# Evaluation

## Introduction

Once the system is complete, it is important to analyse how the finished product compares to the original brief agreed by the customer and producer. To this end I have made my own judgements on the performance of the system against the original “SMART” objectives, as well as collecting feedback from the customer (proof of which is included in the appendix) and discussed what the implementation has succeeded in doing and what it has failed to do. I have also discussed what possible changes and additions could be made to the system and how easy it would be to expand the project.

## Comparison of Implementation with “SMART” objectives

In the analysis section a number of “SMART” objectives were laid out for the system. These objectives are compared to the finished system here, explaining to what extent they were met, to what extent they were not met, and why.

#### The Laplacian method must be implemented

The laplacian method was included in the system. It was decided that the 3x3 pixel kernel method was to be used as opposed to the 5 pixel ‘+’ kernel method as this allows for edge detection in 4 directions as opposed to two.

#### A Sobel method could be implemented

The Sobel method was not included in the system. It was decided not to include this as it was not part of the AS physics course, so time and resources would be better spent on algorithms that were in the course, such as the Laplacian algorithm.

* + 1. For the sobel method the user should be able to change for what value it is said an edge has been detected.

N/A (see 1.1.2).

* + 1. The user must be able to choose whether the edges are white on black, or black on white.

Originally this was to be applicable to the sobel method, however this was not implemented so this functionality was not relevant in the way it was meant to have been. However, it became clear that for some users the laplacian procedure, which is always white on black edges, caused an unclear result for some users. As a result an algorithm that would flip the colours of an image, i.e. a negative function, was added. As with the greyscale procedure this could be used in conjunction with or independently of the edge detection algorithm

#### The system must be able to convert an image to greyscale before any edge detection is carried out

A separate greyscale function was added to the system which can be used independently of the edge detection algorithm if necessary.

#### The conversion of images to edge detection images should take no longer than 30 seconds

When executed on hardware that met reasonable specifications using images within a reasonable size this was achieved. On very sub par hardware or very large Images this sometimes may not be the case, but these cases should not occur to the normal user.

#### The system should be able to smooth images using the mean technique

This method was implemented into the program as it is described in the AS physics course.

#### It could be possible to enter how many times the smoothing process should be repeated, if any.

It was decided that as the program was designed for educational use not for commercial image editing that this repeating function was not essential so it was not implemented in the release. It is possible to manually apply the algorithm multiple times to the same image by applying the algorithm once, texecutedsferring the output image of the input image box, and then applying the algorithm again however, which is useful to see compare the difference on the strength of smoothing achieved by repeating the algorithm. In future it would be fairly simple to implement the repeated smoothing functionality if the situation changed.

#### It should take no longer than 30 seconds for images to be smoothed

When executed on hardware that met reasonable specifications using images within a reasonable size this was achieved, however the algorithm did complete slower in comparison to the other algorithms. On very sub par hardware or very large Images this sometimes may not be the case, but these cases should not occur to the normal user, though even relatively small images could take long times compared to other algorithms.

#### The system should be able to remove noise using the median method

This method was implemented into the program as it is described in the AS physics course.

#### There should be as little loss of data as possible during noise removal

From testing it appeared to the human eye that it was mainly noise that was removed form the image, and the original appeaexecutedce of the image was maintained. The only exception is one pixel thick drawn lines, but this should not normally occur in reasonably high quality photographs

#### It could be possible to enter how many times the noise removal process should be repeated, if any.

This was not implemented for the same reasons as it was not implemented for smoothing (see 1.2.2)

#### It should take no longer than 30 seconds for noise to be removed from an image

When executed on hardware that met reasonable specifications using images within a reasonable size this was achieved. On very sub par hardware or very large Images this sometimes may not be the case, but these cases should not occur to the normal user.

#### It should be possible for the system to convert red pixels in an image to grey pixels

This was implemented into the program using a 50% grey value to convert pixels above some red threshold into. The red threshold is a combination of a >Red and <Green, <Blue check.

#### It should be possible for the user to change the red tolerance of the process

A separate form was implemented into the program to facilitate this. The for allows users to use a slide bar or text box to enter a value for the red tolerance, and preview what this tolerance is before setting it.

#### It could be possible for the user to select an area for which red eye is to be reduced

Along with all other selection tools, this was not implemented into the final product. The reason for this is that these features would require time to implement that was thought to be better spent on improving the quality of the rest of the program. As an educational tool the program still shows very well how red eye reduction works, and so functionality more suited to users who wanted a program for photo editing was not incorporated.

#### It should take no longer than 30 seconds for red eye to be reduced.

When executed on hardware that met reasonable specifications using images within a reasonable size this was achieved. On very sub par hardware or very large Images this sometimes may not be the case, but these cases should not occur to the normal user.

#### The user should be able to save an edited image without overwriting the original

A save feature where the file name can be chosen using a dialogue box was implemented, and so images can be saved without overwriting the original.

#### The user could be able to undo changes to an image for at least 10 image edits

Instead of an undo function where multiple ‘History’ images had to be stored a two image system was implemented into the system. The image on the left is ‘Image 1’, the input image, and is the image that the program will always open to. The Image on the right is ‘Image 2’, the output image, and is the image that the program will always save to. There are right and left arrows at the bottom of the screen allowing the user to set either image to the other image. This system allows the user to see a direct before and after comparison, as well as keeping the original image stored in the program. I feel this helps the system function as an educational tool, and also makes it more compact as only two images are stored at any one time.

#### The user should be able to revert to the original image

This idea was replaced by the system described above (see 2.2)

#### There could be certain selection tools

As is described in the red eye reduction objective (see 1.4.3), the selection tool feature was omitted as it was thought this would be better suited to a commercial image editing package as opposed to an educational image processing system.

#### The user could be able to preview changes before they are made

This aim was also replaced by the two image solution (see 2.2)

#### It should not crash or have problems frequently

From testing, the system did not appear to crash or have errors usually, however very large images could cause the system to become unresponsive, especially on lesser hardware. This was dealt with by specifying minimum system requirements, and a maximum recommended image size

#### It should be clear where different tools are

Feedback from the end-user suggests that this is generally the case, however until the system is distributed to a larger user vase, this objective may be unclear as to its completeness. This being said, no glaringly obvious flaws appeared in the end-user feedback.

#### It should not be possible to easily lose an edited image or delete an existing one

The two image system makes it difficult to lose an edited image within the program, however on exit of the program there is no prompt as to whether or not you wish to save your work. Because the system contains sample images and is designed to be used as an educational tool, it was decided not to include major safeguards against loss of data, however it ma be desirable to implement these in the future

#### The system must contain brief summaries of which each function does

A separate form contains all educational information in the program. Each algorithm (except ‘Negative’, which was not in the original brief) has its own entry of text in this resource, and users can have the educational form open along with the main form for image editing.

#### The system should not damage other systems

Every effort was made to make sure the finished system did not contain anything that would unintentially damage other systems, or any viruses that may have leaked into the distributable zipped folder. To date neither I nor my end user have had our systems at all damaged by the program.