYcentre() << ',';
    output << it->sLeftOrTop.iIS << ',';
    output << it->sLeftOrTop.iPI << ',';
    output << it->sRightOrBottom.iPI << ',';
    output << it->sRightOrBottom.iIS << endl;
  }
}
```

Specific.cc changes from Appendix 5

```
  else{
    output << "Invalid x,y locations" << endl;
}
  // now finished with file
  output.close();
}
```

EyeDetector.cc changes from Appendix 5

- Added new Validate routine

```
// ---------------------------------------------------------------------
// from given x, y and size (in SEyeData) gather slices in x & y directions.
// Check if valid slices and come up with best fit for centres.
// ---------------------------------------------------------------------
bool CEyeDetector::Validate(SEyeData& sEyeData)
{
  // check that size is sensible
  int iSize = sEyeData.iSize;
  if(iSize < 10)
    return false;

  // check that the x value is sensible
  if((sEyeData.iXstart < iSize) || (sEyeData.iXstart > (pImg->getWidth()-iSize)))
    return false;

  // check that the y value is sensible
  if((sEyeData.iYstart < iSize) || (sEyeData.iYstart > (pImg->getHeight()-iSize)))
    return false;

  // ensure that the incoming deques are cleared
  sEyeData.dqX.clear();
  sEyeData.dqY.clear();

  // gather and check x slices from iYstart+0 .. iYstart-iSize
  SEyeBoundaries sEyeBoundaries;
  int iFailuresAllowed = 3;
  for(int i=0; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = -i;
    if(GetEyeFromXslice(sEyeData.iYstart-i, sEyeData.iXstart, iSize,
                        sEyeBoundaries))
      sEyeData.dqX.push_front(sEyeBoundaries);
    else{
      if(iFailuresAllowed-- == 0)
        break;
    }
  }

  // now from iYStart+1 .. iYstart+iSize
  iFailuresAllowed = 3;
  for(int i=1; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = i;
    if(GetEyeFromXslice(sEyeData.iYstart+i, sEyeData.iXstart, iSize,
                        sEyeBoundaries))
      sEyeData.dqX.push_back(sEyeBoundaries); // to preserve order
    else{
      if(iFailuresAllowed-- == 0)
        break;
    }
  }

  // gather and check y slices from iXstart+0 .. iXstart-iSize
  iFailuresAllowed = 3;
  for(int i=0; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = -i;
    if(GetEyeFromYslice(sEyeData.iXstart-i, sEyeData.iYstart, iSize,
                        sEyeBoundaries))
      sEyeData.dqY.push_front(sEyeBoundaries);
    else{
      if(iFailuresAllowed-- == 0)
        break;
    }
  }

  // now from iXStart+1 .. iXstart+iSize
  iFailuresAllowed = 3;
  for(int i=1; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = i;
    if(GetEyeFromYslice(sEyeData.iXstart+i, sEyeData.iYstart, iSize,
                        sEyeBoundaries))
```

EyeDetector.cc changes from Appendix 5

```
        sEyeData.dqY.push_back(sEyeBoundaries); // to preserve order
      else{
        if(iFailuresAllowed-- == 0)
          break;
      }
    }

    // later we will add checks in here
    return true;
  }
```

EyeDetector.h changes from Appendix 5

- Added new include

```
#include <deque>
```

- Added new variable to SEyeBoundaries structure

```
int iSliceNum;// for higher levels to set/use
```

- Added new structure

```
// ------------------------------------------------------------------------
// Structure to hold data for an eye and provide routines to get the data
// ------------------------------------------------------------------------
struct SEyeData
{
  // ####################################################################
  // functions
  // ####################################################################

  // -----------
  // constructor
  // -----------
  SEyeData() : iXstart(-1), iYstart(-1) {}

  // ------------------------------------------------------------------------
  // get the true x centre in to supplied variable, return false if not valid
  // ------------------------------------------------------------------------
  bool GetXcentre(int& x){
    if((iXstart == -1) || (iYstart == -1))
      return false;
    x = iXstart;
    return true;
  }

  // ------------------------------------------------------------------------
  // get the true y centre in to supplied variable, return false if not valid
  // ------------------------------------------------------------------------
  bool GetYcentre(int& y){
    if((iXstart == -1) || (iYstart == -1))
      return false;
    y = iYstart;
    return true;
  }

  // ####################################################################
  // variables - user supplied
  // ####################################################################
  int iXstart,iYstart;          // start positions to search from
  int iSize;  // number of samples either side of starts to use

  // ####################################################################
  // variables - eye detector sets
  // ####################################################################

  // ---------------------------------------------------------------
  // these did not have to be in here but are needed and as we may want
  // to check the data this is the best place to put them
  // ---------------------------------------------------------------
  std::deque<SEyeBoundaries> dqX;  // for x sliced data
  std::deque<SEyeBoundaries> dqY;  // for y sliced data
};
```

- Added validate routine definition

```
// ---------------------------------------------------
// from given x, y and size (in SEyeData) gather slices
// in x & y directions.  Check if valid slices and come
// up with best fit for centres.
// ---------------------------------------------------
bool Validate(SEyeData& sEyeData);
```

Specific.cc changes from Appendix 6

- New Description

```
/*
 Using the calculate function(currently by the user press either 'c'
 for any point on the screen at which they want to test) as well as
 when the user presses F9 to define an eye centre,

 We gather slices in the x direction with the x fixed and y varied and
 then find the boundaries.  If boundaries are found we put them in a
 deque for sanity checks later.

 We then father slices in the y direction with y fixed and x varied and
 then find the boundaries.  If boundaries are found we put themin a
 different (y) deque for sanity checking.

  Simple sanity tests applied before we use the data for adjusting the
 input x or y position.

 (1)  If there are less than 3 valid data slices in the deque the we set the
       confidence state to L1 and go to step 4

 (2) The deque is reordered based on smallest to largest value of LHS
       Pupil/Iris boundary.  The 3/4 offset value is then used and any PI values
       that are <0.7% of it or bigger than 1.5% of it are deleted from the set.
       If the set would be less than three then the step 2 set is reinstated
       and confidence value set to L2 before proceeding to step 4

 (3) The deque is reordered based on smallest to largest value of RHS
       Pupil/Iris boundary.  The 3/4 offset value is then used and any PI values
       that are <0.7% of it or bigger than 1.5% of it are deleted from the set.
       If the set would be less than three then the step 3 set is reinstated
       and confidence value set to L3 otherwise we set the confidence level
       to L4 (highest).

 (4) The incoming set is used to simply add all LH PI values together (sumLHS)
       and then add all RH PI together (sumRHS) using this we can set an
       adjustment level  simply from (sumRHS - sumLHS)/2.
*/
```

- Altered DoUpdate to allow for new ‘c’ key function

```
void DoUpdate(const int iKey, image<unsigned char>& imgMod)
{
  // where the F keys map into the x or y array
  static const int caiNORM_F_KEYS[8] = {0,1,3,4,0,1,3,4};

  // get a pointer to the eye we are working on
  SEyeTestData* pEye = (testData.fEye1Active) ?  &testData.testEye1 :
                                                 &testData.testEye2;

  strMessage = "OK"; // assume we will be ok
  // check initially for the F1..F8 keys

  // check if defining user points
  if((iKey >= KEY_F(1)) && iKey <= KEY_F(8)){
    // get pointer to data to check.alter
    int* pai;
    int iPosn;
    if(iKey < KEY_F(5)){
      pai = pEye->aiXposns;
      iPosn = iXpos;
    }
    else{
      pai = pEye->aiYposns;
      iPosn = iYpos;
    }
    // get offset in to array
    int iOffset = caiNORM_F_KEYS[iKey-KEY_F(1)];

    // the centre point must be defined before anything else
```

Specific.cc changes from Appendix 6

```
      if(pai[2] == -1)
        strMessage = "Must define centre point first";
      else{
        // must be within the user bounding box
        if((iPosn < (pai[2]-pEye->iSizeBox)) ||
                (iPosn > (pai[2]+pEye->iSizeBox)))
          strMessage = "Must be within box";
        else{
          // check if greater than previous one
          if(iOffset && (iPosn <= pai[iOffset-1]))
            strMessage = "must be greater than previous value";
          //check if less than next one
          else if((iOffset < SEyeTestData::eNumIntPosns-1) &&
                    (iPosn >= pai[iOffset+1]) && (pai[iOffset+1] != -1))
            strMessage = "must be less than next value";
          else
            pai[iOffset] = iPosn;// success!
        }
      }
    }
    else{
      // can now check other keys
      switch(iKey){
        case KEY_F(9):// set centre on active eye
          // check that position is at least valid
          if((iXpos > pEye->iSizeBox) && (iYpos > pEye->iSizeBox)){
            pEye->NewValue(iXpos, iYpos);
            pEye->CalcCentres(pEyeDetector);
          }
          else
            beep();
          break;
        case KEY_F(11):
          testData.fEye1Active = true;
          break;
        case KEY_F(12):
          testData.fEye1Active = false;
          break;
        case 'c':
          testData.testPos.NewValue(iXpos, iYpos);
          testData.testPos.CalcCentres(pEyeDetector);
          break;
      }
    }

    // show user points on the image
    ShowUserEyeDataOnImage(testData.testEye1, imgMod);
    ShowUserEyeDataOnImage(testData.testEye2, imgMod);

    // and show data on command line
    ShowCursorPosn(imgMod);
  }
```

- Altered ImageLoaded to call CalcCentres

```
  void ImageLoaded(image<unsigned char>& imgMod, const char* directory,
                   int iFileNumber)
  {
    // get screen limits
    iXposMax = imgMod.getWidth();
    iYposMax = imgMod.getHeight();

    // set deafult positions
    iXpos = iXposMax/2;
    iYpos = iYposMax/2;

    // clear old test data
    testData.Clear();

    // read in any associated test data (if any)
    ifstream input;
```

Specific.cc changes from Appendix 6

```
  char ac[800];
  sprintf(ac, "%s/ProgData/Image%04d.prgData", directory, iFileNumber);
  input.open(ac, ios::binary);
  if(input){
    // read in data
    input.read((char*)&testData, sizeof(testData));
    input.close();

    // set last x,y position as current
    iXpos = testData.iSavedXpos;
    iYpos = testData.iSavedYpos;
  }

  // update the calculated data for the eyes
  testData.testEye1.CalcCentres(pEyeDetector);
  testData.testEye2.CalcCentres(pEyeDetector);

  // actually draws our data
  DoUpdate(-1, imgMod);
}
```

- ShowInfo changed to show extra data for 'c' key press

```
void ShowInfo(const char* file)
{
  clear();
  printw("%s\n", file);
  printw("Current Eye Actions on Eye (%d)\n", (testData.fEye1Active)? 1:2);
  printw("X=%i, y=%i, value=%d\n", iXpos, iYpos, iPixelValue);
  ShowUserVsProg(1);
  ShowUserVsProg(2);
  ShowUserVsProg(3);   // user pressing c as location
  printw("Info: [%s]\n", strMessage.c_str());
  refresh();
}
```

- Added routine to show calculated data

```
// ---------------------------------------------------------------------------
// Show the calculated data to the user
// ---------------------------------------------------------------------------
void ShowUserVsProg(int iEye)
{
 SEyeTestData* pEye=NULL;

 // get correct eye pointer
 if(iEye == 3){
   // cursor test eye
   pEye = &testData.testPos;
   int x = pEye->GetXcentre();
   int y = pEye->GetYcentre();
   if((x != -1) && (y !=-1))
     printw("Position (%d, %d) ", x,y);
 }
 else{
   // user defined eye 1 or eye 2
   pEye = (iEye == 1) ? &testData.testEye1 : &testData.testEye2;
   printw("Eye%d:=>  User x = %d, User y = %d\n         ", iEye,
           pEye->GetXcentre(),
           pEye->GetYcentre());
 }

 // show the calculated x centre
 printw("Prog x = ");
 float fVal;
 if(pEye->sEyeData.sEyeX.GetCentre(fVal))
   printw("%.1f conf(%d)",fVal,  pEye->sEyeData.sEyeX.eConfidence);
 else
   printw("INVALID");

 // show the calculated y centre
```

```
  printw(", Prog y = ");
  if(pEye->sEyeData.sEyeY.GetCentre(fVal))
    printw("%.1f conf(%d)", fVal,  pEye->sEyeData.sEyeY.eConfidence);
  else
    printw("INVALID");

  // state whether this is to be treated as an eye
  if(pEye->sEyeData.IsItAnEye()) // have run eye detector so ok to call
    printw(" [AN EYE!!]\n");
  else
    printw (" [Not an eye]\n");
 }
```

- Altered WriteSliceData routine

```
void WriteSliceData(image<unsigned char>& imgRaw, int iEyeNumber,
                    const char* directory, int iImageNumber)
{
  ofstream output;

  // form filename and path, set default output
  char ac[500];
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  output << "Please enter at least centre value";
  output.close();

  // get pointer to correct eye
  SEyeTestData* pEye = (iEyeNumber == 1) ? &testData.testEye1 :
                                           &testData.testEye2;

  // ensure that we have a centre point
  if((pEye->aiXposns[2] == -1) || (pEye->aiYposns[2] == -1))
    return;

  // write header for x direction
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.x.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  WriteSliceHeader(output, pEye, true, imgRaw);

  // now finished with file
  output.close();

  // write header for y direction
  sprintf(ac, "%s/Eye%dData/SliceData/Image%04d.y.csv", directory,
         iEyeNumber, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  WriteSliceHeader(output, pEye, false, imgRaw);

  // now finished with file
  output.close();
```

Specific.cc changes from Appendix 6

```
// run the algorithm (overkill as should be already valid)
bool fGood = pEye->CalcCentres(pEyeDetector);

// write out the algorithm x sliced data
// 1st header
sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.x.csv", directory,
       iEyeNumber, iImageNumber);
output.open(ac, ios::binary | ios::trunc | ios::out);
output << "slice, xc, si, ip, pi, is" << endl;
int iCentre = pEye->aiXposns[2];
output << "user," << iCentre << ',';
if(pEye->aiXposns[0] > 0)
  output << iCentre - pEye->aiXposns[0];
output << ',';
if(pEye->aiXposns[1] > 0)
  output << iCentre - pEye->aiXposns[1];
output << ',';
if(pEye->aiXposns[3] > 0)
  output << pEye->aiXposns[3] - iCentre;
output << ',';
if(pEye->aiXposns[4] > 0)
  output << pEye->aiXposns[4] - iCentre;
output << endl;

// now cycle the found data
float fVal =10000; // easy to know if error!
std::deque<SEyeBoundaries>::iterator it;
if(fGood){
  for(it=pEye->sEyeData.sEyeX.dq.begin();
      it!=pEye->sEyeData.sEyeX.dq.end(); it++){
       output << it->iSliceNum << ',' << pEye->GetXcentre() << ',';
       output << it->sLeftOrTop.iIS << ',';
       output << it->sLeftOrTop.iPI << ',';
       output << it->sRightOrBottom.iPI << ',';
       output << it->sRightOrBottom.iIS << endl;
    }
    if(pEye->sEyeData.sEyeX.GetCentre(fVal))
      output << "Calculated x centre is " << fVal << endl;
    else
      output << "No calculated x centre" << endl;

}
else{
  output << "Not Eye" << endl;
}
// now finished with file
output.close();

// write out the algorithm y sliced data
// 1st header
sprintf(ac, "%s/Eye%dData/AlgData/Image%04d.y.csv", directory,
       iEyeNumber, iImageNumber);
output.open(ac, ios::binary | ios::trunc | ios::out);
output << "slice, xc, si, ip, pi, is" << endl;
iCentre = pEye->aiYposns[2];
output << "user," << iCentre << ',';
if(pEye->aiYposns[0] > 0)
  output << iCentre - pEye->aiYposns[0];
output << ',';
if(pEye->aiYposns[1] > 0)
  output << iCentre - pEye->aiYposns[1];
output << ',';
if(pEye->aiYposns[3] > 0)
  output << pEye->aiYposns[3] - iCentre;
output << ',';
if(pEye->aiYposns[4] > 0)
  output << pEye->aiYposns[4] - iCentre;
// now cycle the found data
output << endl;
if(fGood){
```

Specific.cc changes from Appendix 6

```
    for(it=pEye->sEyeData.sEyeY.dq.begin();
        it!=pEye->sEyeData.sEyeY.dq.end(); it++){
      output << it->iSliceNum << ',' << pEye->GetYcentre() << ',';
      output << it->sLeftOrTop.iIS << ',';
      output << it->sLeftOrTop.iPI << ',';
      output << it->sRightOrBottom.iPI << ',';
      output << it->sRightOrBottom.iIS << endl;
    }
    if(pEye->sEyeData.sEyeY.GetCentre(fVal))
      output << "Calculated y centre is " << fVal << endl;
    else
      output << "No calculated y centre" << endl;
  }
  else{
    output << "Not Eye" << endl;
  }
  // now finished with file
  output.close();
}
```

EyeDetector.cc changes from Appendix 6

- Added public CalcCentres routine

```
// ------------------------------------------------------------------------
// Main routine to call to calculate (from given x, y) whether the point
// is in an eye and if so where the pupil centres are.
// ------------------------------------------------------------------------
bool CEyeDetector::CalcCentres(const int iXstart, const int iYstart,
                               const int iSize, SEyeData& sEyeData)
{
  // calculate for x slices
  CalcCentre(iXstart, iYstart, iSize, true, sEyeData.sEyeX);

  // and calculate for y slices
  CalcCentre(iXstart, iYstart, iSize, false, sEyeData.sEyeY);

  // inform caller whether we think it is an eye
  return ((sEyeData.sEyeX.eConfidence == L4) &&
          (sEyeData.sEyeY.eConfidence == L4));
}
```

- Added private CalcCentre routine

```
// ------------------------------------------------------------------------
// From given starts gather slices (direction fX), validate and return
// whether the slices form a possible eye
// ------------------------------------------------------------------------
bool CEyeDetector::CalcCentre(const int iXstart, const int iYstart,
                              const int iSize, bool fX,
                              SBasicEyeData& sBasicEyeData)
{
  // for now simply choose direction to gather slices in
  if(fX)
    return ValidateX(iXstart, iYstart, iSize, sBasicEyeData);
  else
    return ValidateY(iXstart, iYstart, iSize, sBasicEyeData);
}
```

- Removed test routines from GetEyeFromSliceData

```
bool CEyeDetector::GetEyeFromSliceData(const int* pai, const int iSize,
                                       SEyeBoundaries& sEyeBoundaries)
{
 // we can now deal with the lhs of the data
 int paiData[2*iSize+1]; // big enough for all
 int iOffset = 0;
 for(int i=iSize; i>=0; i--)
   paiData[iOffset++] = pai[i];

 // can now see if we can get boundaries
 if(!CheckSlice(paiData, iSize+1, sEyeBoundaries.sLeftOrTop))
   return false;

 // now set up data for rhs
 iOffset = 0;
 for(int i=iSize; i<=2*iSize; i++)
   paiData[iOffset++] = pai[i];

 // can now see if we can get boundaries
 if(!CheckSlice(paiData, iSize+1, sEyeBoundaries.sRightOrBottom))
   return false;

 return true;
}
```

- altered Validate routine to split into separate routines

```
// ------------------------------------------------------------------------
// from given x, y and size (in SEyeData) gather slices in x & y directions.
// Check if valid slices and come up with best fit for centres.
```

EyeDetector.cc changes from Appendix 6

```
// ---------------------------------------------------------------------
bool CEyeDetector::Validate(SEyeData& sEyeData)
{
  // ensure no confidence at all initially
  sEyeData.sEyeX.eConfidence = sEyeData.sEyeY.eConfidence = NONE;

  if(ValidateX(sEyeData.iXstart, sEyeData.iYstart,
     sEyeData.iSize, sEyeData.sEyeX)){
    return ValidateY(sEyeData.iXstart, sEyeData.iYstart,
                     sEyeData.iSize, sEyeData.sEyeY);
  }

  return false;
}

// ---------------------------------------------------------------------
// from given x, y and size gather slices in x direction. Check if valid
// slices and come up with best fit for x centre from incoming x.
// ---------------------------------------------------------------------
bool CEyeDetector::ValidateX(const int x,const int y, const int iSize,
                             SBasicEyeData& sBasicEyeData)
{
  // default is no confidence
  sBasicEyeData.eConfidence = NONE;

  // check that size is sensible
  if(iSize < 10)
    return false;

  // move in the user variables
  sBasicEyeData.iStart = x;

  // check that the x value is sensible
  if((sBasicEyeData.iStart < iSize) ||
       (sBasicEyeData.iStart > (pImg->getWidth()-iSize)))
    return false;

  // ensure that the incoming deque is cleared
  sBasicEyeData.dq.clear();

  // gather and check x slices from iYstart+0 .. iYstart-iSize
  SEyeBoundaries sEyeBoundaries;
  int iFailuresAllowed = 3;
  for(int i=0; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = -i;
    if(GetEyeFromXslice(y-i, x, iSize, sEyeBoundaries))
      sBasicEyeData.dq.push_front(sEyeBoundaries);
    else{
      if(iFailuresAllowed-- == 0)
        break;
    }
  }

  // now from iYStart+1 .. iYstart+iSize
  iFailuresAllowed = 3;
  for(int i=1; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = i;
    if(GetEyeFromXslice(y+i, x, iSize, sEyeBoundaries))
      sBasicEyeData.dq.push_back(sEyeBoundaries);  // to preserve order
    else{
      if(iFailuresAllowed-- == 0)
        break;
    }
  }

  // get rid of bad data and validate
  ValidateBoundaries(sBasicEyeData);
  return true;
}

// ---------------------------------------------------------------------
```

EyeDetector.cc changes from Appendix 6

```
// from given x, y and size gather slices in y direction. Check if valid
// slices and come up with best fit for y centre from incoming y.
// ---------------------------------------------------------------------
bool CEyeDetector::ValidateY(const int x,const int y, const int iSize,
                                    SBasicEyeData& sBasicEyeData)
{
  // default is no confidence
  sBasicEyeData.eConfidence = NONE;

  // check that size is sensible
  if(iSize < 10)
    return false;

  // move in the user variables
  sBasicEyeData.iStart = y;

  // check that the y value is sensible
  if((sBasicEyeData.iStart < iSize) ||
       (sBasicEyeData.iStart > (pImg->getHeight()-iSize)))
    return false;

  // ensure that the incoming deque is cleared
  sBasicEyeData.dq.clear();

  // gather and check y slices from iXstart+0 .. iXstart-iSize
  SEyeBoundaries sEyeBoundaries;
  int iFailuresAllowed = 3;
  for(int i=0; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = -i;
    if(GetEyeFromYslice(x-i, y, iSize, sEyeBoundaries))
      sBasicEyeData.dq.push_front(sEyeBoundaries);
    else{
      if(iFailuresAllowed-- == 0)
        break;
    }
  }

  // now from iXStart+1 .. iXstart+iSize
  iFailuresAllowed = 3;
  for(int i=1; i<=iSize; i++){
    sEyeBoundaries.iSliceNum = i;
    if(GetEyeFromYslice(x+i, y, iSize, sEyeBoundaries))
      sBasicEyeData.dq.push_back(sEyeBoundaries);  // to preserve order
    else{
      if(iFailuresAllowed-- == 0)
        break;
    }
  }

  // get rid of bad data and validate
  ValidateBoundaries(sBasicEyeData);
  return true;
}
```

- Added validation of boundaries to give some confidence factor

```
// ---------------------------------------------------------------------
// parse and validate found slice, calc offset and return some level of
// confidence in the data
// ---------------------------------------------------------------------
void CEyeDetector::ValidateBoundaries(SBasicEyeData& sBasicEyeData)
{
  deque<SEyeBoundaries> dqTemp;
  deque<SEyeBoundaries>::iterator it;

  // need at least one slice!
  if(sBasicEyeData.dq.empty()){
    sBasicEyeData.eConfidence = NONE;
    return;
```

```
    }

    // start at lowest confidence
    sBasicEyeData.eConfidence = L1;

    // need at least 3 slices
    if(sBasicEyeData.dq.size() >= 3){
      // set new confidence level
      sBasicEyeData.eConfidence = L2;

      // copy raw data and sort on lhs
      sort(sBasicEyeData.dq.begin(), sBasicEyeData.dq.end(), EyeSortLHS);

      // now remove values that are not within -50% or +80% of 3/4 offset value
      // (gives more bias to the higher numbers)
      int iCheckVal = sBasicEyeData.dq[3*sBasicEyeData.dq.size()/4].sLeftOrTop.iPI;
      int iCheckHigh = static_cast<int>(1.8 * iCheckVal);
      int iCheckLow = static_cast<int>(0.5 * iCheckVal);
      for(it=sBasicEyeData.dq.begin(); it != sBasicEyeData.dq.end(); it++){
        if((it->sLeftOrTop.iPI >= iCheckLow) &&
             (it->sLeftOrTop.iPI <= iCheckHigh)){
          dqTemp.push_back(*it);
        }
      }

      // again need at least three slices
      if(dqTemp.size() >= 3){
        // set new confidence level
        sBasicEyeData.eConfidence = L3;

        // make temp current and sort temp on rhs
        sBasicEyeData.dq = dqTemp;
        dqTemp.clear();
        sort(sBasicEyeData.dq.begin(), sBasicEyeData.dq.end(), EyeSortRHS);

        // now remove values that are not within -50% or +80% of 3/4 offset value
        // (gives more bias to the higher numbers)
        int iCheckVal =
               sBasicEyeData.dq[3*sBasicEyeData.dq.size()/4].sRightOrBottom.iPI;
        int iCheckHigh = static_cast<int>(1.8 * iCheckVal);
        int iCheckLow = static_cast<int>(0.5 * iCheckVal);
        for(it=sBasicEyeData.dq.begin(); it != sBasicEyeData.dq.end(); it++){
          if((it->sRightOrBottom.iPI >= iCheckLow) &&
               (it->sRightOrBottom.iPI <= iCheckHigh)){
            dqTemp.push_back(*it);
          }
        }

        // again require 3 slicesfor better confidence
        if(dqTemp.size() >= 3){
        // set new confidence level
          sBasicEyeData.eConfidence = L4;

        // make temp current
          sBasicEyeData.dq = dqTemp;
        }
      }
    }

    // ensure we are ordered by slice in all cases
    sort(sBasicEyeData.dq.begin(), sBasicEyeData.dq.end(), EyeSortSlice);

    // calc the adjustment
    sBasicEyeData.fAdj = 0;
    for(it=sBasicEyeData.dq.begin(); it != sBasicEyeData.dq.end(); it++)
      sBasicEyeData.fAdj += (it->sRightOrBottom.iPI - it->sLeftOrTop.iPI)/2.0f;
    sBasicEyeData.fAdj = sBasicEyeData.fAdj/sBasicEyeData.dq.size();
  }
```

- Added sort routines

EyeDetector.cc changes from Appendix 6

```
// ------------------------------------------------------------------------
// Sort routines
// ------------------------------------------------------------------------
bool EyeSortLHS(const SEyeBoundaries& slice1, const SEyeBoundaries& slice2)
{
  return slice1.sLeftOrTop.iPI < slice2.sLeftOrTop.iPI;
}

bool EyeSortRHS(const SEyeBoundaries& slice1, const SEyeBoundaries& slice2)
{
  return slice1.sRightOrBottom.iPI < slice2.sRightOrBottom.iPI;
}

bool EyeSortSlice(const SEyeBoundaries& slice1, const SEyeBoundaries& slice2)
{
  return slice1.iSliceNum < slice2.iSliceNum;
}
```

EyeDetector.h changes from Appendix 6

- Added confidence levels

```
// ------------------------------------------------------------------------
// our defined confidence levels
// ------------------------------------------------------------------------
enum EConfidence{NONE, L1, L2, L3, L4};
```

- Added new struct to hold an eyes worth of data

```
// ------------------------------------------------------------------------
// Structure to hold basic data for an eye slice in either x or y direction
// ------------------------------------------------------------------------
struct SBasicEyeData
{
 // ########################################################################
 // functions
 // ########################################################################

 // -----------
 // constructor
 // -----------
 SBasicEyeData() : eConfidence(NONE){}

 // ---------------------------------------------------------
 // if we have some confidence then get the calculated centre
 // as either a float or an integer
 // ---------------------------------------------------------
 bool GetCentre(float& fRet)
 {
   if(eConfidence == NONE)
     return false;

   fRet = iStart + fAdj;
   return true;
 }

 bool GetCentre(int& iRet)
 {
   float fVal;

   if(GetCentre(fVal)){
     iRet = static_cast<int>(fVal + 0.5f);
     return true;
   }

   return false;
 }


 // ########################################################################
 // variables
 // ########################################################################
 int  iStart;// user given start position
 float fAdj;   // adjustment from start to "true" centre
 EConfidence eConfidence;  // confidence level;

 // ---------------------------------------------------------------------
 // these did not have to be in here but are needed and as we may want
 // to check the data this is the best place to put them
 // ---------------------------------------------------------------------
 std::deque<SEyeBoundaries> dq; // for sliced data
};
```

- Simplified SEyeData to use above structure

```
struct SEyeData
{
  // ########################################################################
  // functions
  // ########################################################################
```

EyeDetector.h changes from Appendix 6

```
  // ----------------------------------
  // check if we are deemed to be an eye
  // ----------------------------------
  bool IsItAnEye() // ensure used pEyeDetector to set it 1st!
  {
    if((sEyeX.eConfidence == L4) && (sEyeY.eConfidence == L4))
      return true;

    return false;
  }

  // ########################################################################
  // variables - user supplied
  // ########################################################################
  int iXstart,iYstart; // start positions to search from
  int iSize;  // number of samples either side of starts to use

  // ########################################################################
  // variables - eye detector sets
  // ########################################################################
  SBasicEyeData sEyeX, sEyeY;
};
```

- Added new routine definitions to CEyeDetector and made more private

```
class CEyeDetector
{
  public:
    // ########################################################################
    // Public functions
    // ########################################################################

    // ----------------
    // main constructor
    // ----------------
    CEyeDetector(image<unsigned char>* pImgIn);

    // ----------------------------------------------------------------
    // given an x,y position and size around it to check, see if we can
    // can find an eye, return true if found and confidence levels for
    // both x & y directions are > L4
    // ----------------------------------------------------------------
    bool CalcCentres(const int iXstart, const int iYstart,
                     const int iSize, SEyeData& sEyeData);

    // ########################################################################
    // Private functions
    // ########################################################################
  private:
    // ------------------------------------------------------------
    // From given starts gather slices (direction fX), validate and
    // return whether the slices form a possible eye
    // ------------------------------------------------------------
    bool CEyeDetector::CalcCentre(const int iXstart, const int iYstart,
                                  const int iSize, bool fX,
                                  SBasicEyeData& sBasicEyeData);

    // ------------------------------------------------
    // given a start position in x,y, get the associated
    // x slice and then get PI & IS boundaries for
    // both directions (left and right)
    // ------------------------------------------------
    bool GetEyeFromXslice(const int iYpos, const int iXcentre,
                          const int iSize, SEyeBoundaries& sEyeBoundaries);

    // ------------------------------------------------
    // given a start position in x,y, get the associated
    // y slice and then get PI & IS boundaries for
    // both directions (top and bottom)
    // ------------------------------------------------
    bool GetEyeFromYslice(const int iXpos, const int iYcentre,
```

EyeDetector.h changes from Appendix 6

```
                        const int iSize, SEyeBoundaries& sEyeBoundaries);

// -------------------------------------------------------
// given a slice of data,split it in to left(or top) and
// right (or bottom) and feed it to the CheckSlice routine.
// -------------------------------------------------------
bool GetEyeFromSliceData(const int* pai, const int iSize,
                         SEyeBoundaries& sEyeBoundaries);

// -------------------------------------------------
// given a data set check whether it is a possible eye
// return false if not, otherwise true and sBoundaries
// will contain where the locations are.  NOTE:  the
// input data will be altered.
// -------------------------------------------------
bool CheckSlice(int* paiData, const int iSizeData,
                SBoundaries& sBoundaries);

// -------------------------------------------------
// from given x, y and size (in SEyeData) gather slices
// in x & y directions.  Check if valid slices and come
// up with best fit for centres.
// -------------------------------------------------
bool Validate(SEyeData& sEyeData);

// -------------------------------------------------
// from given x, y and size gather slices in x direction.
// Check if valid slices and come up with best fit for
// x centre from incoming x.
// -------------------------------------------------
bool ValidateX(const int x, const int y, const int iSize,
               SBasicEyeData& sBasicEyeData);

// -------------------------------------------------
// from given x, y and size gather slices in y direction.
// Check if valid slices and come up with best fit for
// y centre from incoming y.
// -------------------------------------------------
bool ValidateY(const int x, const int y, const int iSize,
               SBasicEyeData& sBasicEyeData);

// -------------------------------------------------
// parse and validate found slice, calc offset and return
// some level of confidence in the data
// -------------------------------------------------
void ValidateBoundaries(SBasicEyeData& sBasicEyeData);

// ###################################################################
// variables
// ###################################################################

// -------------------------------------------------
// constants used in program - actually set as variables
// so at a later date we will allow the user/program
// to be able to dynamically alter them
// -------------------------------------------------

// -------------------------------------------------
// number of negative differences allowed whilst looking
// for pupil/iris boundary
// -------------------------------------------------
int ciPI_NEG_ALLOWED;

// -------------------------------------------------
// number of positive differences allowed whilst looking
// for iris/sclera boundary
// -------------------------------------------------
int ciIS_POS_ALLOWED;

// -------------------------------------------------
// the percentage differences (remember x2) for threshold
```

EyeDetector.h changes from Appendix 6

```
        // of pupil/iris
        // -------------------------------------------------------
        float cfPI_PC_DIFF;

        // -------------------------------------------------------
        // the percentage differences (remember x2) for threshold
        // of iris/sclera
        // -------------------------------------------------------
        float cfIS_PC_DIFF;

        // ------------------------------
        // the level of the reflectivity
        // ------------------------------
        int ciHOT_SPOT_VALUE;

        // -------------------------------------------------
        // when starting search ignore if difference is above
        // this value (hot spot effects)
        // -------------------------------------------------
        int ciDIFF_IGNORE;

        // ----------------------------
        // pointer to where raw data is
        // ----------------------------
        image<unsigned char>* pImg;
};
```

Test.cc

```
/*
  As saving structures for ease of getting old data back we have
  the potential for segmentation errors if we edit the .h files.
  To prevent this the data has been moved into here for safe keeping.
*/

// ------------------------------------------------------------------------
// included files
// ------------------------------------------------------------------------
#include "Test.h"
#include "EyeDetector.h"

// ------------------------------------------------------------------------
// constructor - clears data and sets sample size to +/- 50
// ------------------------------------------------------------------------
SEyeTestData::SEyeTestData()
{
  Clear(50);
}

// ------------------------------------------------------------------------
// invalidate the use eye locations
// ------------------------------------------------------------------------
void SEyeTestData::Clear(int iSizeBoxIn)
{
  for(int i=0; i<=eNumIntPosns; i++)
    aiXposns[i] = aiYposns[i] = -1;
  iSizeBox = iSizeBoxIn;
  iXcalc = iYcalc = -1;
}

// ------------------------------------------------------------------------
// if got some confidence in eye then set value and return true, else false
// ------------------------------------------------------------------------
bool SEyeTestData::GetCalcX(int& iVal)
{
  if(sEyeData.sEyeX.eConfidence == NONE)
    return false;

  iVal = iXcalc;
  return true;
}

// ------------------------------------------------------------------------
// if got some confidence in eye then set value and return true, else false
// ------------------------------------------------------------------------
bool SEyeTestData::GetCalcY(int& iVal)
{
  if(sEyeData.sEyeY.eConfidence == NONE)
    return false;

  iVal = iYcalc;
  return true;
}

// ------------------------------------------------------------------------
// set new centre values - old data will be lost (except sample size)
// ------------------------------------------------------------------------
void SEyeTestData::NewValue(int xIn, int yIn)
{
  Clear(iSizeBox);
  aiXposns[2]=xIn;
  aiYposns[2]=yIn;
}

// ------------------------------------------------------------------------
// routine that calls the eye detector to get the slices and validate them
// ------------------------------------------------------------------------
bool SEyeTestData::CalcCentres(CEyeDetector* pEyeDetector)
{
```

Test.cc

```
  // initially we have no confidence in calc value
  sEyeData.sEyeX.eConfidence = sEyeData.sEyeY.eConfidence = NONE;

  // the centre points must be defined
  if((aiXposns[2] == -1) || (aiYposns[2] == -1))
    return false;

  // set start values
  sEyeData.iXstart =  GetXcentre();
  sEyeData.iYstart =  GetYcentre();
  sEyeData.iSize =  iSizeBox;

  // now validate it and set the calculated centres
  if(pEyeDetector->CalcCentres(sEyeData.iXstart, sEyeData.iYstart,
                               iSizeBox, sEyeData)){
    iXcalc = static_cast<int>(GetXcentre() + sEyeData.sEyeX.fAdj + 0.5f);
    iYcalc = static_cast<int>(GetYcentre() + sEyeData.sEyeY.fAdj + 0.5f);
    return true;
  }

  // inform user we do not have any confidence in slice data
  return false;
}

// ---------------------------------------------------------------------
// clear out both eyes data
// ---------------------------------------------------------------------
void STestData::Clear()
{
  testEye1.Clear(testEye1.iSizeBox);
  testEye2.Clear(testEye1.iSizeBox);
}
```

Test.h

```
#ifndef Test_CW_h
#define Test_CW_h

#include "EyeDetector.h"
#include "Point.h"

// ------------------------------------------------------------------------
// We need to hold the user points in a structure for ease
// ------------------------------------------------------------------------
struct SEyeTestData
{
  // ######################################################################
  // functions
  // ######################################################################

  // -----------
  // constructor
  // -----------
  SEyeTestData();

  // ---------------------------------------------------------
  // actual routine that calls eye detector with x,y location
  // and sets the centre values calculated
  // ---------------------------------------------------------
  bool CalcCentres(CEyeDetector* pEyeDetector);

  // -------------------
  // clear out user data
  // -------------------
  void Clear(int iSizeBoxIn);

  // ---------------------------------------------------------
  // if x centre found, set iVal and return true, else false
  // ---------------------------------------------------------
  bool GetCalcX(int& iVal);

  // ---------------------------------------------------------
  // if y centre found, set iVal and return true, else false
  // ---------------------------------------------------------
  bool GetCalcY(int& iVal);

  // ------------------------------------
  // recall the user x centre (-1 if none)
  // ------------------------------------
  int GetXcentre()const {return aiXposns[2];}

  // ------------------------------------
  // recall the user y centre (-1 if none)
  // ------------------------------------
  int GetYcentre()const {return aiYposns[2];}

  // ---------------------
  // set new centre values
  // ---------------------
  void NewValue(int xIn, int yIn);

  // ######################################################################
  // constants
  // ######################################################################
  enum {eNumIntPosns=5}; // pos 0 = SI, 1=IP, 2 = centre, 3 = PI, 4 = IP

  // ######################################################################
  // variables
  // ######################################################################
  int aiXposns[eNumIntPosns], aiYposns[eNumIntPosns];
  int iSizeBox; // how large box (+/-) around eye is for grabbing samples
  int iXcalc, iYcalc;     // calculated eye centre (-1 = invalid)
  SEyeData sEyeData;    // used in calculations
};
```

Test.h

```
struct STestData
{
   // ###################################################################
   // functions
   // ###################################################################

   // -----------
   // constructor
   // -----------
   STestData() : fEye1Active(true) {}

   // ------------------------
   // clear both user eye data
   // ------------------------
   void Clear();

   // ###################################################################
   // variables
   // ###################################################################
   SEyeTestData testEye1, testEye2, testPos;  // two eyes & test position
   bool fEye1Active;                     // which one we are working on
   int iSavedXpos, iSavedYpos;         // last known cursor position
};

#endif
```

specific.cc changes from Appendix 7

- New Description

```
/*
 To increase accuracy near the hot spot we now calculate for
 all values from given value to +/-5 points from it.  Using
 the highest level (which should be L4) we then add the offsets
 and divide by the number of replies at highest level to find the
 new offset.  The number of replies at the highest level is also
 saved as we now deem eye to be valid if the confidence level
 is L4 and there are at least 3 replies at that level (for both
 x and y directions)
*/
```

- Altered ImageLoaded

```
void ImageLoaded(image<unsigned char>& imgMod, const char* directory,
                 int iFileNumber)
{
  // get screen limits
  iXposMax = imgMod.getWidth();
  iYposMax = imgMod.getHeight();

  // set deafult positions
  iXpos = iXposMax/2;
  iYpos = iYposMax/2;

  // clear old test data
  testData.Clear();

  // read in any associated test data (if any)
  ifstream input;
  char ac[800];
  sprintf(ac, "%s/ProgData/Image%04d.prgData", directory, iFileNumber);
  input.open(ac, ios::binary);
  if(input){
    // read in data
    input.read((char*)&testData, sizeof(testData));
    input.close();

    // set last x,y position as current
    iXpos = testData.iSavedXpos;
    iYpos = testData.iSavedYpos;
}
  // update the calculated data for the eyes
  testData.testEye1.CalcCentres(pEyeDetector);
  testData.testEye2.CalcCentres(pEyeDetector);
  testData.testPos.NewValue(iXpos, iYpos);
  testData.testPos.CalcCentres(pEyeDetector);

  // actually draws our data
  DoUpdate(-1, imgMod);
}
```

- Altered ShowUserVsProg

```
void ShowUserVsProg(int iEye)
{
  SEyeTestData* pEye=NULL;

  // get correct eye pointer
  if(iEye == 3){
    // cursor test eye
    pEye = &testData.testPos;
    int x = pEye->GetXcentre();
    int y = pEye->GetYcentre();
    if((x != -1) && (y !=-1))
      printw("Posn:=> x = %d, y = %d\n", x,y);
    else
      return;
  }
  else{
```

specific.cc changes from Appendix 7

```
    // user defined eye 1 or eye 2
    pEye = (iEye == 1) ? &testData.testEye1 : &testData.testEye2;
    int x =pEye->GetXcentre();
    int y= pEye->GetYcentre();
    if((x != -1) && (y !=-1))
      printw("Eye%d:=>  User x = %d, User y = %d\n", iEye, x, y);
    else
      return;
  }

  // show the calculated x centre
  printw("         Prog x = ");
  float fVal;
  if(pEye->sEyeData.sEyeX.GetCentre(fVal))
    printw("%.1f conf(%d) size(%d)",fVal,  pEye->sEyeData.sEyeX.eConfidence,
           pEye->sEyeData.sEyeX.iNumAveraged);
  else
    printw("INVALID");

  // show the calculated y centre
  printw("\n         Prog y = ");
  if(pEye->sEyeData.sEyeY.GetCentre(fVal))
    printw("%.1f conf(%d) size(%d)", fVal,  pEye->sEyeData.sEyeY.eConfidence,
           pEye->sEyeData.sEyeX.iNumAveraged);
  else
    printw("INVALID");

  // state whether this is to be treated as an eye
  if(pEye->sEyeData.IsItAnEye()) // have run eye detector so ok to call
    printw(" [AN EYE!!]\n");
  else
    printw (" [Not an eye]\n");
}
```

EyeDetector.cc changes from Appendix 7

- Tightened confidence factor in CalcCentres

```
bool CEyeDetector::CalcCentres(const int iXstart, const int iYstart,
                               const int iSize, SEyeData& sEyeData)
{
  // calculate for x slices
  CalcCentre(iXstart, iYstart, iSize, true, sEyeData.sEyeX);

  // and calculate for y slices
  CalcCentre(iXstart, iYstart, iSize, false, sEyeData.sEyeY);

  // inform caller whether we think it is an eye
  return ((sEyeData.sEyeX.eConfidence == L4) &&
       (sEyeData.sEyeX.iNumAveraged > 2) &&
       (sEyeData.sEyeY.eConfidence == L4) &&
       (sEyeData.sEyeY.iNumAveraged > 2));
}
```

- CalcCentre now goes +/-5 pixels extra calculations

```
bool CEyeDetector::CalcCentre(const int iXstart, const int iYstart,
                              const int iSize, bool fX,
                              SBasicEyeData& sBasicEyeData)
{
  // need arrays to hold confidence adjustments and count of how many in
  float afAdjust[L4+1];
  int aiNumIn[L4+1];
  for(int i=0; i<=L4; i++){
    aiNumIn[i] = 0;
    afAdjust[i] = 0.0f;
  }

  // calc how we need to move
  int iXstep, iYstep;
  if(fX){
    iXstep = 1;
    iYstep = 0;
  }
  else{
    iXstep = 0;
    iYstep = 1;
  }

  // now get results as we vary is varied
  for(int i=-5; i<=5; i++){
    // check slice data
    bool fGood;
    if(fX)
      fGood = ValidateX(iXstart+i, iYstart, iSize, sBasicEyeData);
    else
      fGood = ValidateY(iXstart, iYstart+i, iSize, sBasicEyeData);

    // if good we can add into the array
    if(fGood){
      int iConf = sBasicEyeData.eConfidence;
      int iNumValid = sBasicEyeData.dq.size();
      afAdjust[iConf] += (sBasicEyeData.fAdj + i) * iNumValid;
      aiNumIn[iConf] += iNumValid;;
    }
  }

  // we need to find the highest level we have
  int iLevel;
  for(iLevel = L4; iLevel>NONE; iLevel--){
    if(aiNumIn[iLevel] > 0)
      break;
  }

  if(iLevel > NONE){
```

EyeDetector.cc changes from Appendix 7

```
      // set new confidence level
      sBasicEyeData.eConfidence = static_cast<EConfidence>(iLevel);

      // and the true start
      sBasicEyeData.iStart = (fX) ? iXstart : iYstart;

      // we now need to average the values found at this level
      sBasicEyeData.fAdj = afAdjust[iLevel]/aiNumIn[iLevel];

      // save how many values at that level
      sBasicEyeData.iNumAveraged = aiNumIn[iLevel];

      return true;
    }

    // not valid at all
    sBasicEyeData.eConfidence = NONE;
    sBasicEyeData.iNumAveraged = 0;
    return false;
}
```

EyeDetector.h changes from Appendix 7

- Added number averaged variable to SBasicEyeData

```
int iNumAveraged;  // how many averaged at that confidence level
```

- SEyeData IsItAnEye routine changed to use new variable

```
bool IsItAnEye() // ensure used pEyeDetector to set it 1st!
{
  if((sEyeX.eConfidence == L4) && (sEyeY.eConfidence == L4) &&
       (sEyeX.iNumAveraged > 2) && (sEyeY.iNumAveraged > 2))
    return true;

  return false;
}
```

specific.cc changes from Appendix 8

- New Description

```
/*
 Hotspot positions are detected from the image (only 1/4-3/4 of screen y
 as they will be contained within it).  These are then checked to see
 if they are an eye and if so will be added to a vector of eyes
 (we expect only two but who knows!).  Once detected the positions
 are show on the image and information pane.
*/
```

- Near end of DoUpdate added ShowFoundEyePositions call

```
// and the found eye positions
ShowFoundEyePositions(imgMod);
```

- And the ShowFoundEyePositions routine

```
// ---------------------------------------------------------------------------
// cycle the data that the FindEyes algorithm detected and show positions
// as large cursor on image
// ---------------------------------------------------------------------------
void ShowFoundEyePositions(image<unsigned char>& imgMod)
{
  vector<SFloatPoint> vPts;
  int iVal = 1;

  // get found positions
  if(pEyeDetector->GetFoundEyes(vPts)){
    // need to draw them all
    vector<SFloatPoint>::iterator it;
    for(it = vPts.begin(); it != vPts.end(); it++, iVal++){
      // manually convert
      int x = static_cast<int>(it->x + 0.5f);
      int y = static_cast<int>(it->y + 0.5f);

      // draw a larger cursir
      for(int i=-30; i<=30; i++){
        imgMod.setPixel(x+i, y, 255);
        imgMod.setPixel(x, y+i, 255);
      }
    }
  }
}
```

- ImageLoaded altered to call auto detect

```
void ImageLoaded(image<unsigned char>& imgMod, const char* directory,
                 int iFileNumber)
{
  // get screen limits
  iXposMax = imgMod.getWidth();
  iYposMax = imgMod.getHeight();

  // set deafult positions
  iXpos = iXposMax/2;
  iYpos = iYposMax/2;

  // clear old test data
  testData.Clear();

  // read in any associated test data (if any)
  ifstream input;
  char ac[800];
  sprintf(ac, "%s/ProgData/Image%04d.prgData", directory, iFileNumber);
  input.open(ac, ios::binary);
  if(input){
    // read in data
    input.read((char*)&testData, sizeof(testData));
    input.close();
```

```
      // set last x,y position as current
      iXpos = testData.iSavedXpos;
      iYpos = testData.iSavedYpos;
    }

    // update the calculated data for the eyes
    testData.testEye1.CalcCentres(pEyeDetector);
    testData.testEye2.CalcCentres(pEyeDetector);
    testData.testPos.NewValue(iXpos, iYpos);
    testData.testPos.CalcCentres(pEyeDetector);

    // find eyes on image
    pEyeDetector->FindEyes(50);  // search +/-50 around points

    // actually draws our data
    DoUpdate(-1, imgMod);
  }
```

- ShowInfo altered to show auto detected information as well

```
void ShowInfo(const char* file)
{
  clear();
  printw("%s\n", file);
  printw("Current Eye Actions on Eye (%d)\n", (testData.fEye1Active)? 1:2);
  printw("X=%i, y=%i, value=%d\n", iXpos, iYpos, iPixelValue);
  ShowAutoDetectedInfo();
  ShowUserVsProg(1);
  ShowUserVsProg(2);
  ShowUserVsProg(3);    // user pressing c as location
  printw("Info: [%s]\n", strMessage.c_str());
  refresh();
}
```

- and the routine that shows the data

```
// ---------------------------------------------------------------------------
// cycle the data that the FindEyes algorithm detected and show info
// ---------------------------------------------------------------------------
void ShowAutoDetectedInfo()
{
  vector<SFloatPoint> vPts;
  int iVal = 1;

  // get found positions
  if(pEyeDetector->GetFoundEyes(vPts)){
    // we have eyes!
    printw("Eyes have been found at:-\n");

    // and show what we have found
    vector<SFloatPoint>::iterator it;
    for(it = vPts.begin(); it != vPts.end(); it++, iVal++){
      printw("    [%d] x = %.1f and y = %.1f\n", iVal, it->x, it->y);
    }
  }
  else
    printw("No eyes automatically found\n");

}
```

- added routine to show positions on screen

```
// ---------------------------------------------------------------------------
// cycle the data that the FindEyes algorithm detected and show positions
// as large cursor on image
// ---------------------------------------------------------------------------
void ShowFoundEyePositions(image<unsigned char>& imgMod)
{
  vector<SFloatPoint> vPts;
  int iVal = 1;
```

specific.cc changes from Appendix 8

```
  // get found positions
  if(pEyeDetector->GetFoundEyes(vPts)){
    // need to draw them all
    vector<SFloatPoint>::iterator it;
    for(it = vPts.begin(); it != vPts.end(); it++, iVal++){
      // manually convert
      int x = static_cast<int>(it->x + 0.5f);
      int y = static_cast<int>(it->y + 0.5f);

      // draw a larger cursir
      for(int i=-30; i<=30; i++){
           imgMod.setPixel(x+i, y, 255);
           imgMod.setPixel(x, y+i, 255);
        }
      }
  }
}
```

EyeDetector.cc changes from Appendix 8

- Seeded new variables for screen start and end for getting retinal reflections

```
CEyeDetector::CEyeDetector(image<unsigned char>* pImgIn)
              : pImg(pImgIn)
{
  // simply set up the deafult values
  ciPI_NEG_ALLOWED = 6;
  ciIS_POS_ALLOWED = 5;
  cfPI_PC_DIFF = 2 * 0.15f;
  cfIS_PC_DIFF = 2 * 0.15f;
  ciHOT_SPOT_VALUE = 170;
  ciDIFF_IGNORE = 75;
  cfY_START_PC= 0.25f;
  cfY_END_PC= 0.75f;
}
```

- Added new public routine to find eyes

```
// ---------------------------------------------------------------------------
// This will find any eyes in the (correctly framed) image.  iSearchSize
// is the number of samples either side of the hot spot to use when
// getting the x & y slices.
// ---------------------------------------------------------------------------
bool CEyeDetector::FindEyes(const int iSearchSize)
{
  // get start and end y positions
  int iYstart = pImg->getHeight();
  int iYend = static_cast<int>(cfY_END_PC * iYstart);
  iYstart = static_cast<int>(cfY_START_PC * iYstart);

  // we need eye data to work on
  SEyeData sEyeData;

  // we need to ensure that we don't keep searching on same hotspot
  vHotSpots.clear();

  // ensure last time values are saved (as may want to use them)
  vEyesLastTime.clear();
  vEyesLastTime = vEyes;
  vEyes.clear();

  // now can search the image
  for(int y=iYstart; y<=iYend; y++){
    // traverse the x points
    for(int x=iSearchSize; x<pImg->getWidth()-iSearchSize-1; x++){
      // is it a hot spot?
      if(pImg->getPixel(x,y) >= ciHOT_SPOT_VALUE){
        // check if already found an eye at this position
        // (x altered past it if so)
        if(!IsInEye(x, y)){
          // is it a hot spot we have already checked?
          // (again x altered past it if so){
          if(!IsAHotSpot(x, y)){
            // add in new hot spot
            AddHotSpot(x, y);
            // is it an eye?
            if(CalcCentres(x, y, iSearchSize, sEyeData)){
              // add in to eye array
              vEyes.push_back(sEyeData);
            }
          }
        }
      }
    }
  }

  // return true if any eyes found
  return (vEyes.size() > 0) ? true : false;
}
```

EyeDetector.cc changes from Appendix 8

- Added new routine to return the found eye positions

```
// ---------------------------------------------------------
// return the eye points that we have found as float values
// ---------------------------------------------------------
bool CEyeDetector::GetFoundEyes(std::vector<SFloatPoint>& vPts)
{
  vector<SEyeData>::iterator it;
  SFloatPoint fp;
  float f=0.0f;

  if(vEyes.size() == 0)
    return false;

  // simply set up point and add into vector
  for(it = vEyes.begin(); it != vEyes.end(); it++){
    it->sEyeX.GetCentre(f);
    fp.x = f;
    it->sEyeY.GetCentre(f);
    fp.y = f;
    vPts.push_back(fp);
  }

  return true;
}
```

- Added routine to add hot spot to list

```
// ---------------------------------------------------------------------------
// from the given start positions,calculate hot spot dimensions and add
// in to vector
// ---------------------------------------------------------------------------
void CEyeDetector::AddHotSpot(const int xIn, const int yIn)
{
  // y is first line we have met it on but it could start
  // earlier on next line so we must expand out both ways
  int iContinueCount=2;
  int y = yIn;

  // for finding hot spot rectangle
  SRect sRect;

  // calculate limits of this hot spot
  sRect.left = sRect.right = xIn;
  sRect.top = sRect.bottom = yIn;
  do{
    bool fContinue = false; // assume we will exit

    // now see if we have a new lower limit
    for(int x=xIn-1; x>2; x--){
      if(pImg->getPixel(x, y) < ciHOT_SPOT_VALUE)
        break;
      if(x < sRect.left)
        sRect.left = x;
      fContinue = true;
    }

    // now see if we have a new upper limit
    for(int x=xIn; x<pImg->getWidth(); x++){
      if(pImg->getPixel(x, y) < ciHOT_SPOT_VALUE)
        break;
      if(x > sRect.right)
        sRect.right = x;
      fContinue = true;
    }
    y++; //next line

    if(!fContinue)
      --iContinueCount;
  }while(iContinueCount);
```

```
    // save end of rectangle (y)
    sRect.bottom = y-1;

    // add in new rectangle
    vHotSpots.push_back(sRect);
  }
```

- Added routine to check if we already actioned this hot spot

```
// ---------------------------------------------------------------------------
// check whether this is an already actioned hot spot
// ---------------------------------------------------------------------------
bool CEyeDetector::IsAHotSpot(int&x, const int& y)
{
  std::vector<SRect>::iterator it;

  // cycle all hot spots we have found so far
  for(it=vHotSpots.begin(); it!=vHotSpots.end(); it++){
    // check if in rectangle
    if(it->Contains(x, y)){
      x += 10;  // skip well past it
      return true;
    }
  }

  return false;
}
```

- Added routine to check if the point is in an already found eye

```
// ---------------------------------------------------------------------------
// checks whether this location is within the detected eye and if so moves
 // past it (actually skips the next 40 pixels in the x direction for speed,
// can not skip y as head can be slanted).
// ---------------------------------------------------------------------------
bool CEyeDetector::IsInEye(int& x, const int&y)
{
 std::vector<SEyeData>::iterator it;

 // cycle all eyes we have found so far
 for(it=vEyes.begin(); it!=vEyes.end(); it++){
   // check if in eye (within +/- 10 pixels of it
   if(it->IsPointInEye(x,y)){
     x += 20;  // skip well past it
     return true;
   }
 }

 return false;
}
```

- **Also changed EyeDetector.h to add the new routines and variables**

Point.h

```
#ifndef Point_CW_h
#define Point_CW_h

// ---------------------------------------------------------------------
// structure for working on a point - not made private as we do not
// need to do so here.  Added Floating point version.
// ---------------------------------------------------------------------
struct SPoint
{
   // ###################################################################
   // functions
   // ###################################################################

   // -----------
   // constructor
   // -----------
   SPoint(int xIn=-1, int yIn=-1) : x(xIn), y(yIn) {}

   // ###################################################################
   // variables
   // ###################################################################
   int x, y;
};

// ---------------------------------------------------------------------
// structure for working on floating point - not made private as we do not
// need to do so here.
// ---------------------------------------------------------------------
struct SFloatPoint
{
   // ###################################################################
   // functions
   // ###################################################################
   SFloatPoint(float xIn=-1.0f, float yIn=-1.0f) : x(xIn), y(yIn) {}

   // ###################################################################
   // variables
   // ###################################################################
   float x, y;
};

#endif
```

Rect.h

```
#ifndef Rect_CW_h
#define Rect_CW_h

// ---------------------------------------------------------------------
// structure for working on a rectangle - not made private as we do not
// need to do so here.
// ---------------------------------------------------------------------
struct SRect
{
  // #####################################################################
  // functions
  // #####################################################################
  bool Contains(const int x, const int y)
  {
    if((x>=left) && (x<=right) && (y>=top) && (y<=bottom))
      return true;

    return false;
  }

  // #####################################################################
  // variables
  // #####################################################################
  int top, left, bottom, right;
};

#endif
```

specific.cc changes from Appendix 9

- New description

```
/*
   Glass reflections were being picked up as eyes.  This was due to low start
   value meaning thresholds were also low.  Now any start pixel value less than
   35 is excluded.

   The number of negatives allowed whilst looking for the pupil/iris boundary
   been reduced from the 6 to 3.

   When skipping past an eye (as already actioned) we only skipped the x axis by
   20 pixels rather than the size specified by the user.  This created a
   possibility of two eyes being detected at the same place, they are now in sync
   to prevent this.

   As we now take samples from +/-5 from centre point we expect the minimum
   sample count to be much higher.  I have now set this as being at least
   10 to be valid as an eye.

   Also lowered hot spot value to 130 to allow side images a better chance
*/
```

EyeDetector.cc changes from Appendix 9

- New parameters set in constructor

```
CEyeDetector::CEyeDetector(image<unsigned char>* pImgIn)
              : pImg(pImgIn)
{
  // simply set up the deafult values
  ciPI_NEG_ALLOWED = 3;
  ciIS_POS_ALLOWED = 5;
  cfPI_PC_DIFF = 2 * 0.15f;
  cfIS_PC_DIFF = 2 * 0.15f;
  ciHOT_SPOT_VALUE = 170;
  ciDIFF_IGNORE = 75;
  cfY_START_PC= 0.25f;
  cfY_END_PC= 0.75f;
  ciMIN_START_PIXEL_VALUE = 35;
}
```

- CalcCentres changed to call check for eye instead of inline

```
bool CEyeDetector::CalcCentres(const int iXstart, const int iYstart,
                               const int iSize, SEyeData& sEyeData)
{
  // calculate for x slices
  CalcCentre(iXstart, iYstart, iSize, true, sEyeData.sEyeX);

  // and calculate for y slices
  CalcCentre(iXstart, iYstart, iSize, false, sEyeData.sEyeY);

  // inform caller whether we think it is an eye
  return sEyeData.IsItAnEye();
}
```

- Changed FindEyes to ensure that iSearchSize is passed to other rotuines

```
bool CEyeDetector::FindEyes(const int iSearchSize)
{
  // get start and end y positions
  int iYstart = pImg->getHeight();
  int iYend = static_cast<int>(cfY_END_PC * iYstart);
  iYstart = static_cast<int>(cfY_START_PC * iYstart);

  // we need eye data to work on
  SEyeData sEyeData;

  // we need to ensure that we don't keep searching on same hotspot
  vHotSpots.clear();

  // ensure last time values are saved (as may want to use them)
  vEyesLastTime.clear();
  vEyesLastTime = vEyes;
  vEyes.clear();

  // now can search the image
  for(int y=iYstart; y<=iYend; y++){
    // traverse the x points
    for(int x=iSearchSize; x<pImg->getWidth()-iSearchSize-1; x++){
      // is it a hot spot?
      if(pImg->getPixel(x,y) >= ciHOT_SPOT_VALUE){
        // check if already found an eye at this position
        // (x altered past it if so)
        if(!IsInEye(x, y, iSearchSize)){
          // is it a hot spot we have already checked?
          // (again x altered past it if so){
          if(!IsAHotSpot(x, y)){
            // add in new hot spot
            AddHotSpot(x, y);
            // is it an eye?
            if(CalcCentres(x, y, iSearchSize, sEyeData)){
              // add in to eye array
              vEyes.push_back(sEyeData);
```

EyeDetector.cc changes from Appendix 9

```
                }
              }
            }
          }
        }
      }

      // return true if any eyes found
      return (vEyes.size() > 0) ? true : false;
    }
```

- Changed IsInEye to use search size

```
bool CEyeDetector::IsInEye(int& x, const int&y, int iSearchSize)
{
  std::vector<SEyeData>::iterator it;

  // cycle all eyes we have found so far
  for(it=vEyes.begin(); it!=vEyes.end(); it++){
    // check if in eye (within +/- 10 pixels of it
    if(it->IsPointInEye(x,y, iSearchSize)){
      x += iSearchSize;  // skip past it
      return true;
    }
  }

  return false;
}
```

- The following check was added to CheckSlice to ensure we don't start with a pixel value that can not be in the pupil

```
// ensure that we start from a high enough pupil value
if(paiData[0] < ciMIN_START_PIXEL_VALUE)
  return false;
```

EyeDetector.h changes from Appendix 9

- Changed IsItAnEye to increase number of slices before deemed an eye

```
bool IsItAnEye() // ensure used pEyeDetector to set it 1st!
{
  // -------------------------------------------------------------
  // note increase in iNumAveraged as we now take readings from
  // -5 to +5 around start point and so we expect a much larger
  // data set to be a valid eye.
  // -------------------------------------------------------------
  if((sEyeX.eConfidence == L4) && (sEyeY.eConfidence == L4) &&
       (sEyeX.iNumAveraged > 9) && (sEyeY.iNumAveraged > 9))
    return true;

  return false;
}
```

- Changed definition of parameter in IsInEye

```
bool IsInEye(int& x, const int& y, int iSearchSize);
```

- Added new variables

```
int ciMIN_START_PIXEL_VALUE;
int ciMIN_START_PIXEL_VALUE;
```

specifi.cc changes from Appendix 10

- New description

```
/*
// I now do the check at hotspot, hotspot.topLeft(x-3,y-3), bottom
// left (x-3, y+3), top right(x+3, y-3) and bottom right(x+3, y-3)
// (i.e. +/-3 diagonal off the hotspot).  The resulting x,y adjustments
// (for those at highest level) are then multiplied by the number of their
// valid slices and all added together and divided by total to give our
// new best guess!  NOTE:  The new combined confidence number of samples
// must be >= 20 to be valid.
*/
```

EyeDetector.cc changes from Appendix 10

- FindEyes altered to call SuperCalc

```
bool CEyeDetector::FindEyes(const int iSearchSize)
{
  SRect sHotSpot;
  // get start and end y positions
  int iYstart = pImg->getHeight();
  int iYend = static_cast<int>(cfY_END_PC * iYstart);
  iYstart = static_cast<int>(cfY_START_PC * iYstart);

  // we need eye data to work on
  SEyeData sEyeData;

  // we need to ensure that we don't keep searching on same hotspot
  vHotSpots.clear();

  // ensure last time values are saved (as may want to use them)
  vEyesLastTime.clear();
  vEyesLastTime = vEyes;
  vEyes.clear();

  // now can search the image
  for(int y=iYstart; y<=iYend; y++){
    // traverse the x points
    for(int x=iSearchSize; x<pImg->getWidth()-iSearchSize-1; x++){
      // is it a hot spot?
      if(pImg->getPixel(x,y) >= ciHOT_SPOT_VALUE){
        // check if already found an eye at this position
        // (x altered past it if so)
        if(!IsInEye(x, y, iSearchSize)){
          // is it a hot spot we have already checked?
          // (again x altered past it if so){
          if(!IsAHotSpot(x, y)){
            // add in new hot spot
            sHotSpot  = AddHotSpot(x, y);
            // is it an eye?
            //if(CalcCentres(x, y, iSearchSize, sEyeData)){
            if(SuperCalcCentres(sHotSpot, iSearchSize, sEyeData)){
              // add in to eye array
              vEyes.push_back(sEyeData);
            }
          }
        }
      }
    }
  }

  // return true if any eyes found
  return (vEyes.size() > 0) ? true : false;
}
```

- AddHotSpot now returns the hotspot dimensions back to the caller

```
SRect CEyeDetector::AddHotSpot(const int xIn, const int yIn);
```

- Added SuperCalc routine

```
// ----------------------------------------------------------------------
// from given position and diagonals get the centres and average to
// get our best guess
// ----------------------------------------------------------------------
bool CEyeDetector::SuperCalcCentres(const SRect sRect, const int iSearchSize,
                                    SEyeData& sEyeData)
{
  // do all possible tests
  CalcCentres(sRect.left, sRect.top, iSearchSize, asEyeData[0]);
  CalcCentres(sRect.left-3, sRect.top-3, iSearchSize, asEyeData[1]);
  CalcCentres(sRect.left-3, sRect.bottom+3, iSearchSize, asEyeData[2]);
  CalcCentres(sRect.right+3, sRect.bottom+3, iSearchSize, asEyeData[3]);
  CalcCentres(sRect.right+3, sRect.top+3, iSearchSize, asEyeData[4]);
```

EyeDetector.cc changes from Appendix 10

```
  // now we need to average out the data - x 1st
  float fVal=0;
  float fSumCentre = 0;
  int    iNumAveraged = 0;
  for(int i=0; i<5; i++){
    if(asEyeData[i].IsItAnEye()){
      asEyeData[i].sEyeX.GetCentre(fVal);
      fSumCentre += (asEyeData[i].sEyeX.iNumAveraged * fVal);
      iNumAveraged += asEyeData[i].sEyeX.iNumAveraged;
    }
  }

  // do we have good eye data?
  if(iNumAveraged < 20){
    // no, simply set to hot spot
    sEyeData = asEyeData[0];
    return false;
  }

  // set up the new x eye data
  fVal = fSumCentre/iNumAveraged;
  sEyeData.sEyeX.iStart = static_cast<int>(fVal + 0.5f);
  sEyeData.sEyeX.fAdj = fVal - sEyeData.sEyeX.iStart;
  sEyeData.sEyeX.iNumAveraged = iNumAveraged;
  sEyeData.sEyeX.eConfidence = L4;

    // now we need to average out the y data
  fVal=0;
  fSumCentre = 0;
  iNumAveraged = 0;
  for(int i=0; i<5; i++){
    if(asEyeData[i].IsItAnEye()){
      asEyeData[i].sEyeY.GetCentre(fVal);
      fSumCentre += (asEyeData[i].sEyeY.iNumAveraged * fVal);
      iNumAveraged += asEyeData[i].sEyeY.iNumAveraged;
    }
  }

  // do we have good eye data?
  if(iNumAveraged < 20){
    // no, simply set to hot spot
    sEyeData = asEyeData[0];
    return false;
  }

  // set up the new x eye data
  fVal = fSumCentre/iNumAveraged;
  sEyeData.sEyeY.iStart = static_cast<int>(fVal + 0.5f);
  sEyeData.sEyeY.fAdj = fVal - sEyeData.sEyeY.iStart;
  sEyeData.sEyeY.iNumAveraged = iNumAveraged;
  sEyeData.sEyeY.eConfidence = L4;

  return true;
}
```

- **Added new definition of AddHotSpot and new SuperCalc definition to specific.h**

Specific.cc changes from Appendix 11

- New description

```
/*
 We expect the first eye to be on the left hand side and eye 2 on the
 right hand side.  The GetFoundEyes returns all *eyes* that it has
 found regardless of how many and this needs to be preserved so that
 users may get the found data and validate it if the project is taken
 further.

 A new routine, GetValidatedFoundEyes, can be called and it will fill
 incoming SPoints variables with the points for left eye and right eye
 where left eye is one that has x less than 1/2 screen width and right
 one greater.  If the eye is not found then the points will contain
 a large negative number (defined as cfNO_EYE and is a float).

 If there are more than one eye on a given side then that eye is deemed
 as not valid.

 In all cases the routine will return the number of eyes that FindEyes
 detected.

 The data on user info panel shows all eyes whereas the image shows only
 the "validated" eyes.

 The modified image is written to disk when closed.
*/
```

- Changed ImageClosing to allow image to be saved as a jpg

```
void ImageClosing(image<unsigned char>& imgRaw, const char* directory,
                  int iImageNumber)
{
  // save current x,y cursor position
  testData.iSavedXpos = iXpos;
  testData.iSavedYpos = iYpos;

  // simply write the test structure out as binary data
  ofstream output;
  char ac[800];
  sprintf(ac, "%s/ProgData/Image%04d.prgData", directory, iImageNumber);
  output.open(ac, ios::binary | ios::trunc | ios::out);
  if(output){
    output.write((char*)&testData, sizeof(testData));
    output.close();
  }

  // now write out the x & y user boxed sliced data for each eye
  WriteSliceData(imgRaw, 1, directory, iImageNumber);
  WriteSliceData(imgRaw, 2, directory, iImageNumber);

  // write out the image (as a jpg)
  if(fSaveImages){
    sprintf(ac, "Captures/%s/Image%04d.jpg", directory, iImageNumber);
    ofstream ofile(ac, ios::binary);
    ofile << setformat( "JPG" ) << *pImgMod;
    ofile.close();
  }
}
```

- Information now only shows left or right eye

```
void ShowAutoDetectedInfo()
{
  vector<SFloatPoint> vPts;
  SFloatPoint ptEyeLHS, ptEyeRHS;
  int iVal = 1;

  // get found positions
  if(pEyeDetector->GetFoundEyes(vPts)){
    // we have eyes!
```

```
    printw("Eyes have been found at:-\n");

    // and show what we have found
    vector<SFloatPoint>::iterator it;
    for(it = vPts.begin(); it != vPts.end(); it++, iVal++){
      printw("    [%d] x = %.1f and y = %.1f\n", iVal, it->x, it->y);
    }

    // and validated versions
    iVal = pEyeDetector->GetValidatedFoundEyes(ptEyeLHS, ptEyeRHS);
    if(iVal){
      printw("The detector found %d eyes, from validation:-\n", iVal);
      if(ptEyeLHS.x > cfNO_EYE)
          printw("    Left eye at x = %.1f, y = %.1f\n", ptEyeLHS.x, ptEyeLHS.y);
      if(ptEyeRHS.x > cfNO_EYE)
        printw("    Right eye at x = %.1f, y = %.1f\n", ptEyeRHS.x, ptEyeRHS.y);
    }
   }
   else
     printw("No eyes automatically found\n");
  }
```

- Altered ShowFoundEyePositions

```
 void  ShowFoundEyePositions(image<unsigned char>& imgMod)
 {
  SFloatPoint ptEyeLHS, ptEyeRHS;

  // get found (and more valid) positions
  if(pEyeDetector->GetValidatedFoundEyes(ptEyeLHS, ptEyeRHS)){
    // convert to ints manually and for speed
    int iXlhs = static_cast<int>(ptEyeLHS.x + 0.5f);
    int iYlhs = static_cast<int>(ptEyeLHS.y + 0.5f);
    int iXrhs = static_cast<int>(ptEyeRHS.x + 0.5f);
    int iYrhs = static_cast<int>(ptEyeRHS.y + 0.5f);

    // need to draw them both
    for(int i=-30; i<=30; i++){
      if(ptEyeLHS.x > cfNO_EYE){
        imgMod.setPixel(iXlhs+i, iYlhs, 255);
        imgMod.setPixel(iXlhs, iYlhs+i, 255);
      }
      if(ptEyeRHS.x > cfNO_EYE){
        imgMod.setPixel(iXrhs+i, iYrhs, 255);
        imgMod.setPixel(iXrhs, iYrhs+i, 255);
      }
    }
  }
 }
```

EyeDetector.cc changes from Appendix 11

- Added new routine GetValidatedFoundEyes

```
// ------------------------------------------------------------------------
// The incoming SFloatPoints variables with the points for left eye and
// right eye where left eye is one that has x less than 1/2 screen width
// and right one greater.  If the eye is not found then the points will
// contain a large negative number (defined as cfNO_EYE and is a float).
// If there are more than one eye on a given side then that eye is deemed
// as not valid.
//
// In all cases the routine will return the number of eyes that FindEyes
// detected.
// ------------------------------------------------------------------------
int CEyeDetector::GetValidatedFoundEyes(SFloatPoint& ptEyeLHS,
                                                    SFloatPoint& ptEyeRHS)
{
  // we need working variables
  vector<SEyeData> vLeft;
  vector<SEyeData> vRight;
  vector<SEyeData>::iterator it;
  int iLeft = pImg->getWidth()/2;
  float fVal=0.0f;

  // ensure incoming variables are set to denote no eye
  ptEyeLHS.x = ptEyeLHS.y = ptEyeRHS.x = ptEyeRHS.y = 4 * cfNO_EYE;

  // split into either left eye or right eye
  for(it = vEyes.begin(); it != vEyes.end(); it++){
    it->sEyeX.GetCentre(fVal);
    if(fVal < iLeft)
      vLeft.push_back(*it);
    else
      vRight.push_back(*it);
  }

  // need only one in left hand side
  if(vLeft.size() == 1){
    vLeft.begin()->sEyeX.GetCentre(ptEyeLHS.x);
    vLeft.begin()->sEyeY.GetCentre(ptEyeLHS.y);
  }

  // and only one in right hand side
  if(vRight.size() == 1){
    vRight.begin()->sEyeX.GetCentre(ptEyeRHS.x);
    vRight.begin()->sEyeY.GetCentre(ptEyeRHS.y);
  }

  // return total number of eyes detectorfound
  return vEyes.size();
}
```

- **in specific.h added**

```
const float cfNO_EYE = -1.0f;
int GetValidatedFoundEyes(SFloatPoint& ptEyeLHS, SFloatPoint& ptEyeRHS);
```

main.cc

```
// ------------------------------------------------------------------------
// Reads image in from the .dv file every 100ms and sends to the eye
// detector.  The validated eyes are then shown as cross marks on the
// output image
// ------------------------------------------------------------------------


// ------------------------------------------------------------------------
// Included files and namespaces
// ------------------------------------------------------------------------
#include <GL/glut.h>
#include "image_xineinput.h"
#include "image_mesaoutput.h"
#include <sys/time.h>
#include <fstream>
#include "EyeDetector.h"

using namespace std;
using namespace mimas;

// ------------------------------------------------------------------------
// Test structure and data
// ------------------------------------------------------------------------
struct STest
{
  timeval tv;
  SFloatPoint ptEyeLHS, ptEyeRHS;
};

const int iMaxImage = 70;
STest asTestData[iMaxImage];

// ------------------------------------------------------------------------
// The main program
// ------------------------------------------------------------------------
int main( int argc, char *argv[] )
{
  int retVal = 0;
  int iImageNumber = 0;


  try {
    // Initialise GLUT.
    glutInit( &argc, argv );

    MMERROR(argc == 2, mimasexception, ,
            "Syntax: " << argv[0] << " <media resource location>" );

    // Open X11 display.
    x11_display display;

    const int delay = 100;

    // open movie
    image_xineinput<unsigned char> input( argv[1], "auto", "auto" );

    // create output window
    image_mesaoutput<unsigned char> window1(&display);

    // Allocate images
    image<unsigned char> img;

    // set up detector with pointer to image
    CEyeDetector eyeDetector(&img);

    // Loop for images.
    do {
      try {
           // get raw image
        input >> img;
```

```
        // get current time for later analysis
        gettimeofday(&asTestData[iImageNumber].tv, NULL);

        // detect all eyes
        eyeDetector.FindEyes(50); // 50 pixels either side of retinal reflection

        // now get validated eyes
        if(eyeDetector.GetValidatedFoundEyes(asTestData[iImageNumber].ptEyeLHS,
                                             asTestData[iImageNumber].ptEyeRHS)){
          // need data as integers (with round up/down)
          int xLHS = static_cast<int>(asTestData[iImageNumber].ptEyeLHS.x + 0.5f);
          int yLHS = static_cast<int>(asTestData[iImageNumber].ptEyeLHS.y + 0.5f);
          int xRHS = static_cast<int>(asTestData[iImageNumber].ptEyeRHS.x + 0.5f);
          int yRHS = static_cast<int>(asTestData[iImageNumber].ptEyeRHS.y + 0.5f);

          // need to draw them both
          for(int i=-30; i<=30; i++){
            if(asTestData[iImageNumber].ptEyeLHS.x > cfNO_EYE){
              img.setPixel(xLHS+i, yLHS, 255);
              img.setPixel(xLHS, yLHS+i, 255);
            }
            if(asTestData[iImageNumber].ptEyeRHS.x > cfNO_EYE){
              img.setPixel(xRHS+i, yRHS, 255);
              img.setPixel(xRHS, yRHS+i, 255);
            }
          }
        }

        // output the img
        window1 << img;// << mimas::delay(delay);

        // see if at max allowed for us
        if(++iImageNumber >= iMaxImage)
          break;

      }
      catch(exception &e){
        cerr << "Warning: " << e.what() << endl;
        window1 << mimas::wait( delay );
      };
    }
    while(input);
}
catch (exception &e){
    cerr << "Error: " << e.what() << endl;
    retVal = 1;
};

// normalise the time wrt first entry
for(int i=1; i<iImageNumber; i++){
    if(asTestData[i].tv.tv_usec < asTestData[0].tv.tv_usec){
      asTestData[i].tv.tv_usec += 1000000;
      asTestData[i].tv.tv_sec--;
    }
    asTestData[i].tv.tv_sec -= asTestData[0].tv.tv_sec;
    asTestData[i].tv.tv_usec -= asTestData[0].tv.tv_usec;
}
asTestData[0].tv.tv_sec = asTestData[0].tv.tv_usec = 0;

// show how many images
cout << "Analysed " << iImageNumber << " images\n" << endl;

// output times to a file
ofstream output;
output.open("TestTimesAndValues.csv", ios::binary | ios::trunc | ios::out);
if(output){
    cout << "Saving data" << endl;
    // write out header first
    output << "seconds,usec,left eye x,left eye y,right eye x,right eye y" << endl;

    // now write out the data
```

main.cc

```
    for(int i=0; i<iImageNumber; i++){
      // output time
      output << asTestData[i].tv.tv_sec;
      output << ',' << asTestData[i].tv.tv_usec << ',';

      // output left eye data
      if((asTestData[i].ptEyeLHS.x > cfNO_EYE) && (asTestData[i].ptEyeLHS.y > cfNO_EYE))
        output << asTestData[i].ptEyeLHS.x << ',' << asTestData[i].ptEyeLHS.y;
      else
        output << ',';
      output << ',';

      // output right eye data
      if((asTestData[i].ptEyeRHS.x > cfNO_EYE) && (asTestData[i].ptEyeRHS.y > cfNO_EYE))
        output << asTestData[i].ptEyeRHS.x << ',' << asTestData[i].ptEyeRHS.y;
      else
        output << ',';

      output << endl;;
    }
    output.close();
    cout << "Data saved" << endl;
  }

  return retVal;
}
```