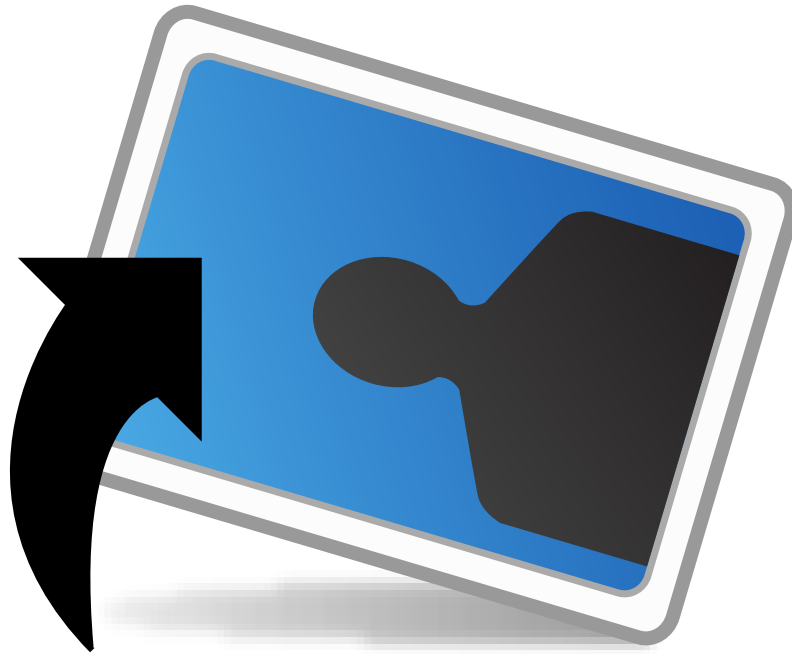


Automatic Photograph Orientation



Hugo King

Candidate Number: 91103

Multimedia & Digital Systems,
The Department of Informatics.

Supervisor: Dr Paul Newbury

April 2007

Statement of Originality

This report is submitted as part requirement for the degree of Multimedia & Digital Systems at the University of Sussex. It is the product of my own labour, except where indicated in the text. The report may be freely copied and distributed, provided the source is acknowledged.

Signed:

Date:

Acknowledgements

I would like to thank the following people:

- Dr. Paul Newbury
For his supervision and direction throughout this project;
- David Tschumperlé - Developer of the Clmg C++ library
For the extremely useful Clmg library and his help and assistance;
- Members of the following forums:
 - i. Ars Technica Programmers Symposium;
 - ii. Clmg Help (English);
 - iii. CocoaBuilder.com;
 - iv. CocoaDev.com.

For all the technical advice they gave;

- My fellow Multimedia & Digital Systems students
For their advice, and for having to suffer through such projects as well;
- My parents
For their advice, support, and proof-reading;
- All the parties referenced in this report
For their work, and for publishing that work.

Counts

Total words	11,839
Summary words	117
Total pages	94
Appendices pages	20

Summary

This project details research into feature extraction of digital images with the goal of automatically orientating images. An application was developed towards this goal to correctly orientate images of two categories - indoor, and outdoor. An iterative approach was taken to development, incorporating research, requirements analysis, detailed design diagrams, implementation, and user-testing.

The application was implemented using C++, and Objective-C++ with the Apple Cocoa windowing system. The image processing, written in C++, utilises the CImg library and comprises of custom written orientation determination algorithms, based on several areas of research.

The system is shown to achieve a 92% success rate for orientating outdoor images, with an average success rate of 66% for both indoor and outdoor images.

Table of Contents

1.0 Introduction	7
1.1 Digital Imaging	7
1.2 Report Structure	8
2.0 Professional Considerations	9
3.0 Requirements Analysis	10
3.1 The Problem	10
3.2 Existing Solutions	10
3.2.1 Hardware	10
3.2.2 Research	11
3.2.3 Critical Analysis	11
3.3 User Needs	13
3.4 Requirements	13
3.5 Programming Language Justification	14
3.6 Project Scope	15
4.0 Research Theories	16
4.1 Feature Extraction	16
4.1.1 Colour Analysis	16
4.1.2 Complexity Analysis	17
4.1.3 Edge Analysis	18
4.2 The Application of Analysis	19
4.3 Inconsistencies	20
4.4 Alternative Methods	20
5.0 Development Plan	21
6.0 System Design	22
6.1 High Level System Diagram	22
6.2 Image Processing Execution	23
6.3 User Interaction	25
7.0 Implementation	29
7.1 The Interface	29
7.1.1 Usability Principles	29
7.1.2 The Interface Controller	32
7.2 Image Processing	37
7.2.1 Methods of Analysis	37
7.2.2 The Main Method	38
7.2.3 Other Methods	38
7.2.4 Colour Determination	39
7.2.5 Complexity Determination	40
7.2.6 Edge Determination	41
7.2.7 Decision Merging	42
7.3 Program Interconnection	43

8.0 Testing and Improvement	45
8.1 Initial Testing	45
8.1.1 Usability Testing	45
8.1.2 White-box Testing	46
8.2 Improvements from Initial Testing	47
8.3 Application Release	49
9.0 Final Testing	54
9.1 User Requirements Testing	54
9.2 System Performance	56
9.2.1 Image Processing Time	56
9.2.2 CPU and Memory Usage	57
10.0 Experiments	58
10.1 Rotation Performance	58
10.1.1 Indoor	59
10.1.2 Outdoor	61
10.2 Confidence Performance	62
10.3 Collaborative Performance	64
11.0 Conclusions	68
11.1 Meeting the Objectives	68
11.2 Improvements and Extensions	69
11.2.1 Improvements	69
11.2.2 Extensions	70
11.3 Alternative Methodologies	70
11.4 Overall Success of the Project	70
12.0 Bibliography	72
13.0 References	73
14.0 Appendices	75
14.1 Appendix A - Testing Results	75
14.1.1 Usability Surveys	75
14.1.2 Time Performance	79
14.1.3 CPU Performance	79
14.1.4 Memory Performance	79
14.2 Appendix B - Experiment Results	80
14.2.1 Rotation Results	80
14.2.2 Confidence Results	85
14.2.3 Collaborative Results	88
14.3 Appendix C - Source Code	89
14.3.1 Interface Controller	89
14.3.2 Image Processing	92
14.3.3 Header Files	94

1.0 Introduction

This project aims to research into feature extraction of digital images, with the objective of automatically determining a correct orientation for images where the orientation is incorrect. The ultimate goal is to create a stand-alone application with the functionality to open, process, and save images, correcting their orientation.

Digital images - especially digital photographs - are often incorrectly orientated, a good example of this being digital photographs which have been taken in a portrait orientation but when displayed are in a landscape orientation by default. To a human the orientation of an image is often obvious, because we have the ability to easily recognise shapes and patterns within an image deriving from these the correct orientation of the image. The challenge is mimicking these natural feature extraction methods in a computer as, unlike the human brain, a computer has no natural ability to recognise such features.

The intention in this project is to construct an algorithmic approach to extracting features from an image, and making a determination from these features as to the correct orientation. This functionality will be built into a simple application, although the emphasis of the project is that of research into feature extraction.

1.1 Digital Imaging

Digital imaging holds several advantages over traditional analogue imaging. Storing an image digitally allows for faster access and indexing, quicker editing - even for complex tasks, and the ability to share images - i.e. making digital copies and sending these over networks. These advantages have led to the proliferation of digital imaging, demonstrated by the fact that the market share of digital cameras overtook that of traditional cameras in the U.S. in 2003¹.

It is now possible for the average consumer to touch-up digital photographs, where only fifteen years ago this was the realm of only the photographic studio. The proliferation of consumer grade digital cameras, sales of which are expected to hit 82 million units by 2008², has given access to digital photography on a budget. This area has developed so fast however that some problem areas with digital photography have been overlooked.

The question of image orientation is one of these problem areas. This problem is commonly caused when shooting digital photographs - as many cameras orientate the image in landscape view irrespective of whether the camera was held to take a portrait photograph, i.e. on its end. The result of this is a mixture of portrait and landscape photographs, with no determination of which is which. Traditionally a user will cycle through images in a photo-management application, rotating individual photographs as appropriate, but this is a time consuming task.

The motivation for this project is to explore whether there is a more efficient way of conducting this process, perhaps automating a large portion of it, and whether such a method would produce satisfactory results.

1.2 Report Structure

The structure of this report is as follows:

2. An introduction to the professional considerations that were involved in this project;
3. A discussion of the requirements of the project, including an examination of existing solutions in this field, and research that has previously been conducted;
4. Detailed discussion of the theories of feature extraction;
5. A plan of the development process;
6. An explanation of the program design;
7. A discussion of the program implementation process;
8. Explanation of the testing and improvements made;
9. Explanation of the final testing;
10. Discussion of the experiments conducted;
11. Conclusions drawn from the results found;
12. A bibliography;
13. References.

At all points throughout this report where information has been gathered from other sources credit is made to the source, which is also listed in the references section.

2.0 Professional Considerations

There are relatively few professional issues to consider in this project, due to the fact that it is largely research based, without expectation to implement a software application for retail. There are however the following considerations, as set out by the British Computer Society (BCS)³, that apply to the scope of the project:

- Code of Conduct, Section 3⁴ - *"You shall have regard to the legitimate rights of third parties."*

This project gives consideration to the rights of any parties from which information is gathered through research, by correctly referencing and attributing any knowledge to such parties. This project is also aware of any copyright attached to any photographs used as test subjects.

- Code of Conduct, Section 15⁵ - *"You shall not claim any level of competence that you do not possess. You shall only offer to do work or provide a service that is within your professional competence."*

This project undertakes work of an appropriate level in all aspects. This project does however strive to implement new technologies that are involved in its scope, where it is felt that they are relevant.

- Code of Practice, Section 2⁶ - *"Maintain Your Technical Competence", "Manage Your Workload Efficiently"*

This project undertakes to use new skills and to keep up to date with technological advances, in order to implement a more efficient and innovative system. It also undertakes to use time efficiently with respect to the requirements, and to not over-burden the workload.

- Code of Practice, Section 3.1⁷ - *"Programme/Project Management"*

The project adheres to the code laid out in this section when defining, planning, and closing the project.

- Code of Practice, Section 4.2⁸ - *"Research"*

The project undertakes to only proceed with research into areas which can be justified, acknowledge the contribution to the project of research of other people and organisations, and share the results at the discretion of the author through published papers.

3.0 Requirements Analysis

The following chapter describes the process of determining the requirements of this project.

3.1 The Problem

In recognising the orientation of an image the human brain is performing a very complicated series of deductions relying on natural recognition of shapes and patterns, and the importance which we associate to them. The natural approach when designing a system to mimic this process is to perform the same analysis - looking for features in an image - and then basing a judgement on those found. The problem is extracting the patterns from an image and relating these to image orientation.

In different categories of images different feature patterns emerge, for instance in an image of a group of people there will be many oval shapes - the heads - whose longest dimension will be orientated vertically. However, in an image of a landscape scene we might expect there to be a lighter shade at the top of the image. These two rules may converge but as with most patterns will not hold true in every instance.

Most images will fall into a category where one or more approach will provide a strong answer, therefore, the implementation of detecting the correct orientation of an image must use several approaches to extracting patterns

3.2 Existing Solutions

There are no commercial applications which perform automatic image rotation, including plug-ins for applications such as *Adobe Photoshop*⁹ or *GNU GIMP*¹⁰. The likely reason for this is that there is no guarantee with such automatic rotation solutions of a 100% success rate - this would be required in a commercial application of such technology to achieve marketability.

3.2.1 Hardware

In the growing digital camera market a solution to the orientation problem is becoming increasingly implemented in the hardware. Digital cameras - especially those at the higher end of the market - are now being fitted with a tilt switch to detect the orientation of the camera when shooting a photograph, this is then stored as the orientation of the image.

While this feature is becoming more widespread in digital cameras the problem of orientation is still present in lower end digital cameras, legacy digital images, and digitised traditional images.

3.2.2 Research

There have been several research projects conducted into the area of automatic image orientation, and feature extraction of digital images. Several of the research papers in these areas are summarised below:

- *Z. Su (Z. Su, 2004) - Automatic Image Orientation Detection*¹¹
This paper details the extensive use of peripheral block segmentation of images, along with the colour moment (CM) system of analysis. An edge Detection Histogram (EDH) method is also explored here, and the effectiveness of the use of the RGB, HSV, and LUV colour spaces in these methods is outlined.
- *S. Lyu (S. Lyu et al, 2005) - Automatic Image Orientation Detection with Natural Image Statistics*¹²
This paper takes a new approach to feature extraction using decomposition of colour channel data with a single internal block at the centre of the image - this is to accommodate for varying image sizes. The paper deals mainly with neural networking classifiers.
- *Vailaya et al (Vailaya 1999d) - Automatic Image Orientation Detection*¹³
This paper primarily explores a Bayesian Learning framework approach to neural networking for this problem, but does describe a colour moment segmentation process. The paper describes the group's use of CM in LUV space, Colour Histogram in HSV space, EDH, and "MRSAR texture features".
- *Zhang (Zhang et al 2001) - Boosting Image Orientation Detection with Indoor vs. Outdoor Classification*¹⁴
This paper details the CM and EDH feature extraction methods but also looks at comparisons between using larger amounts of segments in the implementation of these methods. It finds that - when division is made into NxN blocks - for CM $N=5$ "has almost the same accuracy as $N=9$ and $N=10$ ", and for EDH "12 directions also yielded a good result compared to 36 directions".
- *Vailaya (Vailaya and Jain 2000) - Automatic Image Orientation Detection*¹⁵
This paper expands on the 1999 paper - *Vailaya et al (Vailaya 1999d)* - and details more thoroughly the neural networking methods outlined in the previous paper.

3.2.3 Critical Analysis

One approach that is common to the automatic orientation problem is that of neural networking - using a learning framework to mimic intelligence in the system - in this way an application can be developed to orientate images by being 'taught' with example sets. This project has chosen not to pursue this path of research as it is felt

that there is not adequate time for such an approach to be implemented, given the skill set available. For this reason this section only analyses algorithmic solutions.

In the paper *Automatic Image Orientation Detection* (Z. Su, 2004)¹⁶ the method of colour moment (CM) analysis is used throughout as an efficient way to classify images. This method makes use of the pattern of colours in an image which can denote the arrangement of features, for example it is common for the area at the top of an image to be predominantly of a blue colour - the sky. This paper describes the process of determining colour moments - which involves segmenting the image and extracting the predominant colour in the respective segments, each of these being colour moments.

This paper describes the use of colour moments taken from only the periphery of the image - from the top, bottom, left, and right - looking for a predominance of blue. Both the RGB and HSV colour spaces are used in the colour moment method here, and the paper mentions that these methods proved superior in comparison to using the LUV colour space.

An edge detection histogram (EDH) method is also used in this papers approach. The paper describes that in EDH an edge determination algorithm is run at each pixel location of the image to determine if there is (and in which direction) an edge at that location. Several bins are used - depending on the EDH complexity - to count for each increment of direction about 360° , plus one bin for any location not determined to be of an edge. The particular type of edge detection algorithm is not elaborated on in the paper, but it is concluded that the use of 13 bins is close in effectiveness to the use of 37 bins.

In the paper *Boosting Image Orientation Detection with Indoor vs. Outdoor Classification* (Zhang et al 2001)¹⁷ particular attention is paid to results for indoor and outdoor images, with the aim of improving the balance of orientation determination between these two distinct groups. The methods used are again CM and EDH and the paper determines that in both CM and EDH a segmentation of 5x5 blocks is sufficient, and that for EDH 13 bins are sufficient for each of these blocks. This paper considers all blocks within the image, making no distinction between periphery blocks and internal blocks.

This paper makes an interesting deduction from the target source of images. The paper recognises that in most cases images are acquired from a digital camera. *"When taking photos, peoples often hold the cameras in the normal way, i.e. they hold the cameras in 0° to take horizontal photos, or rotate the camera by 90° or 270° to take vertical photos. But they seldom rotate the camera by 180° . This phenomenon results in that the images uploaded from digital cameras are seldom in the wrong orientation of 180° ."* This observation can refine the precision of the

classification of image orientations by counting any determined orientation of 180° as incorrect and so setting it as 0°.

Both papers comment on the performance of comparative analysis methods, in final testing they conclude that no single method is more accurate for a range of image types, and that any combined method will not produce 100% accurate results in any situation. Both papers however claim to achieve a success rate of over 90% on combined analysis methods.

Both of these papers thoroughly explore the use of both CM and EDH, with promising results, this can then be taken as an indication of CM and EDH being viable options when implementing an automatic orientation detection method in this project. With further research these analysis methods could be implemented in the manners described above, or simplified for this project. The possible implementation of these methods will be explored in a later chapter.

3.3 User Needs

The intention behind the project is that user input should be extremely limited - in so much that the application's core should be fully automated, leaving the only user input to be to specify the images to process.

The interface provided for the application should follow basic human-computer interaction principles as follows:

- *The system status should be clearly visible* - when processing images this should be made clear, possibly through a progress indicator;
- *The system should be flexible and efficient to use* - the system design should allow various image file-types to be processed and the number of actions that must be performed from application launch to processing should be low;
- *The interface should be aesthetically pleasing and of a minimalist design* - the user should not be presented with unneeded options and a cluttered interface;
- *The system should contain useful help and documentation* - this will facilitate the ease of use and help recovery from errors.

3.4 Requirements

The objective of this project is to create a simple application to process batches of digital images, orientating them correctly, and saving them to disk with the new orientation.

The scope of this project is extremely broad, with many categories of images to be potentially processed, each requiring different approaches to analyse their correct

orientation. Due to time constraints therefore the requirements of the project are as follows:

R.1 To develop a software package with a complete graphical user interface following human computer interaction principles, to facilitate ease of use and efficiency.

The software package should provide the following functionality:

- i. The ability to select a series of images to process;
- ii. Save the processed images to a directory;
- iii. Progress indication of this process;
- iv. A measure of confidence in the returned rotation for each image.

R.2 To implement an effective automatic rotation algorithm for:

- i. Outdoor photographs;
- ii. Indoor photographs.

This allows for several techniques of image analysis to be researched;

R.3 To achieve a success rate of roughly 80% for such a complete algorithm.

These requirements are realistically achievable, but also challenging. There are several directions in which this project can be extended, including the following:

A.1 Broadening the application to process more categories of images, and so increasing its usefulness;

A.2 Porting the application to multiple platforms - as the intended operating environment is a UNIX-based system;

A.3 Increasing the algorithm success rate.

These extensions are time dependant and as such will only be undertaken if time constraints allow.

3.5 Programming Language Justification

The language chosen for development of the application is C++, this is as a basis for the image processing. The interface to the application is developed for the *Apple* OS using the *Cocoa*¹⁸ system with *Objective-C++*, although the back-end image processing algorithms are written fully in C++. The advantages of using C++ are as follows:

- The language is cross platform compatible, allowing the core of the application to be developed on the available *Apple*¹⁹ system and facilitating adaptation of the application to run on multiple systems;
- The language has many constructs and classes provided for image analysis and manipulation across a range of systems. Use will be made of the *CImg* open source class library, provided by David Tschumperlé²⁰;
- The language is widely used in industry and it is felt that it will be a useful skill to learn.

There were other languages that were considered in the approach to this project, in particular *Java*²¹ and C. Despite previous experience using *Java*²², and the extreme flexibility of C, the use of C++ proved to be the sensible choice. Since as well as providing an object-orientated approach, C++ provides a great deal of flexibility, especially through the use of open source libraries such as the *CImg*²³ library, giving greater reign to tune the language to the needs of the project, as well as experience of a new language.

3.6 Project Scope

A foreseen problem area is the existence of abstract photographs such as macro photographs, and aerial photographs - these having the property of an almost indiscernible orientation, rendering the possibility of designing an algorithm to orientate them correctly extremely unlikely. Due to this the project does not explore the possibility of analysis of these images.

The project will also only look at image rotations of a simple set - meaning that the only orientations considered will be on the vertical and horizontal planes, at 0°, 90°, 180°, and 270°.

As previously commented this project will only pursue algorithmic solutions to the problem of automatic image orientation, and not look to develop a neural networked system.

4.0 Research Theories

This chapter expands on the theories proposed in other research projects, critical analysis of which can be found in section 3.2.3, discussing the principles and ideas within this research. These theories contribute towards the approach that this project takes in determining image orientation.

4.1 Feature Extraction

The process of feature extraction is that which, through analysis of an image, determines the specific properties of that image - its features - which are then used to make some judgement about the image in question. In the case of this project the judgement to be made is that of the image's correct orientation.

Through extracting the features of an image the process of contextual recognition that the human brain performs on making judgements about images can be mimicked. This is done through determining patterns in the features extracted from the image, and comparing these to 'real-life' patterns, from these results a judgement can be made on the image.

There are several methods of feature extraction that are possible, the following described methods are pertinent to this project.

4.1.1 Colour Analysis

The colours within an image can hold a lot of information about that image. Patterns are often formed in colours, for instance areas of predominant colour can indicate a certain feature. Through analysing the colours within an image, and the relative spread of these, features can be extracted.

A common method of feature extraction from colour information is that of colour moments. In colour moment analysis the image is segmented and the colour of several of these segments analysed, each segment result being a colour moment, these colour moments can highlight patterns in the image - features.

There are several methods of segmentation that can be employed in colour moment analysis, each having advantages and disadvantages:

- **Complete segmentation**

In this method the entire image is segmented into $N \times N$ blocks. The advantage of this is that a more complete analysis is made of the image, increasing the accuracy of the feature extraction. The disadvantage is that the entire image must be analysed, and much of the data gathered could be redundant.

- **4-sides segmentation**

In this method each edge of the image is taken as a colour moment - i.e. the top,

right, left, and bottom N^{th} of the image. The advantage of this is that the processing time is reduced - as less of the image is being analysed. The disadvantage is that some accuracy in the pattern analysis may be lost, this however depends on the segmentation level - the size of N .

- **Central segmentation**

In this method the central 4 segments are analysed, after segmentation into $N \times N$ blocks, these are taken as the colour moments. The advantage of this method is that the processing time is greatly reduced due to the smaller analysis area. The disadvantage is that if features are not present in the central area of the image they will be missed.

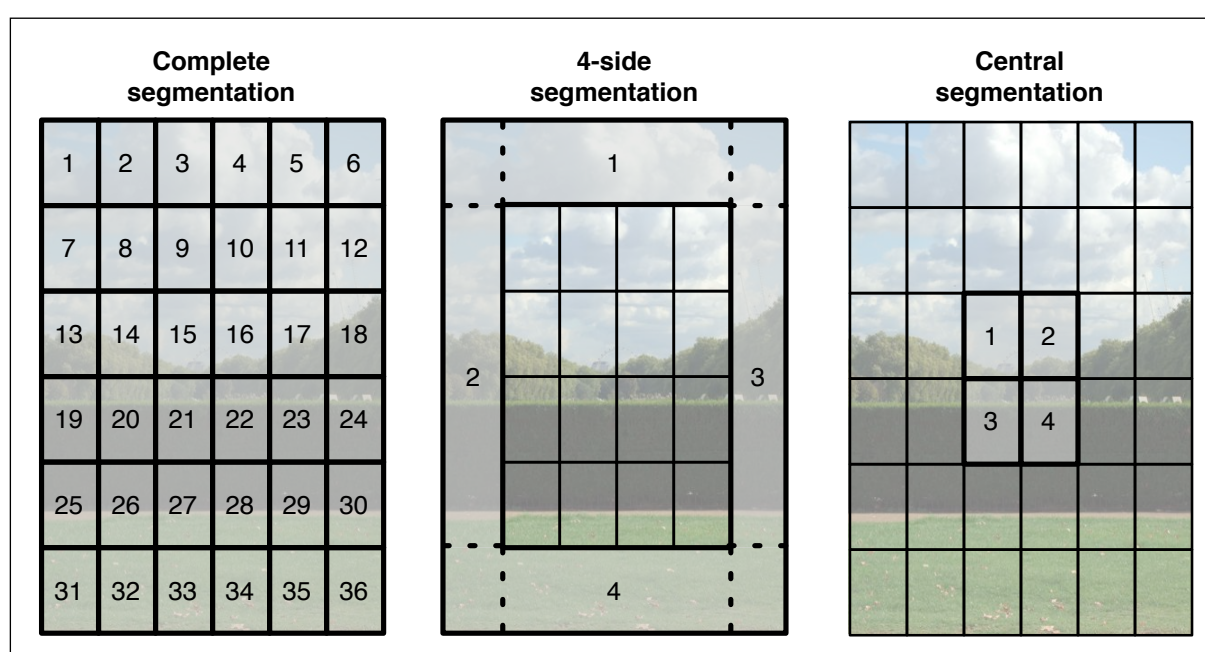


Figure 4.1 - The different methods of colour moment segmentation

The method of colour moment analysis can be applied in measurement of the intensity of the moment rather than colour, for instance measuring in the HSV colour space - using V (or intensity) as the measurement parameter - different patterns, of intensity, can be found. This approach can be useful and is often used in parallel with colour moment analysis in the RGB colour space to find common patterns in both colour and intensity.

4.1.2 Complexity Analysis

Complexity analysis, sometimes referred to as texture analysis, is a technique whereby the variance of pixel complexity is measured. Each pixel is compared to its neighbours and from this the level of complexity at that point in the image is

measured. Through this method the complexity of the image in different segments is determined, highlighting possible patterns.

The complexity level in a segment of an image can indicate various features. In a typical image it is expected that the lower half will be more complex than the upper half, a small area of complexity in an image could indicate the presence of a person at that location (especially if the background is solid), or an area of complexity could indicate any feature that a user may be interested in - this can be useful in scientific imaging.

The method of measuring complexity can be approached in several ways, in a colour image each colour channel can be analysed at each point - giving a measurement of the complexity at each point on each colour channel, or the intensity values can be taken and the complexity at each point derived from these. The first method can be useful if complexity is being examined in a specific colour, i.e. complexity in foliage - the green channel. The second method is a more generic approach which looks for any complexity in the image.

4.1.3 Edge Analysis

Edge analysis is a method by which an image is processed for distinct edges - indicated by a change in pixel intensity in a certain direction. From the results of edge detection features are highlighted, such as distinct shapes, or horizon lines.

There are various methods of edge detection including sobel, roberts-cross, and compass²⁴. The principles of these algorithms is the same - the image is analysed each pixel location at a time, using a convolution matrix covering the surrounding pixels, the algorithm is then run giving a reading of the edge direction at that pixel location. Figure 4.2 shows the application of a sobel edge detection using convolution matrices.

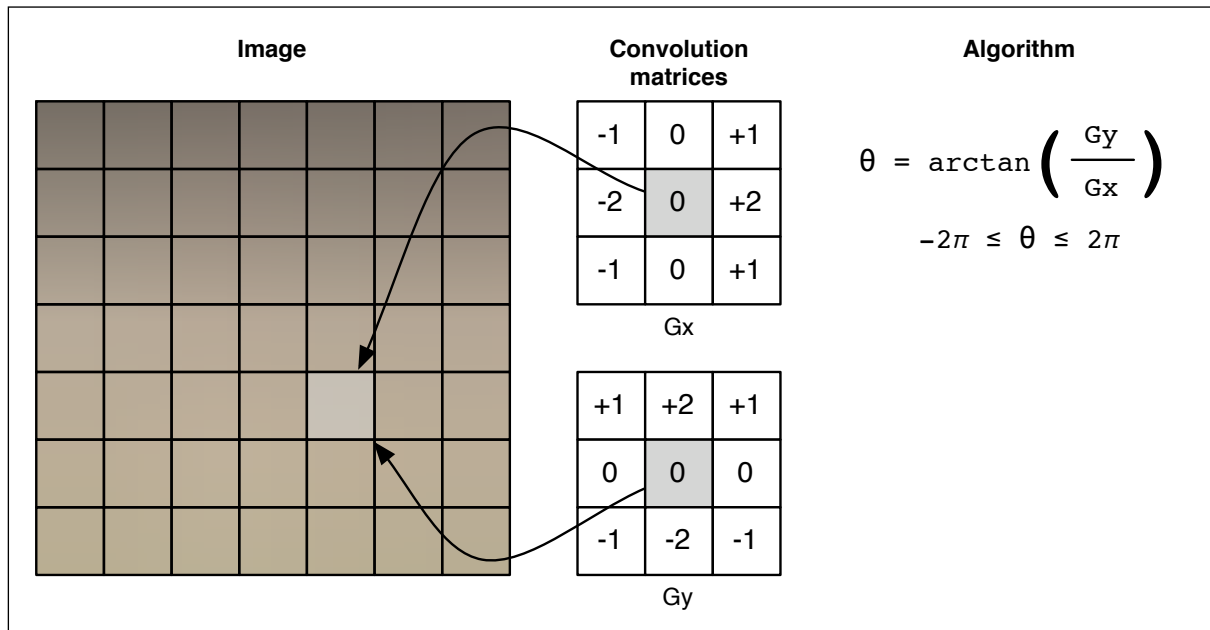


Figure 4.2 - Applying convolution matrices to an image for sobel edge detection

The method shown in figure 4.2 sums all the products of the convolution grid with its respective pixel value, to give the convolution result at that pixel location - Gx and Gy. These are then used to determine θ with the given algorithm, θ is a measurement of the edge direction at that pixel location.

A common application of edge detection is to count the number of edges found in each orientation over the entirety of the image - indicating the predominant edge orientation - and comparing this to that of the expected predominant edge orientation. By this process the orientation of the image can be deduced as correct or incorrect.

4.2 The Application of Analysis

When looking at determining the correct orientation of an image using the analysis techniques described above the results must be compared to expected patterns were the image orientated correctly. The following patterns are expected in a correctly orientated image:

- The predominant colour at the top of the image will be blue and of a greater intensity;
- The lower half of the image will be more complex than the upper half;
- The predominant edge orientation will be vertical.

These patterns do not hold true for every image but are a good basis when determining the correct orientation. If the patterns found when analysing an image

differ from these it can be inferred that the image is incorrectly orientated, and based on the actual pattern the image can be rotated to match the expected pattern.

4.3 Inconsistencies

The patterns described above are not true for every image but do commonly work on certain categories of images. It is common for strong vertical edges to be found in architectural images and indoor images, however it is less common in outdoor images. It is common for there to be a predominance of blue at the top of a daytime landscape image but not in a night-time landscape image, or an indoor image. It is common in most images for the lower half to be more complex than the upper half, but this does not hold true for aerial images.

The problem therefore is that these methods of analysis may produce results contradictory to one another. To overcome this problem a method of deciding upon an orientation, when several are indicated, is necessary. The most commonly implemented of these being 'one takes all', in this method the most popular result among the various options is the final output, the popularity of this option can however be based upon many variables, varying with the application.

There is always the possibility of misleading results in image analysis, for instance in an image consisting of a landscape with a lake in the lower half the blue colour of the water could be interpreted as sky, by a colour analysis algorithm, and a false result returned. This is a problem that is important when evaluating the analysis methods.

4.4 Alternative Methods

A very common approach to the problem of image analysis, and especially that of orientation detection, is the use of a neural network. This is where the system is 'taught' the best way to process and analyse images, and which decisions to make in which scenarios, through a learning framework.

This method is considered a natural approach to the problem as it uses artificial intelligence, with the desired result that it will mimic the human cognitive process as closely as possible, therefore achieving better results than can be hoped for with an algorithmic approach. This project has not chosen to implement a neural network approach to this problem as it was felt that the time and experience constraints were too great.

5.0 Development Plan

The process of developing a system can be complex, more so when the deployment of the system must conform to a deadline, as in the case of this project. To facilitate efficient development a careful plan must be followed, with thought given to each stage.

A typical development will transition through four basic stages, those of research, design, implementation, and testing. Good design will take an iterative approach to development²⁵, with each stage being cycled through several times.

Each phase of development should encapsulate a specific area of the system. By developing in this way each area of the system can be more thoroughly explored - through research and design - before being implemented, and then through testing be refined to ensure correct operation. This iterative approach can limit errors and inconsistencies in the system and streamline the development process enabling deadlines to be met with greater ease.

The development plan for this project follows this iterative process and is illustrated in figure 5.1.

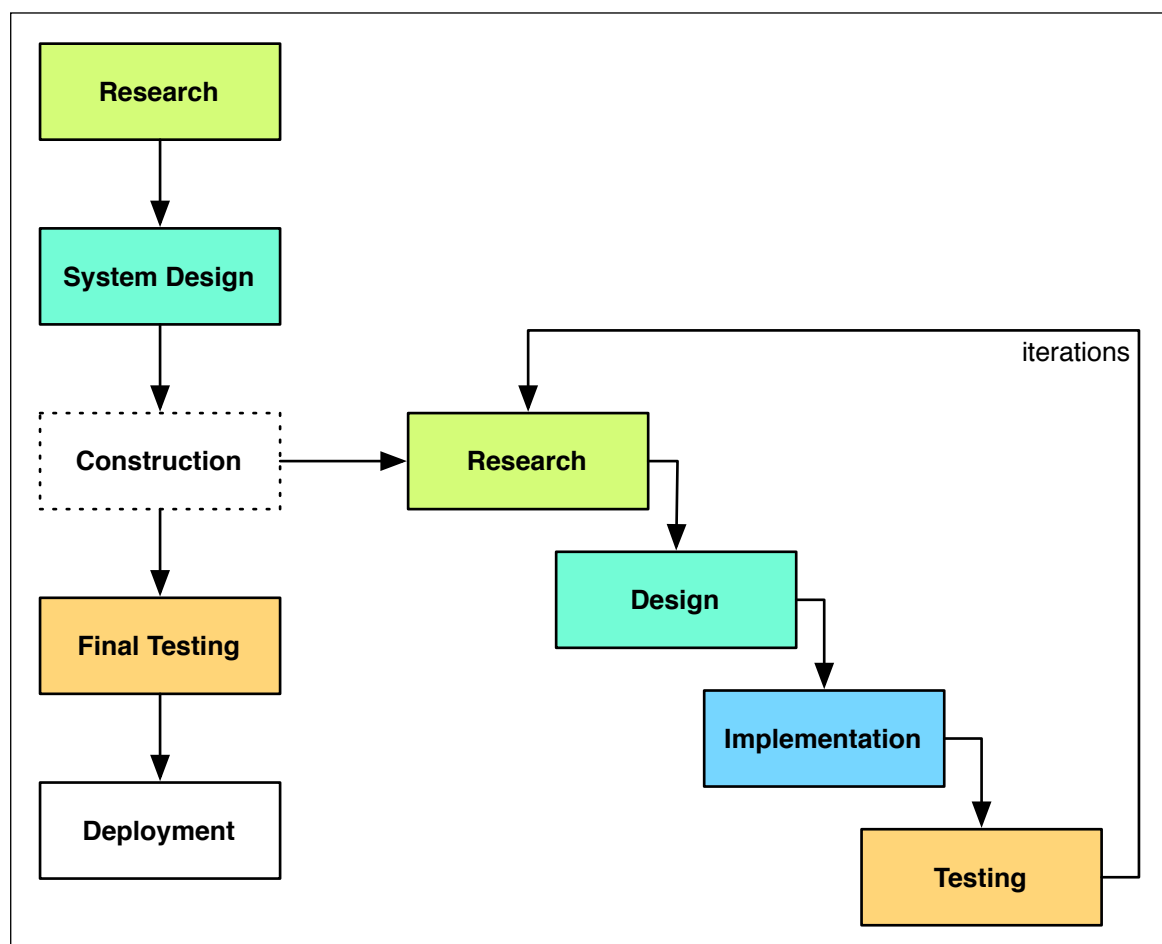


Figure 5.1 - The application development process

6.0 System Design

The program design can be split into three distinct sections, those of the interface, the interface controller, and the back-end image processing. The interface allows the user to specify the images to process, and the parameters with which to process these. The interface controller is an intermediary between the interface and the image processing. The image processing back-end performs all the feature extraction and orientation determination.

While the interface developed for this system is implemented only to facilitate research into the image analysis techniques it's design and implementation will be approached as though the system is intended for distribution. For this reason the interface must be developed with attention to user requirements and usability.

6.1 High Level System Diagram

The user will interact with the system through a graphical user interface (GUI), from this the analysis variables will be controlled and processing initiated. The GUI will communicate with the image processing through the interface controller to specify the images to process and the variables with which to do so. The image processing will run the determination algorithms and then run decision merging from the output of these. The results will then be returned to the user.

This is illustrated in figure 6.1.

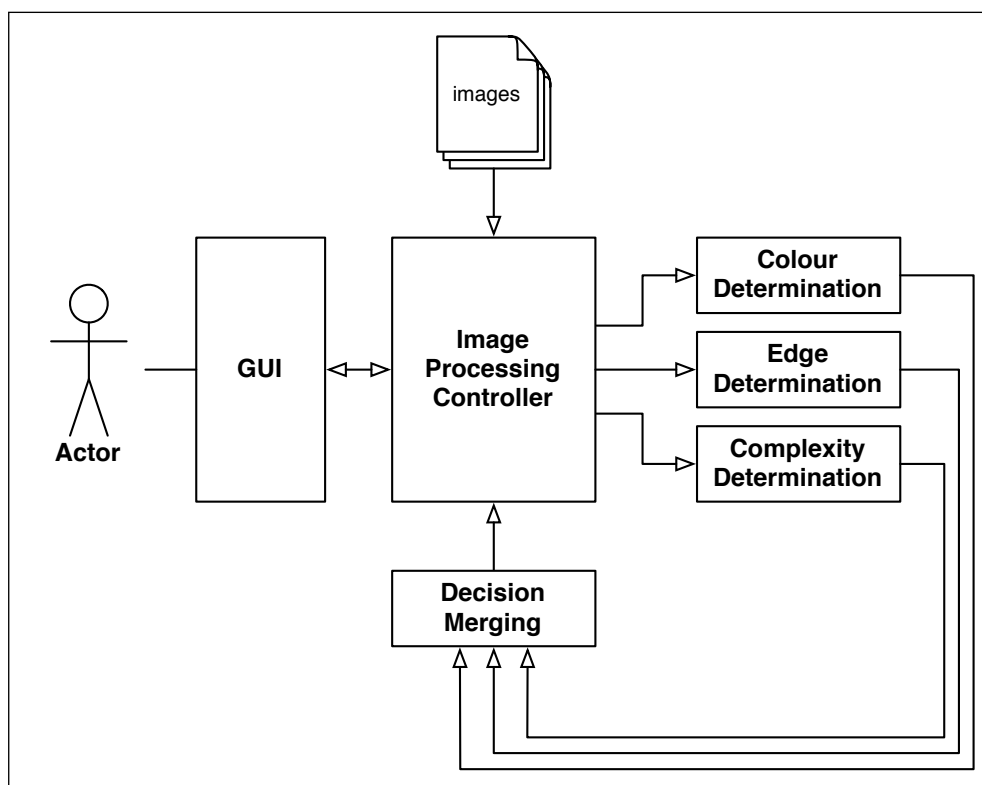


Figure 6.1 - A high level diagram of the designed system interconnections

6.2 Image Processing Execution

When the image processing is started the system transitions through 4 basic steps:

1. Fetching the list of images from the interface;
2. Extracting the features from the image;
3. Making an orientation determination upon the features found;
4. Rotating and saving the image - this is dependant upon the user specified preferences.

Steps 2-4 being repeated for each image.

This is illustrated with the flowchart in figure 6.2.

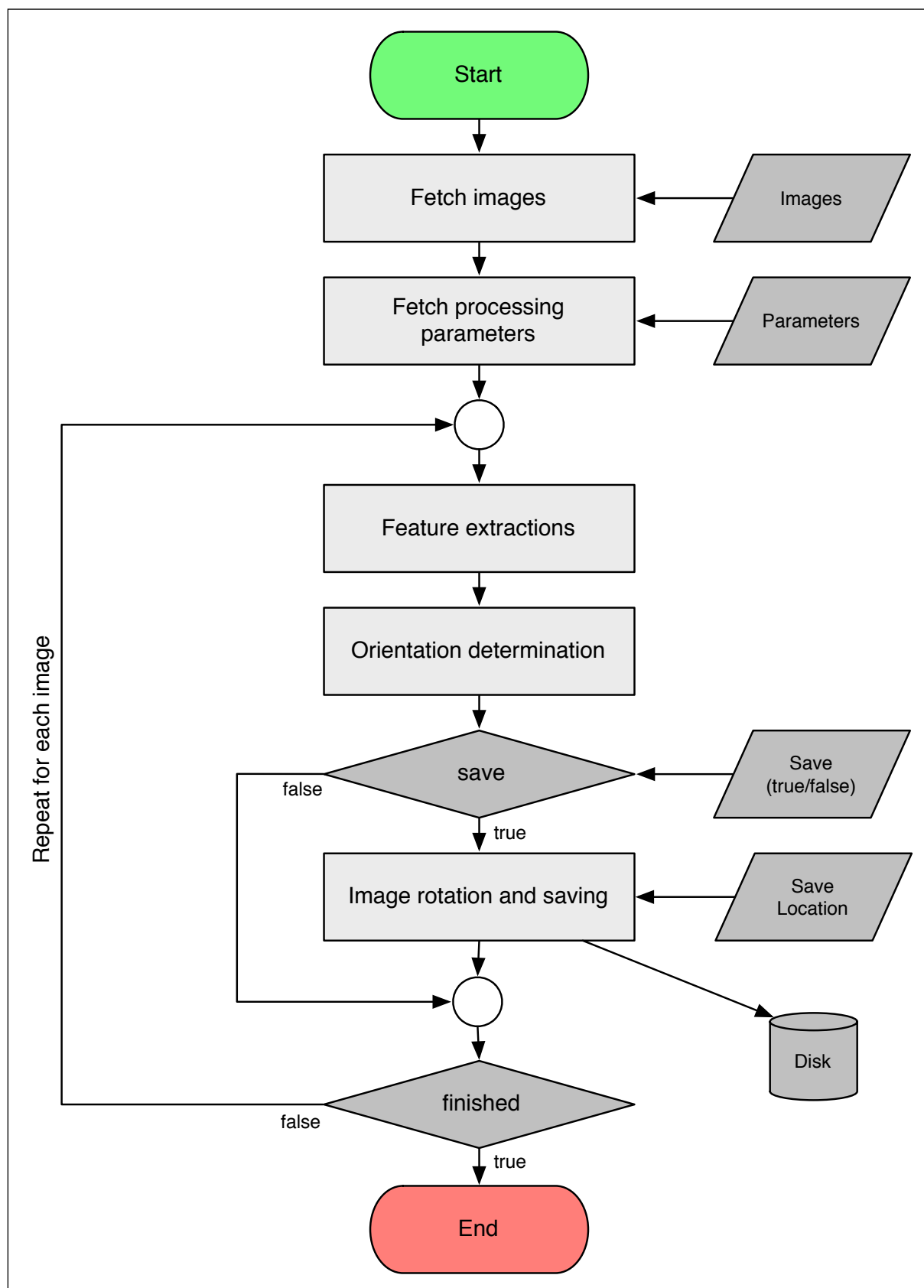


Figure 6.2 - Flowchart showing the image processing execution stages

6.3 User Interaction

The user interacts with the system through a graphical user interface (GUI) and performs a limited number of operations using interface controls. An actor is restricted to only users as all interaction with the system is through the GUI.

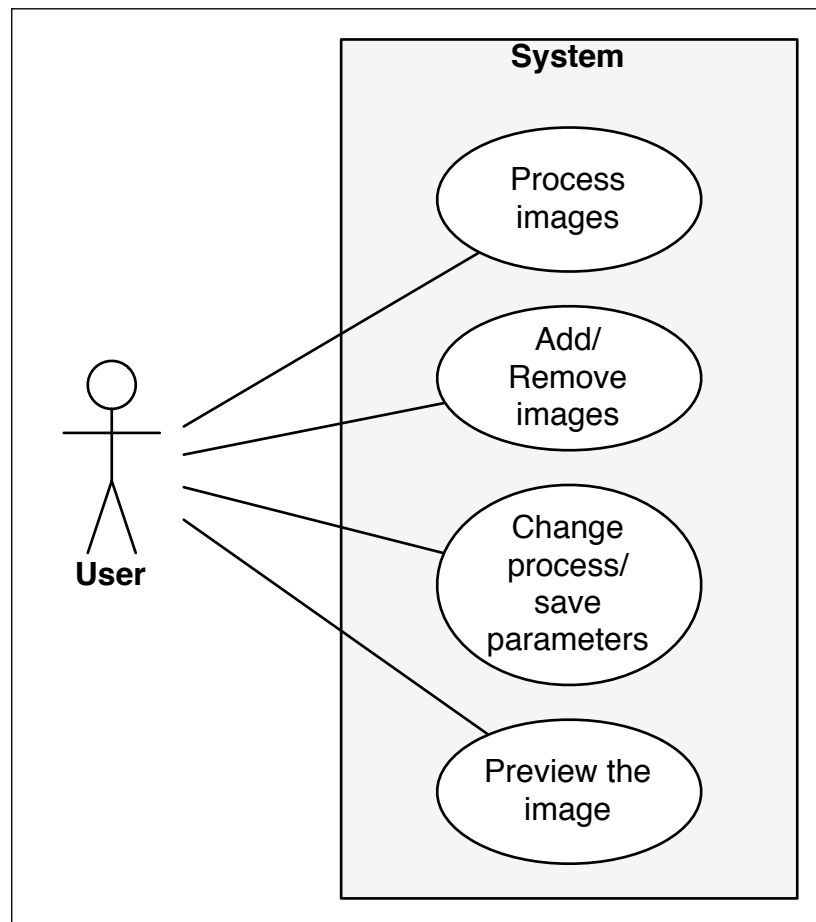


Figure 6.3 - A use case diagram of the system

There are four basic system interaction scenarios as shown in the following figures - figure 6.4, figure 6.5, figure 6.6, and figure 6.7.

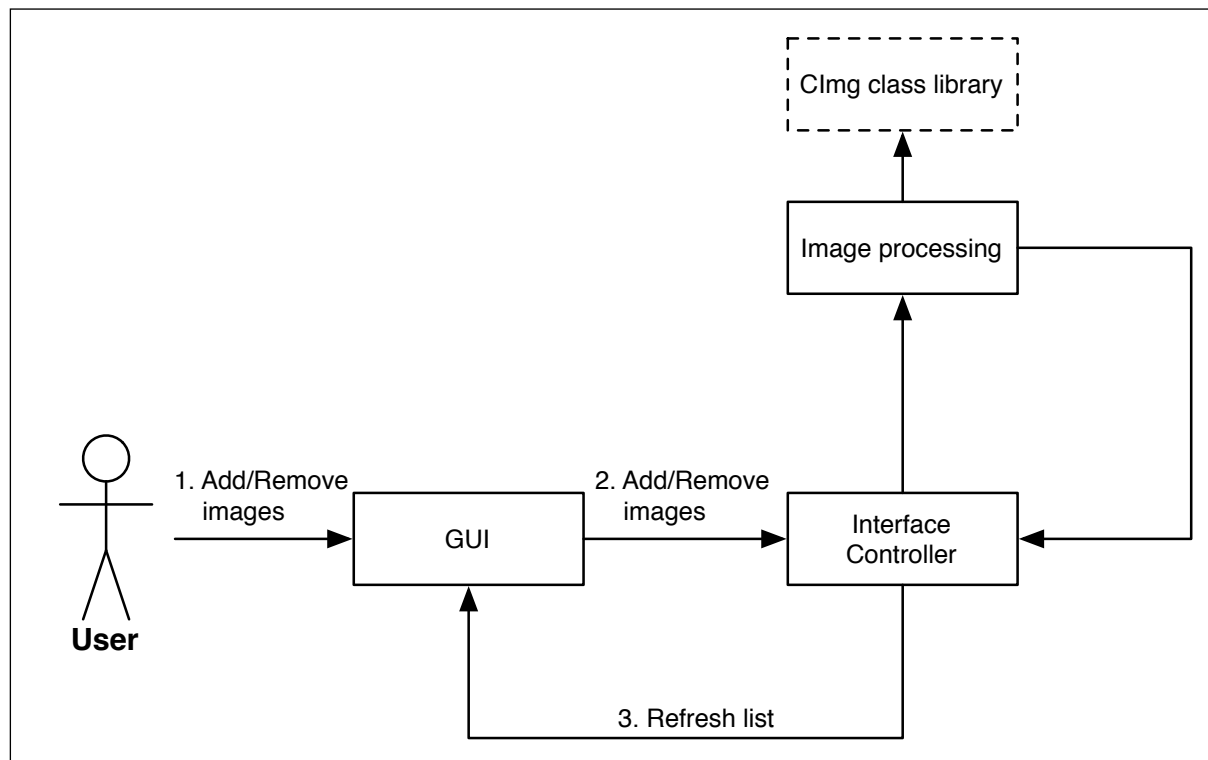


Figure 6.4 - An interaction diagram of the system for "Add/Remove images"

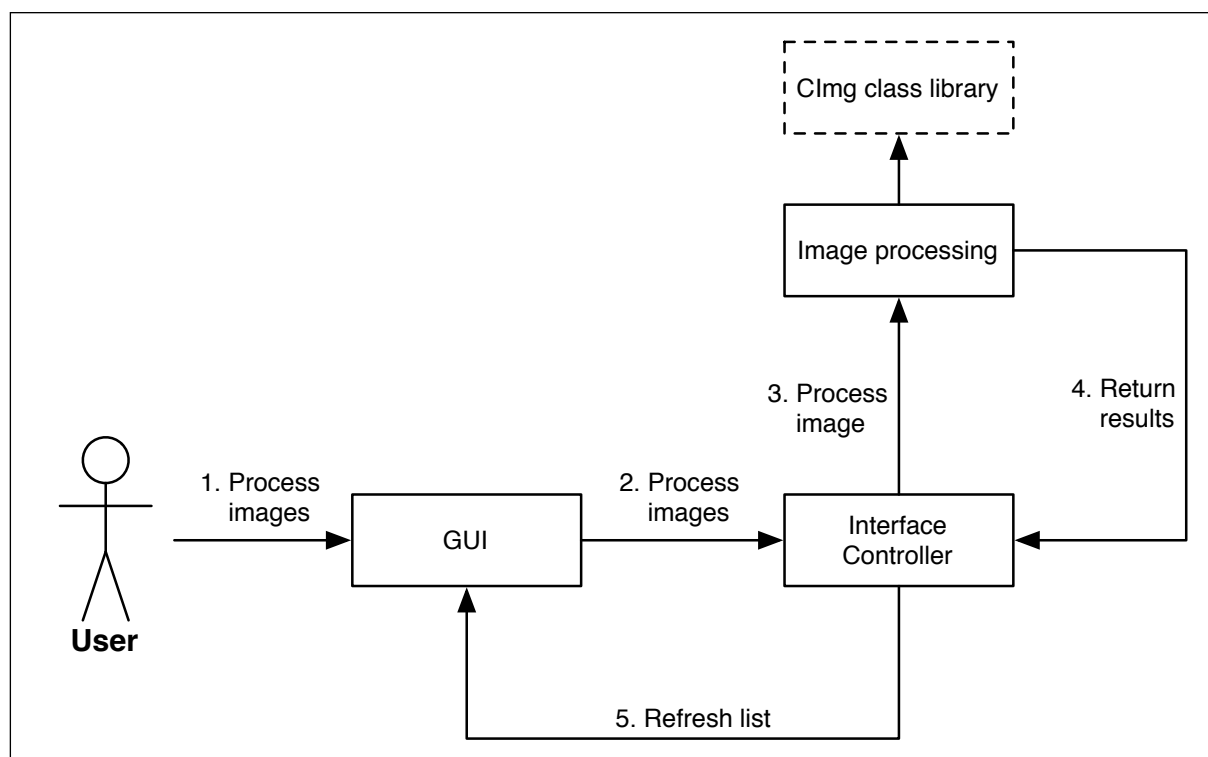


Figure 6.5 - An interaction diagram of the system for "Process images"

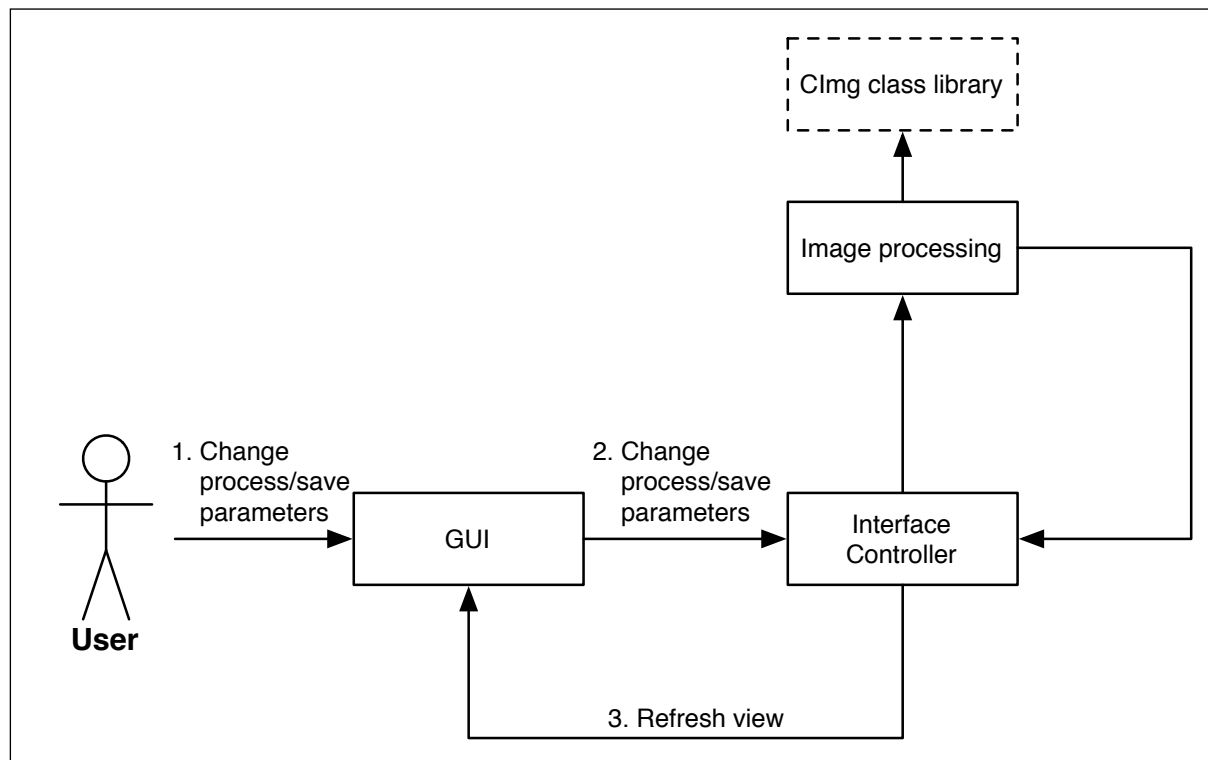


Figure 6.6 - An interaction diagram of the system for “Change process/save parameters”

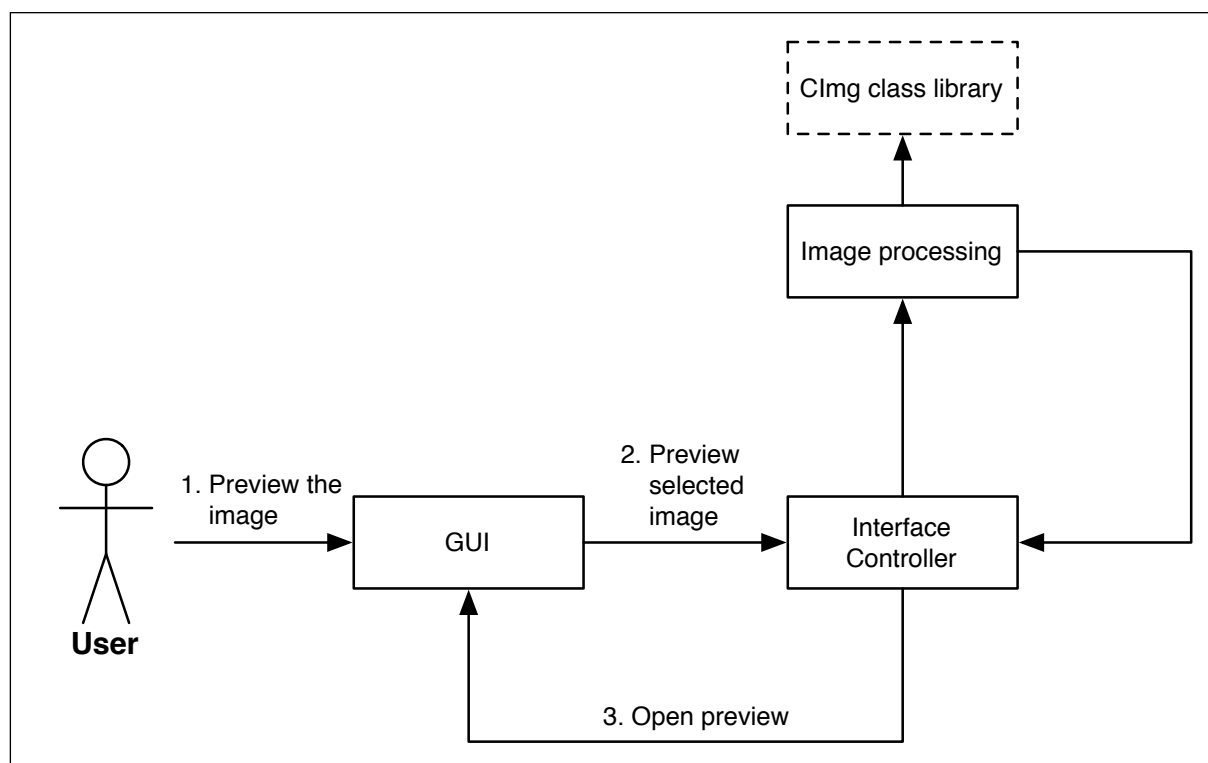


Figure 6.7 - An interaction diagram of the system for “Preview the image”

Each of these scenarios requires separate control from the GUI. A draft of the GUI design is given in figure 6.8 showing all user controls necessary for the interaction scenarios above.

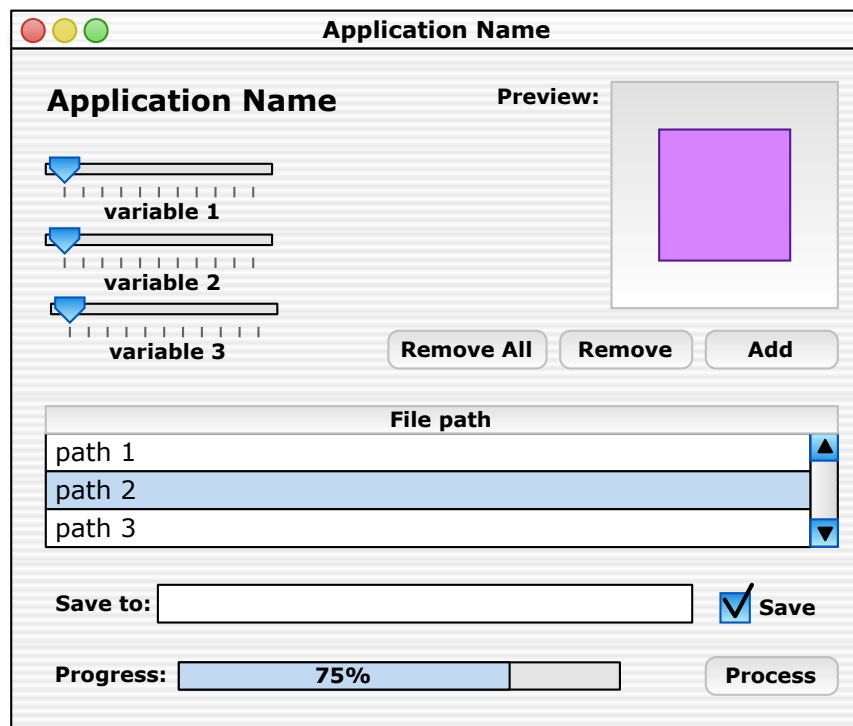


Figure 6.8 - A draft design of the systems graphical user interface (GUI)

7.0 Implementation

This chapter examines the implementation of the design principles from the previous chapter including how user control is dealt with, how data is managed within the processing algorithms, and how parameters are managed between the user interface and the image processing.

7.1 The Interface

The user interface provides the control mechanism for the system, this can be considered to consist of two parts the GUI itself, and the interface controller. The interface controller sits behind the GUI and is triggered, through its methods, to react to user input from manipulation of the controls on the GUI.

7.1.1 Usability Principles

The GUI is designed to follow human-computer interaction usability principles, allowing the user to easily understand and manipulate the system. Nielsens ten heuristics²⁶ - the 'rules of thumb' for designing user friendly systems - are as follows.

1. *Visibility of system status;*
2. *Match between system and the real world;*
3. *User control and freedom;*
4. *Consistency and standards;*
5. *Error prevention;*
6. *Recognition rather than recall;*
7. *Flexibility and efficiency of use;*
8. *Aesthetic and minimalist design;*
9. *Help users recognise, diagnose, and recover from errors;*
10. *Help and documentation.*

The interface design follows these heuristics with attention to 'aesthetic and minimalist design' and 'recognition rather than recall' in general, the following features emphasise specific heuristics.

Keyboard shortcuts

The program makes extensive use of keyboard shortcuts allowing the user to manipulate the program using only the keyboard, these shortcuts are indicated in the program menus alongside the operation they relate to.

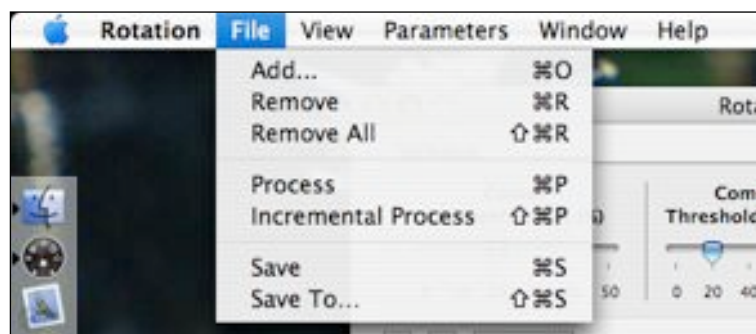


Figure 7.1 - Showing the applications menu and indication of keyboard shortcuts

The inclusion of keyboard shortcuts allows advanced users to use the program much faster, fulfilling ‘...efficiency of use’. The shortcuts used in the program are also consistent with shortcuts in many other applications, allowing users experienced at using shortcuts to easily translate those skills to the program.

Progress indication

It is important that the user be informed as to the programs operation and current state - following the heuristic ‘visibility of system status’ - there are two features enabling this:

1. A progress indication bar;
2. The disabling of controls during processing.

These two features are triggered when the system is processing images - which can take several seconds per image - updating through this process, and releasing control to the user when the process is finished.

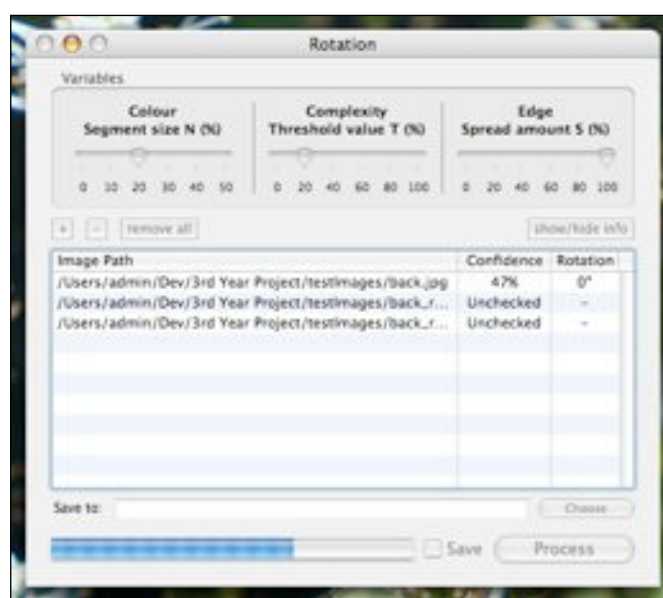


Figure 7.2 - The application interface during processing

Validation and errors

Where the program relies on input from the user it must be validated to ensure that it is of an expected type, i.e. of a type that the system will be able to process. The user interface handles most of this checking through both validation of inputs and restricting possible inputs.

When selecting files to be processed (opening files) the file-picker dialogue is restricted to only allow selection of certain file types, these being JPEG, PNG, and GIF, thus restricting the input of files to only compatible types. The same principle is also applied when a user must specify a location to save processed images to, this time the file-picker dialogue is restricted to the selection of single folders.

There are several instances where user operation is restricted to eliminate vulnerabilities in the system integrity, in these cases the user is informed with the use of alert panels. For instance when the user tries to remove files and none are selected, or when the user tries to process the images when none have been opened. Through this behaviour the systems fulfils the heuristics of 'error prevention' and 'visibility of system status'.

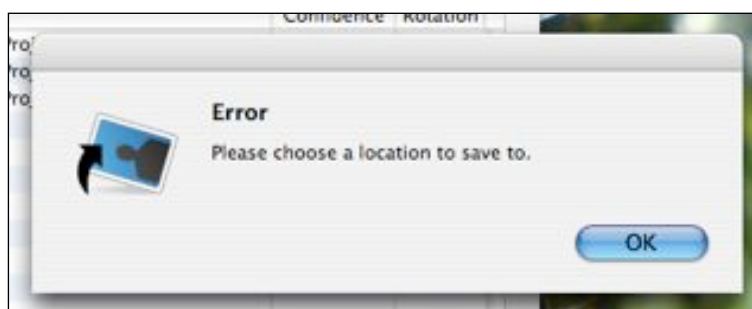


Figure 7.3 - A warning displayed when no save location is specified

Where the system encounters an error due to some problem in the execution of the program the user is provided with a clear error warning and it is ensured that the program does not suffer from any unrecoverable problem.

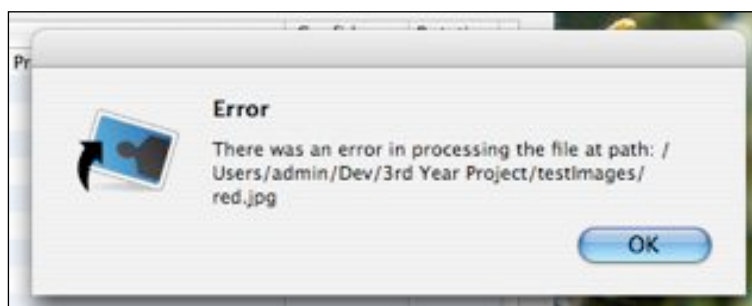


Figure 7.4 - A warning displayed when the application encounters an error processing

Image preview/information

The user is provided with the ability to preview the processed or unprocessed image with additional information, this is displayed in a 'drawer' attached to the right of the main window. This additional information is not necessary to the operation of the program but provides the user with a more visual way of checking the program results. This fulfils 'user control and freedom' and 'aesthetic and minimalist design'.

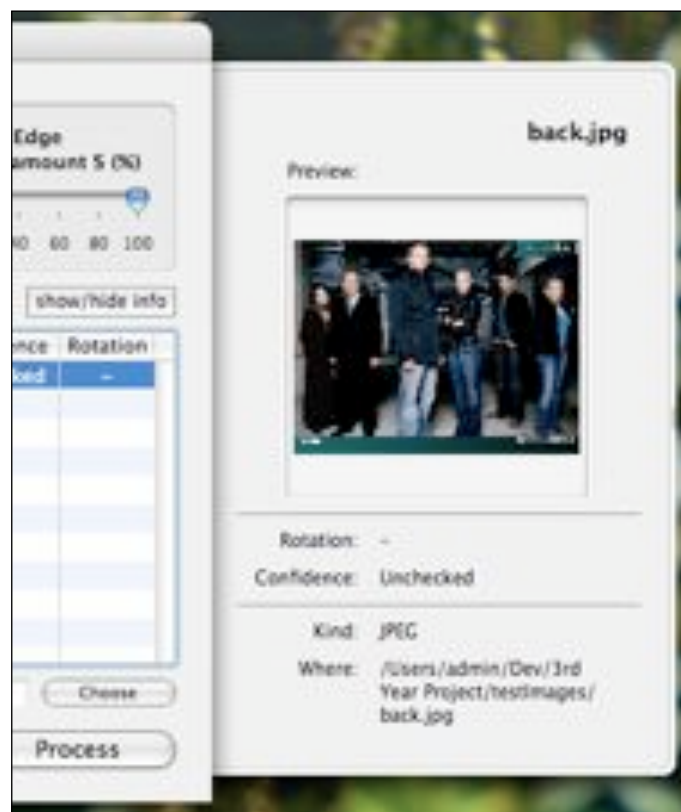


Figure 7.5 - The info drawer displaying an image preview and information

7.1.2 The Interface Controller

The following sections describe the code implementation of the interface controller. The iterative process of where some were problems encountered during testing and were solved is also described. In places description is made of elements of the *Apple Inc. Cocoa* windowing system²⁷.

Full and commented code is given in appendix C, section 14.3.1.

File-picker dialogues

The file-picker dialogue is used to provide the user with a file system navigation interface from which files/folders can be selected, this is used in the program to specify files to open and save locations. The system makes use of the Cocoa class

NSOpenPanel which is provided for use as a file-picker and facilitates consistency across all Mac OS²⁸ applications.

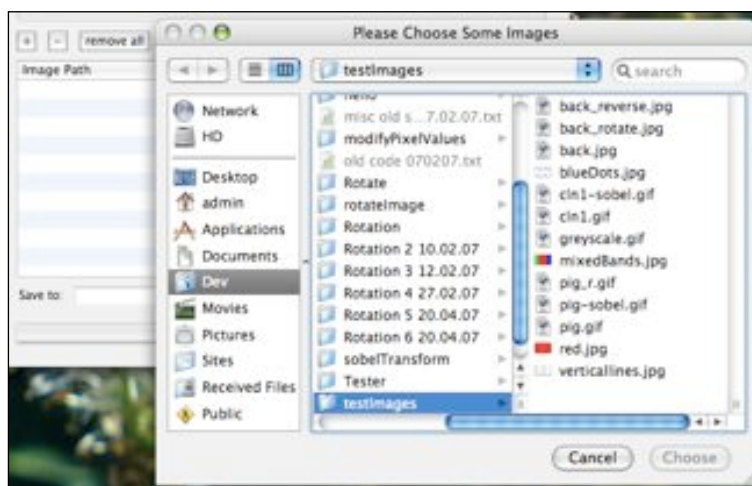


Figure 7.6 - The file-picker to choose images to open

The NSOpenPanel object supports several methods to define its functionality, these include allowing restriction to choosing only files and/or folders, allowing multiple selections, setting button text, and setting allowed file types. The NSOpenPanel when closed will then return the selected files file paths in an array - an NSArray object specified at creation.

Initially it was found that the program would crash if the 'Cancel' button was clicked on an NSOpenPanel, this was as the program assumed the array would always contain file paths on exit from the file-picker, inserting a check on the return value from NSOpenPanel solved this problem.

```

//create NSOpenPanel object
NSOpenPanel *panel = [NSOpenPanel openPanel];

//set its properties:
//can choose multiple valid image files but not directories
//set panel text items
[panel setCanChooseDirectories: NO];
[panel setCanChooseFiles: YES];
[panel setAllowsMultipleSelection: YES];
NSArray* allowedFileTypes = [NSArray arrayWithObjects:@"jpg", @"JPG",
    @"gif", @"GIF", @"png", @"PNG", nil];
[panel setTitle: @"Please Choose Some Images"];
[panel setPrompt: @"Choose"];

//create variables to store results in
NSArray* fileList;
int chosen;

//open the panel, storing the results in the variables
if (chosen = [panel runModalForTypes: allowedFileTypes]) {
    fileList = [panel filenames];
}

```

Figure 7.7 - Example code showing the implementation of a file-picker

Working with tables

The list of files displayed in the interface uses the Cocoa object `NSTableView`, this object requires a data-source to display from - this is a class containing a data structure which can be accessed using several overridden methods. For this reason the interface controller overrides these methods, acting as the data structure, and stores the data for the `NSTableView` in a `NSMutableArray`. When data in the table must be refreshed the method `reloadData` is called on the `NSTableView` object, which then calls the overridden methods of the data-source. This allows for complete control over what is displayed in the table from the interface controller.

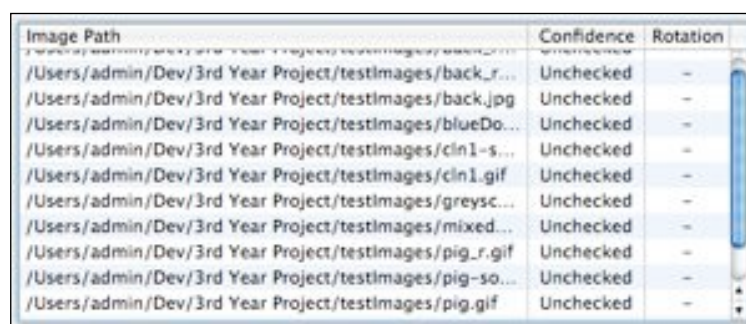


Image Path	Confidence	Rotation
/Users/admin/Dev/3rd Year Project/testimages/back_r...	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/back.jpg	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/blueDo...	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/clin1-s...	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/clin1.gif	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/greysc...	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/mixed...	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/pig_r.gif	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/pig-so...	Unchecked	-
/Users/admin/Dev/3rd Year Project/testimages/pig.gif	Unchecked	-

Figure 7.8 - The file list of the interface

The list of files also contains the determined rotation and confidence of each file. The data structure behind the list stores a series of `NSMutableDictionary` objects, each representing a file. Each of these `NSMutableDictionary` objects contains three object-key pairs for file path, rotation, and confidence.

To store that a file hasn't yet been processed two error codes are used in the place of 'image rotation' and 'confidence' in the `NSMutableDictionary` object, these being 361 and 101 respectively. When a call is made to the `NSTableView` to refresh data these codes are looked for and indicators that the file hasn't been processed are returned to be displayed.

Removing files

When the user wishes to remove files from the processing list the method `deleteFiles` is triggered. This method will remove the selected files from the data structure and refresh the `NSTableView`. The method retrieves the selected rows of the `NSTableView` - which act as indexes to the data structure items to be removed. To simply remove each of the objects at each index from the `NSMutableArray` would cause problems - this is as the `NSMutableArray` object will automatically resize, rendering the index of the next item removed to be incorrect.

Instead the data structure must be cycled through adding each object at each selected index to a temporary `NSMutableArray`, after this is complete the method `removeObjectsInArray` can be called on the data structure using the temporary `NSMutableArray` as a parameter. This is shown in figure 7.9.

```
//Initialise
//int limit - as number of items to be removed
//unsigned indexBuffer - as array containing the indexes to be removed

//create temporary locations to store in
NSMutableArray *tempArray = [NSMutableArray array];
id tempObject;

//store each object to be removed in a temporary array
for (int idx = 0; idx < limit; idx++) {
    tempObject = [records objectAtIndex:indexBuffer[idx]];
    [tempArray addObject:tempObject];
}

//remove all the items in the temporary array
[records removeObjectsInArray:tempArray];
```

Figure 7.9 - Example code showing the removal of objects from an array

Image preview/information

The image preview drawer makes use of the Cocoa NSDrawer object, this is a simple panel attached to a window that can be toggled on or off. This drawer displays information about the selected image, including a preview of the image (with the determined rotation), the filename, the file kind, the file-path, the determined rotation, and the rotation confidence. This drawer is toggled open and closed with a button on the interface.

To keep updated with the information of the currently selected file the program makes use of a timer, which triggers a refresh method every 0.1 seconds. To save on CPU usage this method checks whether the drawer is open, only refreshing if it is, the method then goes on to set the various pieces of information about the file in the drawer view.

The spaces set for information about the file are as NSTextField objects, when these need to be updated the required information is passed as a string parameter with the method `setStringValue` called on the relevant NSTextField object.

To update the preview of the image it is fetched from disk - using the file path stored in the data structure - and read in as an NSImage object. The NSImage object is then transformed with the determined rotation²⁹, also stored in the data structure, and set in an NSImageView object using the `setImage` method. The full code for transforming the NSImage object can be found in appendix C, section 14.3.1.

Processing of the images

When the user hits the 'Process' button the `processList` method is triggered in the interface controller. The interface controller handles the initialisation of processing the images, retrieving all the variables needed, and processing each image in turn using the back-end image processing code.

In performing the processing the following steps are taken:

1. The data structure is checked for the existence of files to process - this will throw an error if no files are in the list;
2. The controls of the interface are disabled;
3. The variables are fetched from the interface controls - these are also validated;
4. The list of file paths is fetched from the data structure;
5. Each file is then processed - this is done in conjunction with the image processing back-end code, details of this interconnection can be found in section 7.3;
6. The list of files is refreshed from the data structure;

7. The controls of the interface are enabled again.

An alert panel is displayed if any of the processed files are returned with a low confidence of rotation, this is when the confidence is below a set global variable.

7.2 Image Processing

This section covers the code implementation of the image processing C++ class - `Image.cpp` - which is instantiated with the file-path of an image. The class contains several private and public methods for processing the image at the file-path specified, to determine its correct orientation.

This section of the program uses the *CImg C++ library*³⁰ for many of the image processing methods. Full and commented code is given in appendix C, section 14.3.2.

7.2.1 Methods of Analysis

Three methods of image analysis have been implemented for determining the orientation of images, these are implemented using C++ and the image processing methods provided in the *CImg library*. These determination techniques are as follows:

A. Colour Moment analysis with 4-side segmentation

This method segments the image into a user specified number of blocks - $N \times N$ - and using the top, left, right, and bottom regions of the image - defined by these segments - analyses the predominance of blue in each. This method determines that the region with the greatest predominance of blue should be the top of the image.

B. Complexity analysis

This method uses a 9 pixel map, positioning the central pixel over each pixel in the image and determining if the difference in intensity between that pixel and any of the surround 8 is greater than a threshold value.

C. Edge detection analysis

This method is a modified version of the EDH method, using a sobel transform to identify edges within the image. If the orientation of an edge lies within one of two ranges - one for horizontal lines, one for vertical - then a count is incremented pertaining to that orientation. The size of these ranges is a user determined variable.

7.2.2 The Main Method

The main method - `processImage` - is the controlling method of the class. From this method each analysis method is called, the decision merging method is called, and the re-orientated image is saved - if this has been specified.

The file at the given file-path is read using the `CImg` construct - `CImg<unsigned char> image(filePath)` - the number of colour channels is then retrieved to determine whether the image is grayscale or colour, this uses the `dimv()` method called on the image.

If the image is found to be colour a call is made to the colour determination method and the image is converted to grayscale - this is done using the `toGrayscale(image)` method, taking the colour image as a parameter and returning a grayscale version. If the image is already grayscale then error codes are stored for the colour determination results - to indicate this analysis method was not conducted - and the program flow continues.

The complexity determination and edge determination methods are then called using the grayscale image - whether it is the original image or a converted version.

A call is then made to calculate the final rotation and confidence, this is the decision merging method.

The rotated image may be saved at this point depending on the users preference.

7.2.3 Other Methods

Convert to grayscale

This method takes a colour image as a parameter and cycles over each pixel in the image - using a library function `cimg_mapXY()` - storing the grayscale converted pixel value of that location - using the equation given in figure 7.10 - in a temporary image. The temporary image is then returned.

$$\text{grayscale} = (0.3 \times \text{red}) + (0.59 \times \text{green}) + (0.11 \times \text{blue})$$

Figure 7.10 - The equation for conversion from RGB to grayscale

Fetch rotation/confidence

These are two methods which simply return the stored final rotation and confidence.

7.2.4 Colour Determination

The colour determination algorithm examines the predominant colour of each pixel in the top, right, left, and bottom regions of the image, the size of these regions being a user set variable - in percentage of the image dimensions. For each colour (red, green, and blue) in each region a count is made, this is incremented every time a pixel of that predominant colour is found in that region. Once the image has been examined, the colour count for each region is analysed and, based on the amount of blue in that region, a confidence that the region is the top of the image is stored. Whichever region has the greatest confidence assigned to it is assumed to be the top of the image.

This process uses two image loops to loop over the top and bottom portions of the image, and the left and right portions. These image loops are the CImg construct `cimg_for_borderXY(img,x,y,n)` which loop over the image's sides (or borders) - the size of which are a parameter `n` - in the X and Y direction. Two loops are used, this is as the size of the border at the top and bottom, as opposed to the right and left, will be different if the image is not square.

At each pixel location the amount of red, green, and blue, is obtained, and using a series of conditional statements the predominant colour is determined. A count is then incremented for that colour in that region.

A confidence determination is then run - this segment of code examines the count of blue pixels against that of red and green in each region. If the blue count is greater than both the red and green counts, then a confidence is calculated that this region is the top of the image. This confidence is calculated using the formulae in figure 7.11.

$$\text{confidence} = 100 \times \left(1 - \left(\frac{\text{red count} + \text{green count}}{2 \times \text{blue count}} \right) \right)$$

Figure 7.11 - The formulae used to calculate the confidence with which the edge is the top of the image

A series of conditional statements is then used to evaluate the region with the greatest confidence, deciding the rotation returned by the colour determination - e.g. if the region with the greatest confidence is on the right then the image must be rotated by 270°CW. This amount of rotation is stored in a global variable, along with the confidence relating to this rotation.

7.2.5 Complexity Determination

The complexity determination algorithm examines each pixel within the image and measures the difference between that pixel's intensity and its neighbours, if the difference is above a set threshold then that pixel is determined to be a location of complexity. The count of locations of complexity in the top, right, left, and bottom halves of the image is made. Based on these counts a judgement is made as to the orientation of the image.

To loop over the image and examine each pixel's neighbours a special loop from the *CImg* library is used along with another *CImg* construct called a map. This allows the image to be looped over with the ability to reference each pixel's neighbours. This is shown in figure 7.12 and figure 7.13.

```
//define a 3x3 pixel map
CImg_3x3(I,float);

//loop over the image - img - using the map - I
cimg_map3x3(img,x,y,z,v,I) {
    //perform some image processing on the current pixel
    //reference the neighbouring pixels using the map I
}
```

Figure 7.12 - Example code showing the use of a *CImg* map

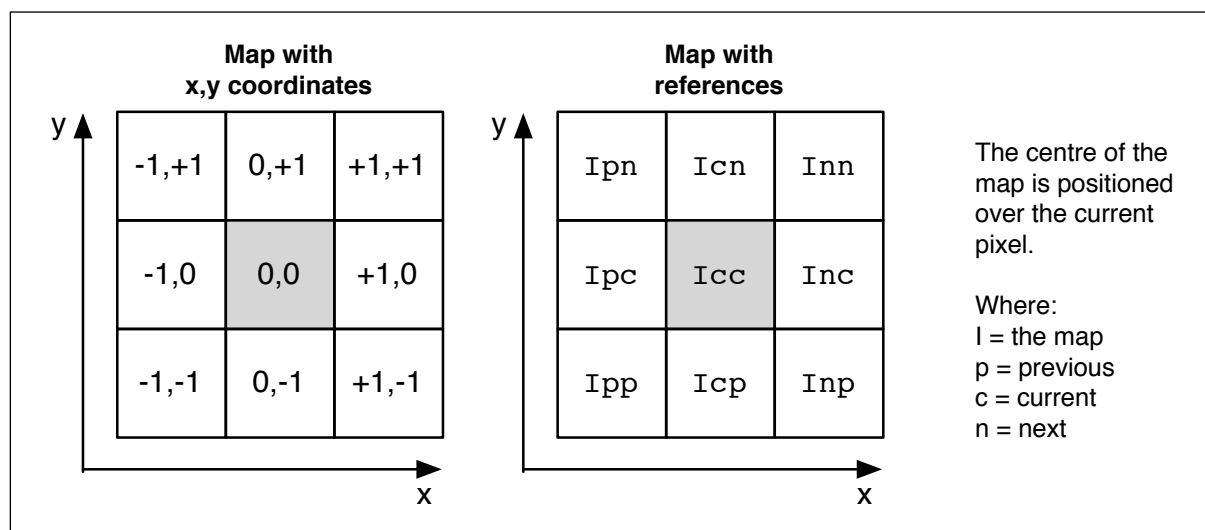


Figure 7.13 - The use of maps within *CImg*

The image is looped over obtaining the difference between the current pixel and each of its neighbours, the magnitude of the difference is obtained using the C++ Math

function $\text{fabs}(x)$. If the magnitude of any of these differences is greater than the set threshold value, then the count for the current half of the image is incremented.

After running this loop a count of the complex locations in each half of the image is obtained. Conditional statements are then used to examine which count is the greatest, this half of the image is assumed to be the lower half, and a rotation based on this is stored. A confidence of this rotation is also calculated using the formulae shown in figure 7.14.

$$\text{confidence} = 100 \times \left(1 - \left(\frac{\text{bottom count}}{\text{top count}} \right) \right)$$

Figure 7.14 - The formulae used to calculate the confidence of a complexity rotation determination

The determined rotation and complexity are both stored in global variables.

7.2.6 Edge Determination

The edge determination algorithm loops over all the pixels in the image, and using a convolution matrix calculates the direction of an edge at each pixel location. These edges are counted into two bins, if the direction is within a certain range - one for vertical edges, and one for horizontal edges. Based on the amount of horizontal edges versus the amount of vertical edges a judgement is made on the image orientation.

This process uses an image loop with a 3x3 map to reference each pixel's neighbours, as shown in figure 7.12 and figure 7.13.

Each pixel of the image is examined, and using the image map to reference the surrounding pixels two variables - G_x and G_y - are calculated. From these a value of theta (θ) obtained, this process is demonstrated in section 4.1.3. Theta is a measurement of the orientation of the edge at that pixel location in radians, which is then examined for being within two ranges - for vertical and horizontal edges. The spread of these ranges being a user set variable, if the result is within one of these ranges a count is incremented. This is shown in figure 7.15.

```

//create a 3x3 map
CImg_3x3(I,float);

cimg_map3x3(img,x,y,0,0,I) {
    //calculate Gx and Gy using map references
    float gx = Inp + Inc*2 + Inn - Ipp - Ipc*2 - Ipn;
    float gy = Ipp + Icp*2 + Inp - Ipn - Icn*2 - Inn;

    //find the result: theta = arctan(Gy / Gx)
    float result = atan2 (gy,gx);

    //if edge is in vertical range
    //increment vertical edge count

    //if edge is in horizontal range
    //increment horizontal edge count
}

```

Figure 7.15 - Example code showing the implementation of sobel edge detection

A conditional statement is now used to determine if there are more horizontal edges or vertical edges, determining the orientation judgement. The rotation from this judgement can only be determined to be one of two groups - 90° or 270°, or 0° or 180° - as the edges are only orientated in the vertical and horizontal planes with no direction attached to them.

A confidence calculation is also made using the same method as in the complexity determination algorithm, shown in figure 7.14. Both the determined rotation and confidence are stored as global variables.

7.2.7 Decision Merging

Once the determination algorithm methods have been run, the results from each must be merged to give a final decision on the rotation, and the confidence pertaining to that rotation. This process is called decision merging.

The process of decision merging follows the following steps for each determination result:

1. The result is examined for an error code - this is where 101 is stored as a confidence value - indicating that no determination was made by that algorithm;
2. If no error code exists then a series of conditional statements extract the determined rotation;

3. The counter assigned to that rotation amount is incremented;
4. The confidence is added to a bin assigned to that rotation's cumulative confidence.

This varies for the edge determination where the result is between two options - 0° or 180° and 90° and 270° - in this case the counters for each rotation are incremented and the confidence split between the cumulative counters for each of these rotation amounts.

After this process is complete a series of conditional statements pick the largest rotation counter value, which is set as the determined rotation. The set confidence is that of the cumulative confidence counter assigned to that rotation amount, over the rotation counter amount, or the mean confidence of the number of determinations that made that rotation judgement.

If no determination can be made then an error code is stored.

7.3 Program Interconnection

Bringing together the two distinct parts of the application holds a few problems as the image processing algorithms are written in C++ and the interface is built using Cocoa³¹ which natively uses *Objective-C*. This section covers the implementation of the interconnection between the two parts of the application.

To allow for communication between the interface and the C++ image processing the interface is implemented using *Objective-C++*, this a variation of *Objective-C* which allows for use of C++ syntax - and so the use of C++ objects - along with regular *Objective-C* syntax. By implementing the interface controller using *Objective-C++* it can instantiate a C++ object - representing the image to be processed - and call methods on this object. This process is shown in figure 7.16.

```
//create a c++ object
CppClass* myCppClass;
myCppClass = new CCppClass(var1);

//call a method on the C++ object which returns an int
int returned = myCppClass -> main(var2);

//delete the C++ object as it's no longer needed
delete myCppClass;
```

Figure 7.16 - Example code showing the use of C++ objects in *Objective-C++*

Objects in *Objective-C* - such as strings - are not necessarily found in C++, for this reason conversion or simplification was necessary between the two sides. The C++ code deals with strings as arrays of characters, whereas the interface controller code uses the object `NSString`. So conversion between these must occur, this is done using the code shown in figure 7.17.

```
unsigned int strLen = [str length];
char temp[strLen + 1];
strcpy(temp, [str cString]);

//where
//str is the NSString to convert
//and temp is the destination character array
```

Figure 7.17 - Example code showing the conversion between an `NSString` and a character array

All other variables are passed using the primitive `int`, which is common to both *Objective-C* and C++.

Processing each image may take several seconds, and while processing is being carried out the program may hang, this causes a problem for drawing the progress indication bar. To solve this problem the window must be redrawn each time an image is processed, by making a call to the application 'run loop'. Along with this the progress indicator must be forced to run on a separate thread to the application, so it will redraw. The code for this is shown in figure 7.18.

```
//execute this line when initialising the controller
[progressIndicator setUsesThreadedAnimation:YES];

//execute this line for every image processed
[[NSRunLoop currentRunLoop] runMode:NSEventTrackingRunLoopMode
 beforeDate:[NSDate date]];
```

Figure 7.18 - Example code showing the implementation of a progress indicator

8.0 Testing and Improvement

This chapter describes the various stages of testing and the methods of testing performed, improvements were carried out after initial testing, which are also described here.

During implementation testing was carried out in each iteration phase of construction - the process of which is described in chapter 5.0. After completion of a functioning system 'initial testing' was carried out - this being the final phase of iteration in the construction of the system. After carrying out improvements from this stage of testing final testing was carried out, this is to analyse the effectiveness of the system in meeting user requirements.

8.1 Initial Testing

This phase of testing was conducted on two levels. To test the usability of the system a usability survey was conducted on a small group of potential users. To test the operation of the image processing a 'white-box' testing approach³² was taken.

8.1.1 Usability Testing

In usability testing users were given tasks to perform, designed to test the interaction scenarios of the system set out in section 6.3. Usability testing is often referred to as 'black-box' testing³³ - where the user (or tester) doesn't know the details of the systems operation - so giving a more 'real world' test of the systems performance. Through this testing the usability of the interface, and how well it meets user requirements, can be analysed, and potential failures in the systems integrity may be uncovered.

Usability surveys were conducted on a small group of users with a range of experience and computer literacy levels. Users were asked to perform six basic tasks, and while conducting these were monitored and directed when necessary. A full copy of the results of these user surveys can be found in appendix A, section 14.1.1.

The following findings were made:

- A. There was ambiguity about the process of adding and removing images to be processed. Users commented that the menu option to open images was titled "Add..." where as the survey referred to this operation as "opening images". The shortcut for this operation is also "command-O" indicating that the operation should be "Open...". Users also encountered problems with the operation of the buttons to open and remove images, it was commented that the function of buttons titled "+" and "-" was not immediately obvious, and it was suggested they should be titled "open..." and "remove".

- B. Users commented that using a tick box to indicate saving images on processing was unintuitive as it was not clear that ticking a box titled “Save” meant that images would be saved with the determined rotation.
- C. A user commented that the error alert panels were not informative enough, in particular that no indication was given of how to avoid such an error in the future.
- D. Several users indicated that they were not aware of the functionality of the variables, referring to the image processing variables.
- E. A user commented that the information displayed about an image did not indicate whether that image was rotated or not.
- F. A user made the comment that once processing has been initiated it could not be stopped without aborting the program.
- G. One user discovered that the system would crash if, after a directory had been specified to save to, it was deleted, and then the images processed.
- H. Another user complained that the system appeared to become slow and ‘jerky’ when previewing images using the info drawer.

8.1.2 White-box Testing

During ‘white-box’³⁴ testing various images were used as test subjects to specifically test the systems operation. This is in line with the theory of ‘white-box’ testing, which is to deliberately try to exploit possible system vulnerabilities, with knowledge of the systems operation.

Some images were created to test specific aspects of the operation of the image processing algorithms, and the decision merging process. Others were used to test the operation of the interface, and how this dealt with errors. The images created for testing are shown in figure 8.1.

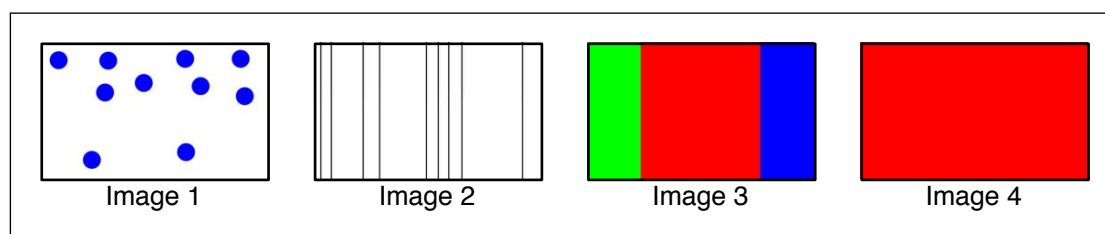


Figure 8.1 - The test images used in white-box testing

Image 1 was created to test the complexity detection algorithm, image 2 to test the edge detection algorithm, image 3 to test the colour detection algorithm, and image 4 to test the case where no determination can be made.

The following findings were made from testing with these specifically created images:

- I. On processing image 4 the system displayed a rotation of 1° and a confidence of over -300,000, without any user feedback being given.
- J. When previewing image 3 - determined with a rotation of 270° - it was noticed that the info drawer containing the image preview became very slow, and after some time unresponsive. This was also noted in the usability survey (point H).
- K. It was noticed during testing that some images returned a rotation of 180° , this degree of rotation should be excluded from decisions as it was determined that images are not expected to be out of orientation by 180° .

8.2 Improvements from Initial Testing

After conducting initial testing several problems were remedied, the process of these improvements, relating to each problem, is described below:

- A. The interface design was altered to consistently use the word “open” when referring to opening images to be processed, the buttons on the interface were also changed to read “open...” and “remove”.
- B. The text of the save tick box was changed to read “Save on process”, this should make it more obvious as to its purpose.
- C. All alert panels were refined to give more detail on the error encountered.
- D. The titles of the variables container was changed to “processing variables”, making their purpose clearer. As the interface is not developed for complete user release but more of a research platform this is not an issue, but is however an interesting usability point.
- E. The info drawer was updated to indicate if an image has not been processed, this should also indicate to the user that the image has not been rotated.
- F. This user-raised point has not been remedied, as it is felt that it is beyond the scope of the project, as the interface is designed for research purposes.
- G. Checks have been inserted to ensure that, before running processing, any save locations are valid paths, this is done using the code shown in figure 8.2.

```
//convert NSString to NSURL for checking
NSURL *url = [NSURL URLWithString:filePath];

//check to be valid path
BOOL isValid = [url isFileURL];

//if it's not valid display error message and exit method
if(!isValid) {
    NSRunAlertPanel(@"Error",@"An error message.",@"OK",NULL,NULL);
    return;
}
```

Figure 8.2 - Example code showing the implementation of file path checking

- H. The problem in which the system became slow when previewing images has been remedied by implementing a caching system for image previews. On opening an image it is stored that there is no valid preview for that image. When the info drawer is refreshed a check is made for a valid preview for the selected image, if it is found that there is none then the image is loaded and stored with the determined rotation, it is also set that there is a valid preview. When the image list is processed all the images are set to have an invalid preview, thus forcing a preview refresh when the info drawer is shown. By this method the image is not rotated - a time consuming task - every time the info drawer is refreshed (set to happen every 0.1 seconds), this conserves memory and removes the 'jerkiness' of the info drawer.
- I. When this problem is encountered the system is expected to give user feedback indicating that the specified image's orientation could not be determined. From examining the code using a debugger, it became clear that if no determination could be made then no confidence value was stored, therefore these variables were not initialised. Both rotation and confidence are stored as an `int`, the un-initialised value of which is over -300,000. This un-initialised value was being returned for the confidence. The code in figure 8.3 was inserted to account for no determination being made - storing error codes in this event - removing the problem.


```

//other determination code has run
else {
    //set error codes
    rotation = 361;
    confidence = 101;

    //indicate no determination was made
    return 0;
}

```

Figure 8.3 - Example code showing the implementation of storing error codes

- J. The solution for this problem is described in point H.
- K. This problem was solved by inserting a check for a rotation of 180° and setting this to be 0° if found. The justification for this decision can be found in section 3.2.

8.3 Application Release

The following are screen-shots of the functioning system.

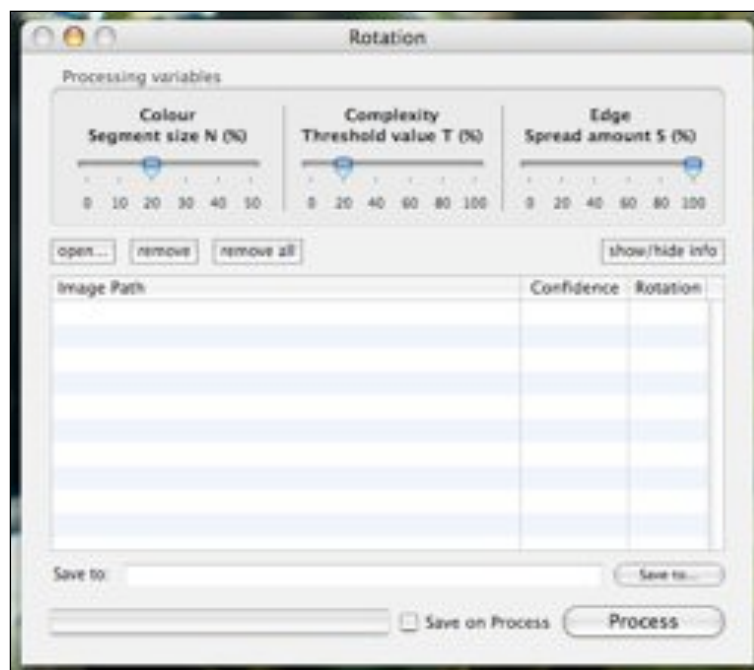


Figure 8.4 - The application interface

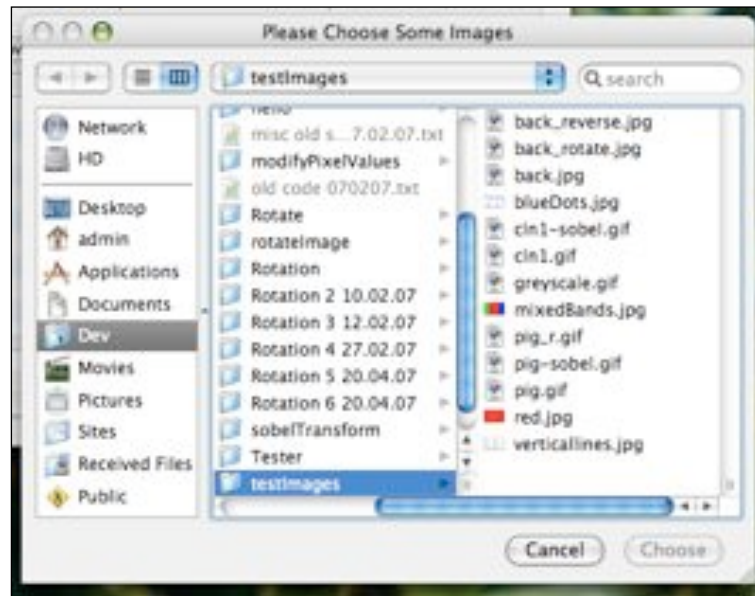


Figure 8.5 - The file-picker to choose images to open

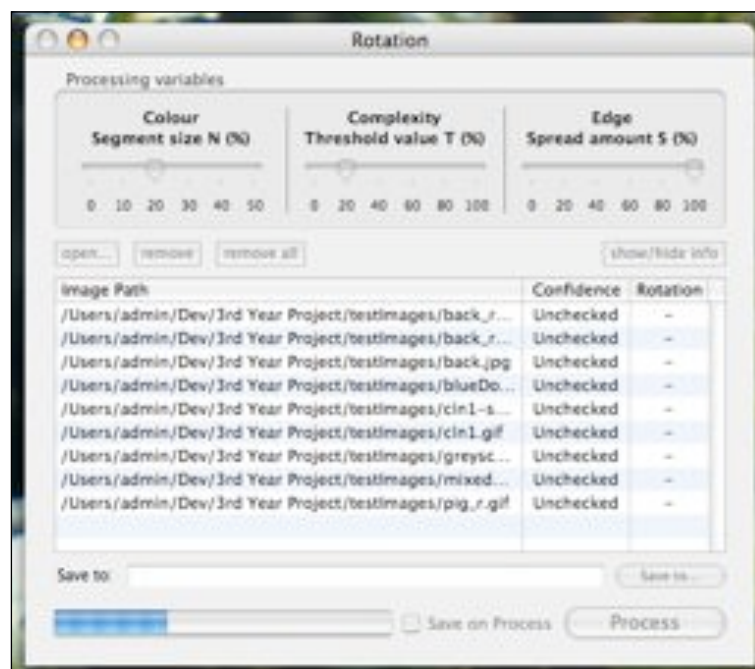


Figure 8.6 - The application during processing

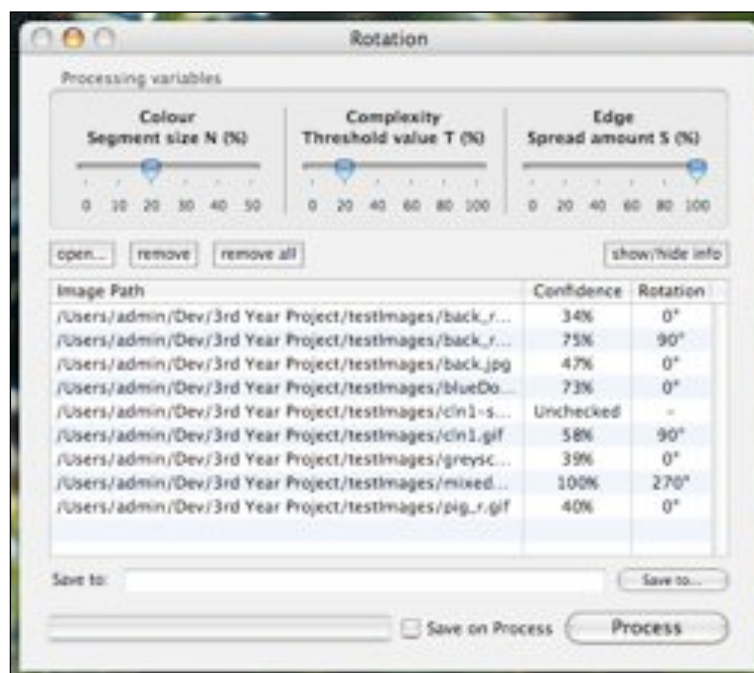


Figure 8.7 - The interface showing processed images



Figure 8.8 - The info drawer showing image preview and information

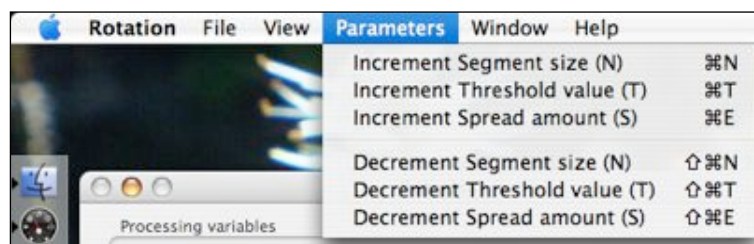


Figure 8.9 - The application menu showing indication of keyboard shortcuts

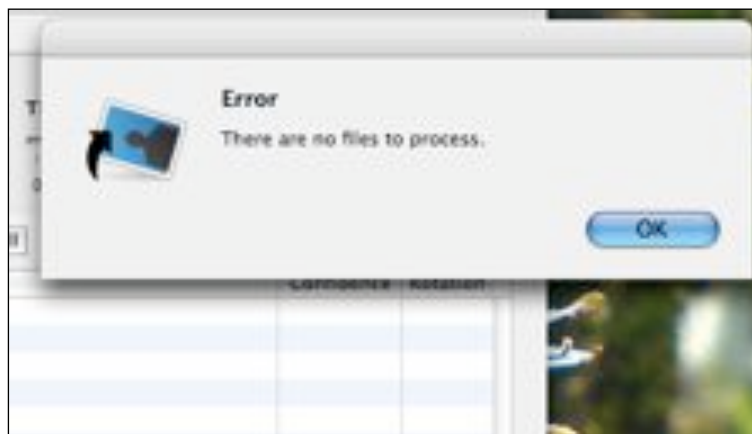


Figure 8.10 - The warning displayed when processing is started with no files having been opened

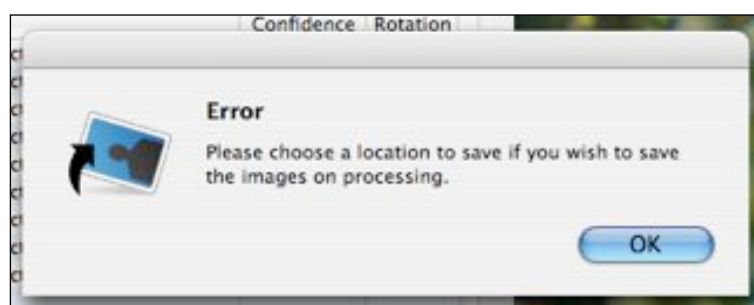


Figure 8.11 - The warning displayed when processing is started with saving specified but no save location has been specified

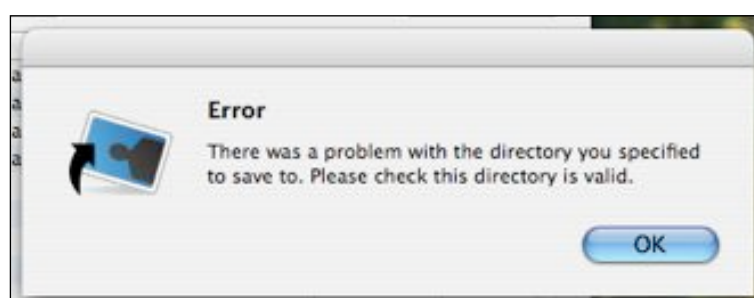


Figure 8.12 - The warning displayed when the directory specified for saving to is not valid



Figure 8.13 - The warning displayed when an error is encountered during processing



Figure 8.14 - The warning displayed when some images have a low confidence level

9.0 Final Testing

A system must be tested as to its performance, and how well it meets its user requirements. This testing will allow assessment of how well the project meets its goals.

9.1 User Requirements Testing

The system was tested against the interaction scenarios set out in chapter 6.0. The tests and the results of these are given in table 9.1 along with references to screen captures of each result found in section 8.3.

Test	Result	Figure
Open several images	<p>After clicking the “open...” button, the system displayed a file-picker, where only image files could be chosen. Using the keyboard shortcut or menu item had the same result.</p> <p>On clicking “OK” in the file-picker, the selected files were listed on the file list, on clicking “Cancel” the file-picker closed and no change to the interface was made.</p>	8.5
Remove some images	<p>On clicking the “remove” button, the selected files were removed from the list. When no files were selected an alert panel was displayed indicating this.</p> <p>The same result was obtained using the keyboard shortcut and menu item.</p>	-
Remove all images	<p>On clicking the “remove all” button, all files were removed from the list.</p> <p>The same result was obtained using the keyboard shortcut and menu item.</p>	-
Process images	<p>On clicking the “Process” button, the images in the list were processed, this took a little time. While processing was taking place the buttons on the interface were disabled and the progress indicator incremented for each image.</p> <p>Images with low confidence were warned of with an alert panel. Images that could not be processed were alerted to with an alert panel describing the error.</p> <p>If no files were in the list an alert panel was displayed detailing this.</p> <p>The same result was obtained using the keyboard shortcut and menu item.</p>	8.6, 8.7, 8.10, 8.13, 8.14

Test	Result	Figure
Process and save the images without specifying a save location	After ticking the “Save on Process” check-box, and clicking the “Process” button, the system displayed an alert panel detailing that there was no save location specified and to specify one before processing and saving.	8.11
Specify a save location	After clicking the “Save to...” button, the system displayed a file-picker where only directories could be selected. On clicking “OK” the directory path was added to the “Save to” text field, on clicking “Cancel” the file picker closed and no action was taken. The same result was obtained using the keyboard shortcut and menu item.	-
Process and save the images with a valid save location	After clicking the “Process” button, with a valid save location selected, and “Save on Process” ticked, the system processed the images - with the same results as the <i>Process images</i> test - and the saved images were verified as being in the correct location with the determined orientation.	8.7
Process and save the images after removing the save location from disk	On clicking the “Process” button, the system displayed an alert panel warning that the save location was not valid, processing did not continue and the interface was enabled.	8.12
Open an image, remove it from disk, then process	On clicking the “Process” button, the system performed the processing until reaching the missing image, the program then crashed. Before this happened an exception was thrown in C++ (this was found through debugging).	-
Increment and decrement the variables	On moving the variable sliders the interface updated, the same result was achieved using the keyboard shortcuts and menu items. The slider values did not overflow beyond maximum or minimum. On processing, the current slider values were taken as processing variables.	8.9

Test	Result	Figure
Display the 'info drawer' with no files selected, one file selected, and multiple files selected	<p>The info drawer was opened and closed using the "show/hide info" button, and when pressed in quick succession no ill effects were found. The same result was achieved using the keyboard shortcut and menu item.</p> <p>When no files were selected the info drawer displayed "No Items Selected".</p> <p>When one file was selected the info drawer displayed the following:</p> <ul style="list-style-type: none"> • The file name (including extension); • A preview of the image with determined rotation; • The rotation amount; • The confidence; • The file type (JPEG, GIF, PNG); • The file path. <p>These details were refreshed when another file was selected.</p> <p>When multiple items were selected the info drawer displayed "Multiple Items Selected".</p>	8.8

Table 9.1 - The results of user requirements testing

9.2 System Performance

The performance of a system is important when part of it's functionality is time consuming, such as image processing. The performance of the system is outlined here through tests. During testing no other applications were running, ensuring more accurate results.

9.2.1 Image Processing Time

The system was tested with a range of sizes of both colour and grayscale images, the processing time of these is shown in table 9.2.

Image size	Average Colour Processing Time	Average Grayscale Processing Time
200x150px 128KB colour 112KB grayscale	0.40ms	0.34ms
800x600px 404KB colour 292KB grayscale	2.0ms	1.9ms

Image size	Average Colour Processing Time	Average Grayscale Processing Time
1600x1200px 1125KB colour 814KB grayscale	3.8ms	3.7ms
2800x2100px 2892KB colour 2127KB grayscale	5.5ms	5.4ms

Table 9.2 - The results of image processing time tests

These tests were conducted using timing components within the *Objective-C++* code. The results are an average of 10 measurements, all results are to 2 significant figures, and published in full in appendix A, section 14.1.2.

These results clearly show that larger images take longer to process, but also that colour images take longer to process than grayscale images. The processing times are not excessive for the purposes of research, however if developing an application for retail these times would ideally be reduced.

9.2.2 CPU and Memory Usage

The systems use of CPU and memory was tested during processing, previewing an image, and idle running. These measurements were taken with *Activity Monitor*, the average of 4 readings is given in table 9.3.

Task	Average CPU Usage	Average Memory Usage
Idle	0.1%	7.24MB
Previewing image	7.2%	13.6MB
Processing	35%	17.9MB

Table 9.3 - The results of CPU and memory usage tests

CPU usage is given to 2 significant figures, whereas memory usage is given to 3 significant figures. These results are published in full in appendix A, sections 14.1.3 and 14.1.4.

The system can be seen to perform well when idle, consuming limited system resources. When processing more system resources are used, although due to the nature of the application the time spent processing will be limited. When previewing and image the application uses acceptable amounts of system resources.

10.0 Experiments

Experiments were run to determine the ideal parameters with which to process images, finding their correct orientation. Firstly the performance of the individual determination algorithms was tested, then using the results from these tests the ideal variables were derived and used to test the collaborative determination.

In individual testing two groups of images were used as test subjects - a group of indoor images, and a group of outdoor images mainly of natural landscapes. Each image group contained 10 colour images, and 5 grayscale images. The image groups were of mixed orientations, some images were correctly orientated while others were incorrectly orientated by 90° or 270°.

Each test subject in each group was processed, taking the reading of rotation and confidence for each variable individually. Variables were incremented from minimum to maximum in 10 steps, giving 11 readings per test-subject, per variable, for both rotation and confidence. The full results of these experiments can be found in appendix B, section 14.2.

10.1 Rotation Performance

The determined rotation for each reading was compared to the correct rotation amount for that image, counts were made per variable increment of the number of correct orientations, and the number of incorrect orientations. Where a rotation could not be determined - given by an error code - this was counted as a rejected image.

The following are graphs of performance of the individual determination algorithms for rotation, they are split into indoor and outdoor.

10.1.1 Indoor

The performance of individual determination methods for rotation on indoor images.

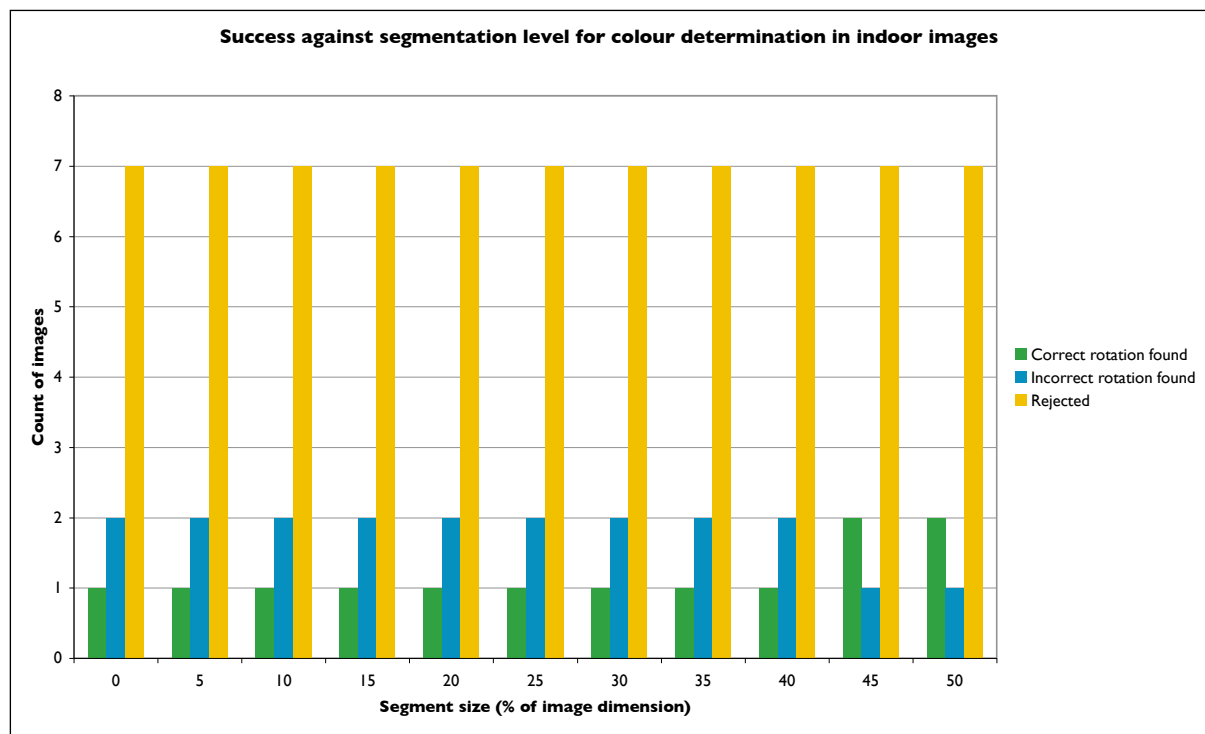


Figure 10.1 - A graph showing the success rate of increasing segmentation size for colour determination on indoor images

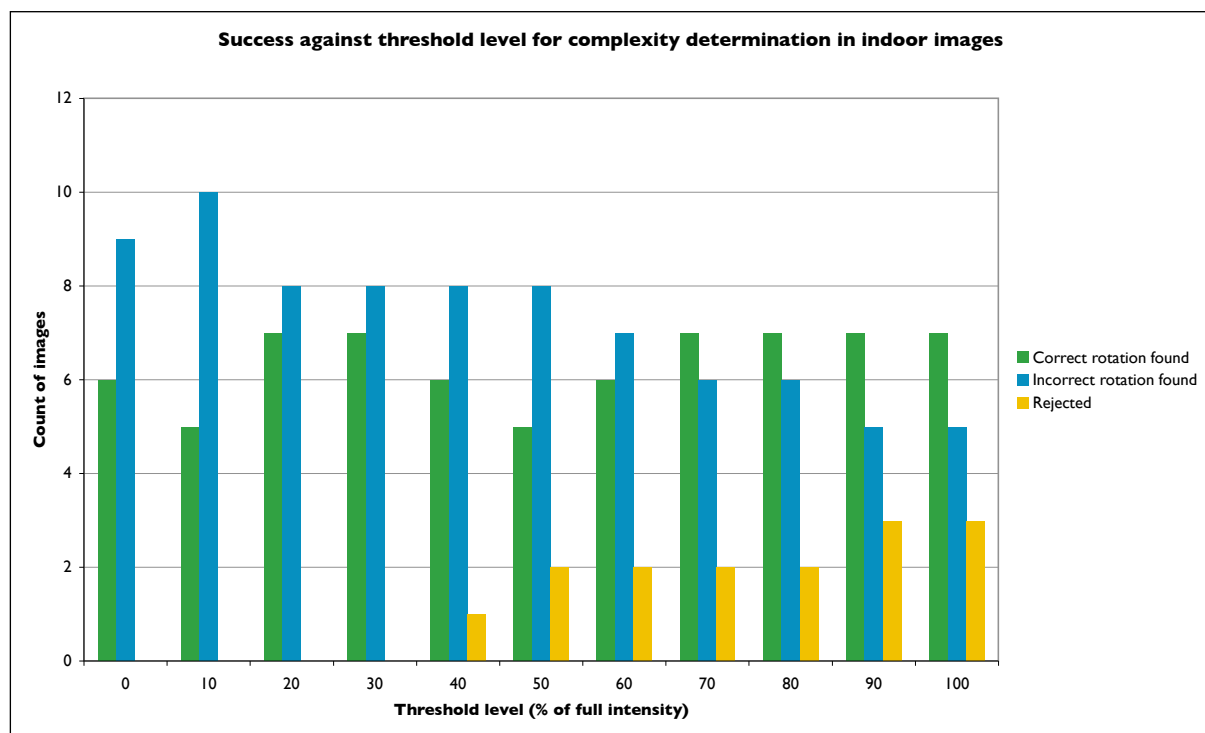


Figure 10.2 - A graph showing the success rate of increasing threshold level for complexity determination on indoor images

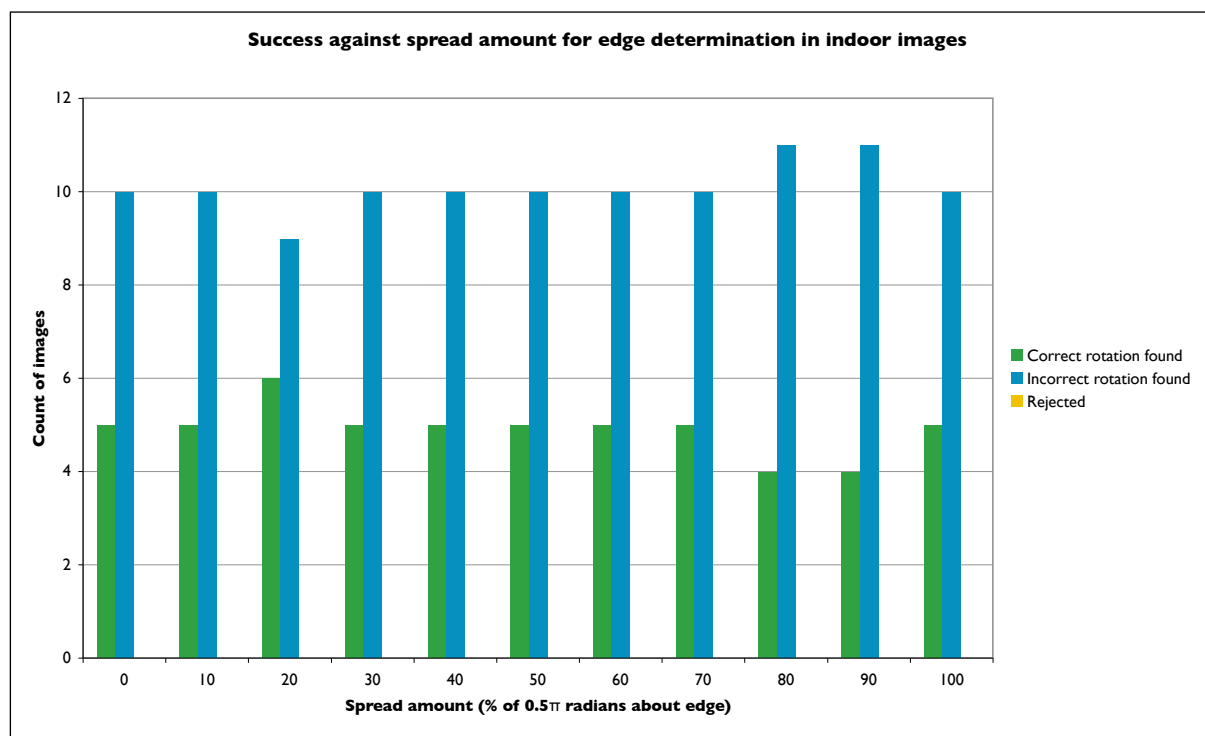


Figure 10.3 - A graph showing the success rate of increasing spread amount for edge determination on indoor images

These results indicate the following:

- That for indoor images colour determination is not very successful;
- That for complexity determination of indoor images while a higher threshold level produces more correct orientations it also produces more rejected images, comparably a threshold level of 20-30% is more suitable - producing no rejections and a slightly higher correct to incorrect ratio;
- That the majority of determinations for edge detection of indoor images are incorrect.

It can be inferred from these results that correct determination for indoor images will be unlikely, but that using a complexity threshold level of 20-30% is desirable.

10.1.2 Outdoor

The performance of individual determination methods for rotation on outdoor images.

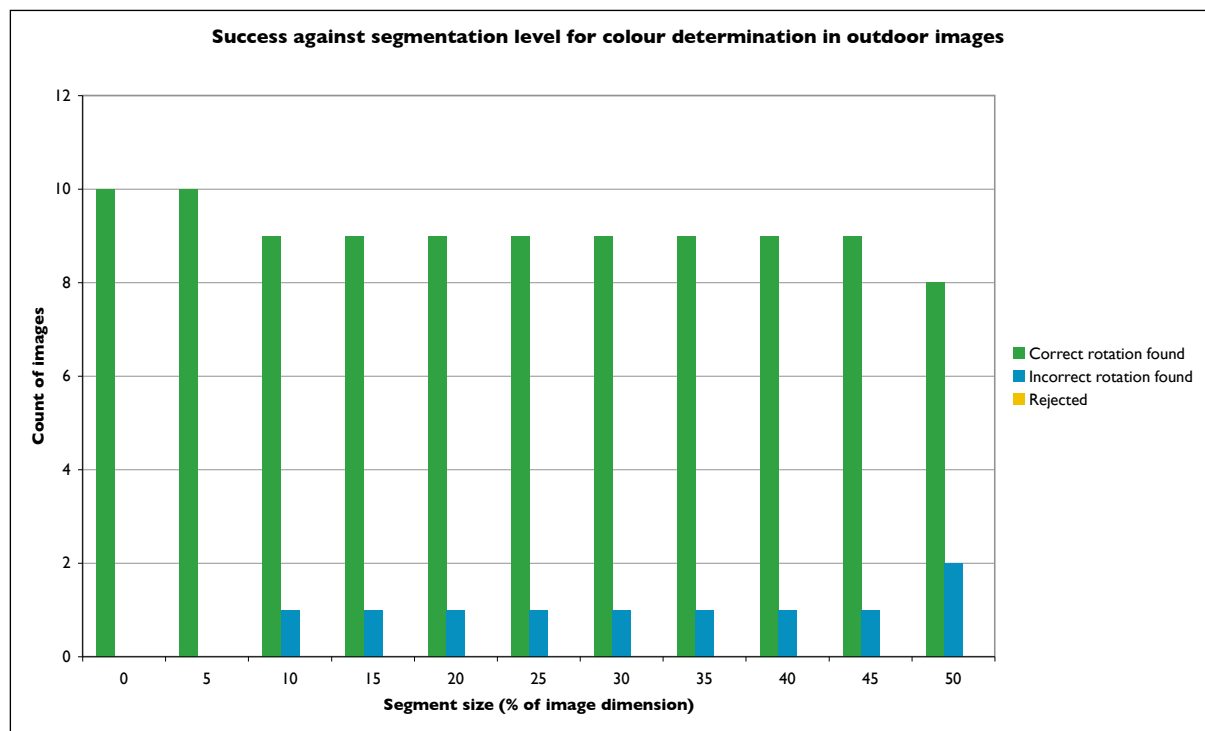


Figure 10.4 - A graph showing the success rate of increasing segmentation size for colour determination on outdoor images

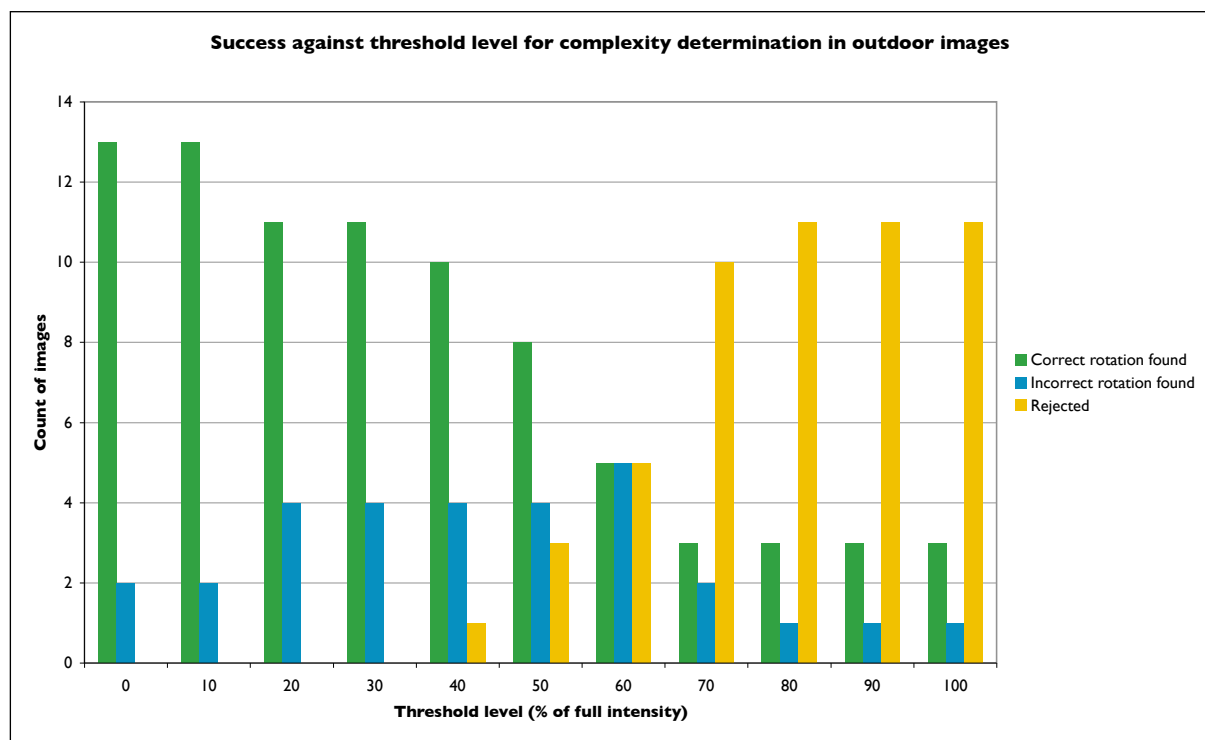


Figure 10.5 - A graph showing the success rate of increasing threshold level for complexity determination on outdoor images

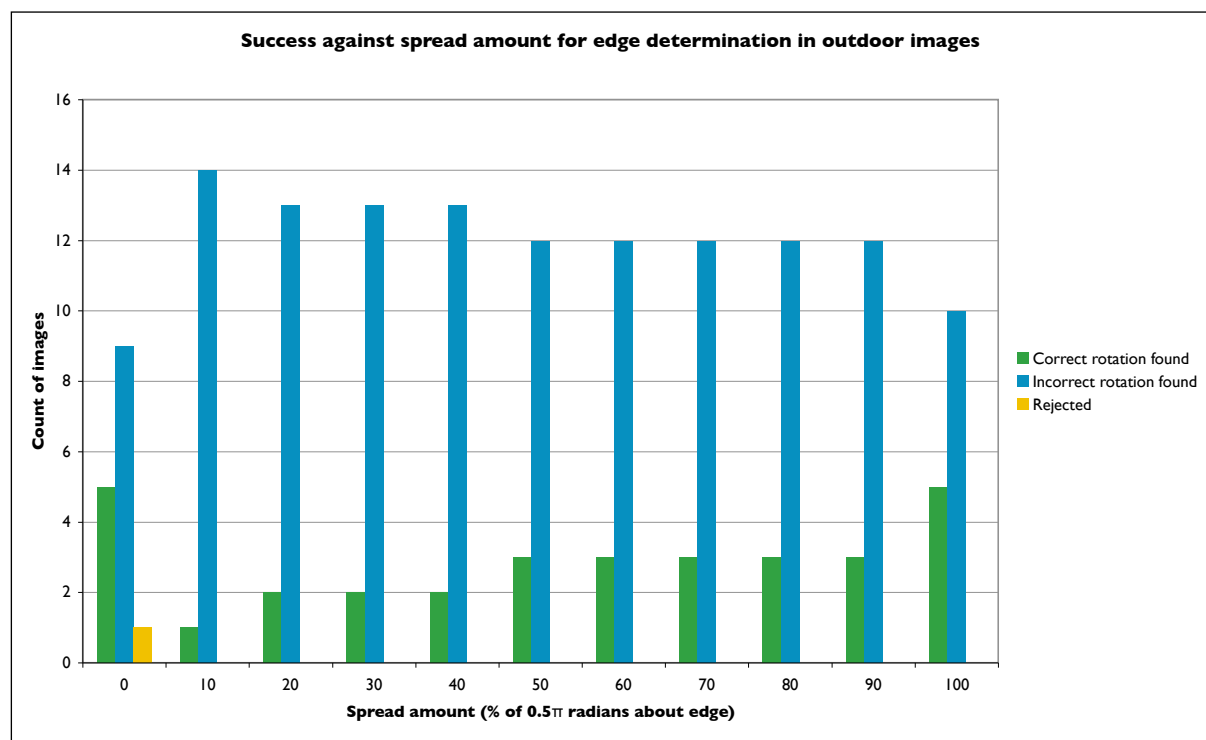


Figure 10.6 - A graph showing the success rate of increasing spread amount for edge determination on indoor images

These results indicate the following:

- That for outdoor images using colour determination a low segment size is desirable;
- That edge determination is not successful for outdoor images, although a spread amount of 100% seems to be slightly more desirable;
- That in complexity determination a low threshold level of 10-20% is desirable, as this produces no rejections and higher correct orientations than incorrect.

Therefore it can be inferred that for outdoor images both complexity and colour determination will be relatively successful - at levels of 5-10% and 10-20% respectively - whereas edge determination will be relatively unsuccessful.

10.2 Confidence Performance

The confidence generated for the rotation for each image at each increment of each variable was also measured, the average of the readings at each increment level was taken for both indoor and outdoor images. Where a determination could not be made the result was not counted in the average.

The graphs of these average confidence levels are given below.

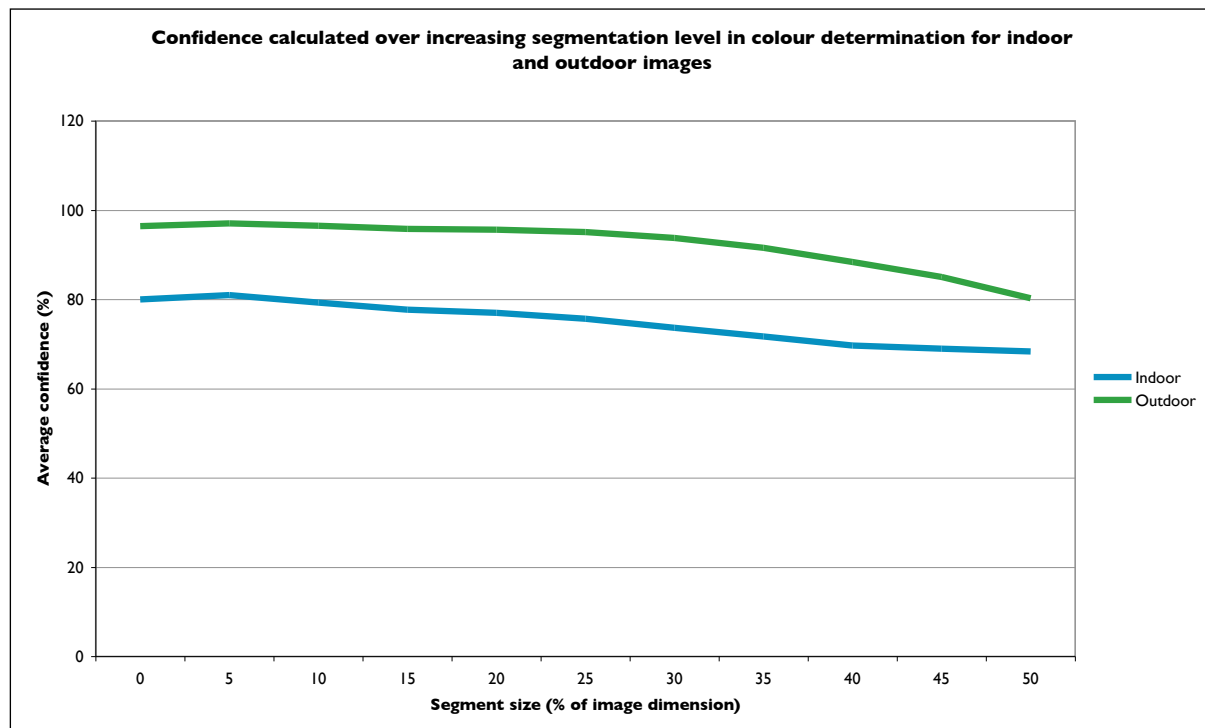


Figure 10.7 - A graph showing average confidence of increasing segmentation size for colour determination on indoor and outdoor images

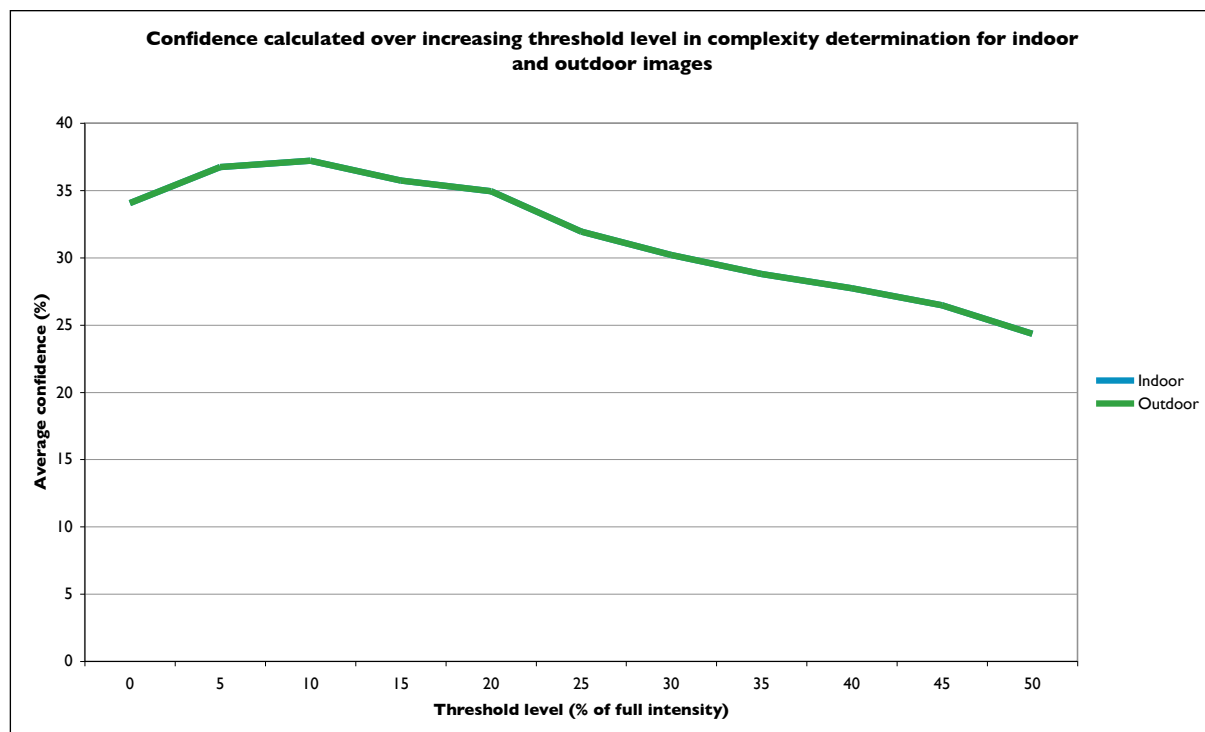


Figure 10.8 - A graph showing average confidence of increasing threshold level for complexity determination on indoor and outdoor images

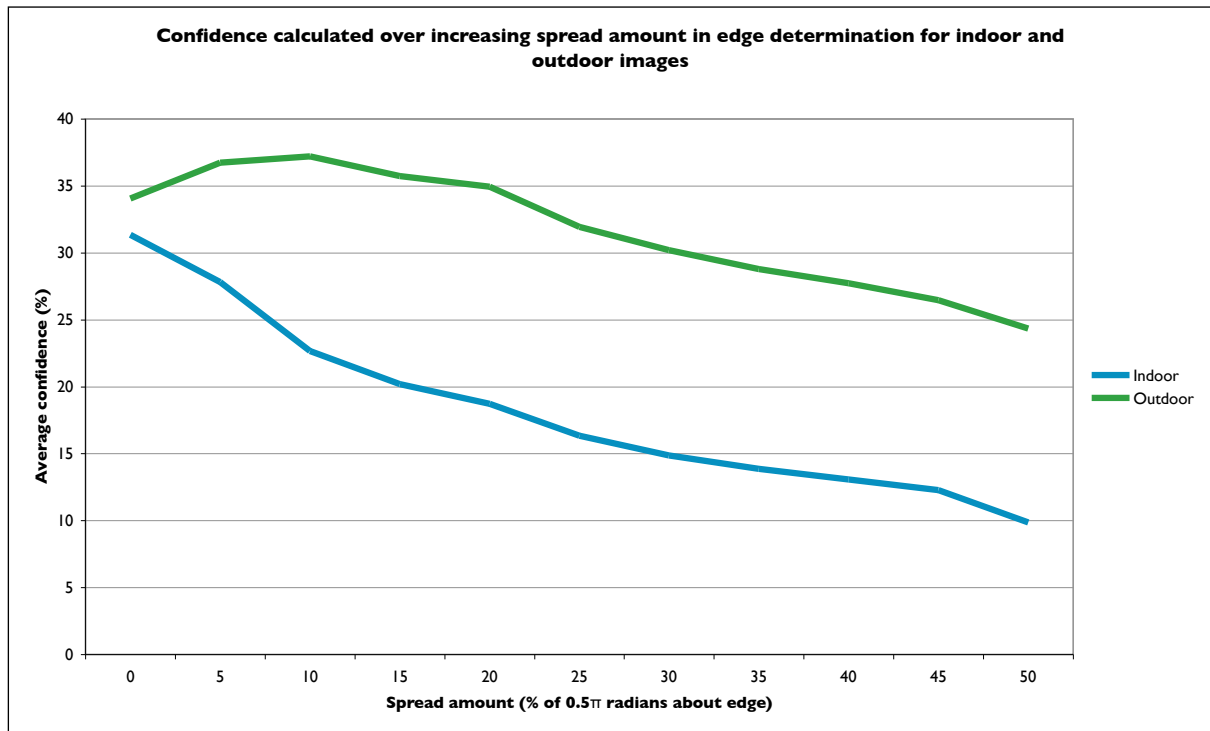


Figure 10.9 - A graph showing average confidence of increasing spread amount for edge determination on indoor and outdoor images

These results show that in both edge determination and complexity determination a higher variable reduces confidence, whereas in colour determination confidence remains relatively steady over all variable levels. It can therefore be inferred that smaller variable values are more likely to be accurate, however this does raise doubts of the relevance of confidence measurements.

10.3 Collaborative Performance

From the results of experiments with each determination algorithm the following parameters have been determined the most suitable:

- Colour determination - segment size 5%
- Complexity determination - threshold level 20%
- Edge determination - edge spread amount 100%

Using these variables a test group of 50 images was processed, 25 of which were indoor images, 25 outdoor, the results of which are shown in the figure 10.10.

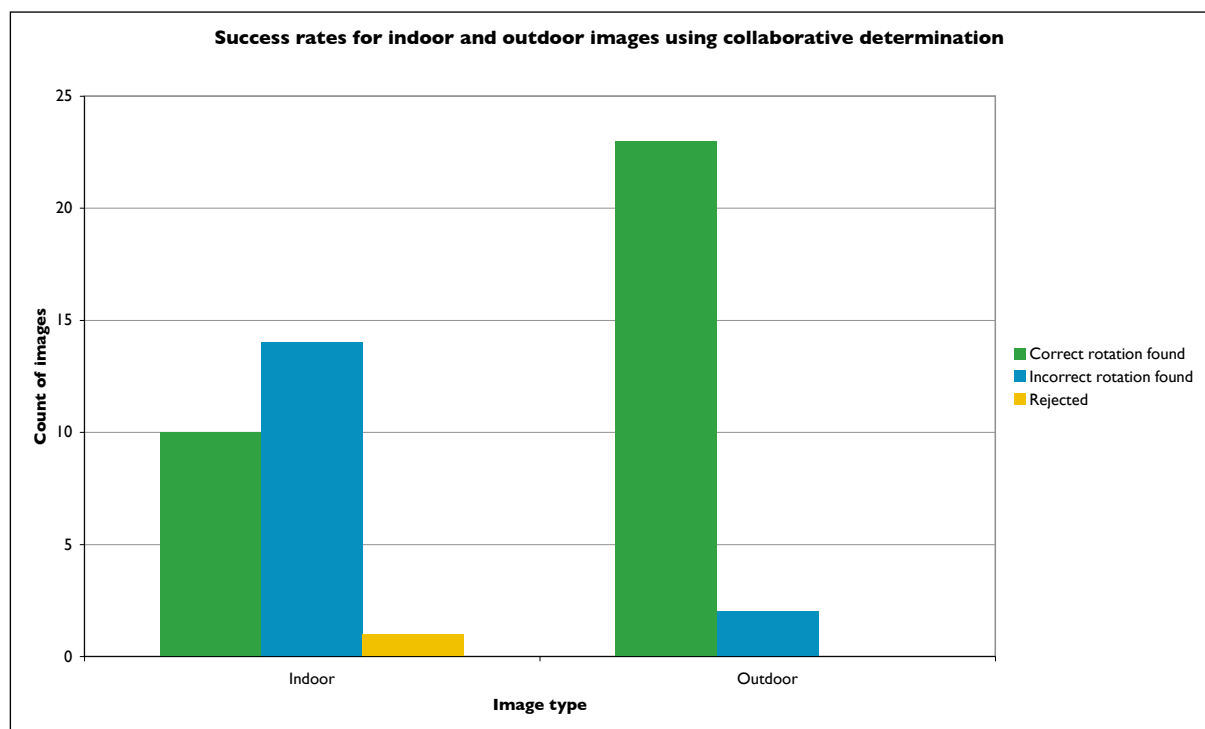


Figure 10.10 - A graph showing success rates for collaborative determination of both indoor and outdoor images

These results show a high success for outdoor images, at a rate of 92%, but a low success rate for indoor images, of only 40%. Only one image was rejected over both image types demonstrating the ability of the algorithms to produce a result in most cases.

These results corroborate those of the results of individual determination tests - that correct indoor image orientation determination is unlikely, but that correct outdoor image orientation determination is much more likely.

Figure 10.11 shows some of the incorrectly determined images, along with their correct orientation.

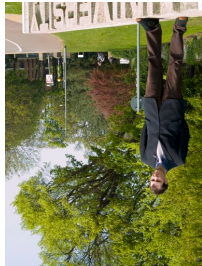



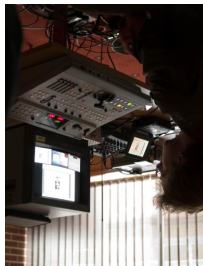






Determined orientation	Correct orientation		Determined orientation	Correct orientation
 90°	 270°		 0°	 270°
 90°	 270°		 Rejected	
 0°	 270°		 0°	 90°

Figure 10.11 - Showing some of the incorrectly determined images and their correct orientation from collaborative determination

In image 1.1 there is a lack of blue shades towards the top of the image, there is also more complexity at the top of the image, these factors would cause misleading

results from the colour determination and complexity determination methods, giving a result of 90° rather than 270° . As image 1.2 is grayscale colour determination is not possible, the complexity level is also relatively uniform over the image, therefore edge detection would be the strongest result, and as there are more horizontal edges the image it was orientated as 0° .

In image 2.1 there is a lot of complexity towards the top of the image, and no blue colours for colour determination, this is likely the reason for incorrect determination. As image 2.2 is grayscale colour determination is not possible, there is also a very low complexity level and no strong edges, on these features the algorithms would not have been able to base a determination, for this reason the image was rejected. In images 2.3 and 2.4 there is stronger complexity in the right and left (respectively) of the images, this is likely the reason for incorrect determination.

11.0 Conclusions

This chapter examines the outcomes of the project, how well the goals were met, the possible extensions and improvements to the project, and alternative approaches to the project.

11.1 Meeting the Objectives

The main objective of this project was “to create a simple software package to process batches of digital images, orientating them correctly - with a reasonable success rate”. This objective has been met with the completion of a functioning system.

The user requirements of the project, and the extent to which they are met are detailed as follows:

R.1 To develop a software package with a complete graphical user interface following human computer interaction principles, to facilitate ease of use and efficiency. The software package should provide the following functionality:

- i. *The ability to select a series of images to process;*
- ii. *Save the processed images to a directory;*
- iii. *Progress indication of this process;*
- iv. *A measure of confidence in the returned rotation for each image.*

The interface provides this functionality with a ‘user friendly’ interface following human-computer interaction principles.

R.2 To implement an effective automatic rotation algorithm for:

- i. *Outdoor photographs;*
- ii. *Indoor photographs.*

The system implements an effective automatic rotation algorithm for outdoor images, and a partially effective automatic rotation algorithm for indoor images. This requirement has been met.

R.3 To achieve a success rate of roughly 80% for such a complete algorithm.

In implementing a complete algorithm the system has been proven to achieve a success rate of 92% for outdoor images, and a 40% success rate for indoor images, giving an average of 66%. Therefore this requirement has been partially met.

A.1 Broadening the application to process more categories of images, and so increasing its usefulness.

This advanced requirement has not been met.

A.2 Porting the application to multiple platforms.

This advanced requirement has not been met, the system is however implemented to allow easy porting of the image processing segment.

A.3 Increasing the algorithm success rate.

The success rate for outdoor images exceeds the 80% target success rate, however for indoor images the success rate is only 40%. Therefore it cannot be said this advanced requirement has been met overall.

11.2 Improvements and Extensions

There are several ways in which the system could be improved upon, and several areas in to which the project could be extended. These are discussed here.

11.2.1 Improvements

The following areas of the systems operation could be improved upon:

- **The accuracy of indoor image processing**

The success rate of 40% for indoor images could be improved upon with more determination methods or image category classification.

- **The meaning of the confidence measurement**

Currently the relevance of confidence measurement is not clear - i.e. images with low confidences are still rotated - and the accuracy of the confidence produced is not very meaningful. The use of confidence measurements in the system could be improved.

- **Aborting processing**

During usability testing one user commented that the processing cannot be aborted once it has started, this functionality could be implemented.

- **Reducing processing time by streamlining the system or multi-threading**

- **Remedy system crash on non-existent file**

Currently the system will crash when trying to process files that have been removed after opening in the interface, the existence of files could be checked for before processing, removing this problem.

- **Preview caching**

Currently previews are cached to memory, if several preview images are very large memory will be used up quickly, and images must be page-swapped to disk. Caching smaller previews or caching to disk may solve this.

11.2.2 Extensions

There are a few areas into which the project could be extended, these are described here:

- Expanding the system to be compatible with more image formats
Currently the system will only accept images of types JPEG, GIF, and PNG, the ability to process other image types is a possible extension.
- Implementing more feature extraction methods, therefore enabling the system to deal with more image categories.
- Expanding the system's functionality into other feature extraction areas, these could include locating people or objects within scenes, or extracting meta-data from an image such as the season in which a photograph was taken.

11.3 Alternative Methodologies

As previously mentioned in this report, a common approach to the problem of automatic image orientation is that of a 'neural network' - in which an artificial intelligence system is designed and trained to categorise images based on identified patterns. This approach has been examined in many other research papers, some of which are discussed in chapter 3.0.

This project has chosen not to implement such a method, taking an algorithmic approach to the problem. The 'neural network' approach is however a viable - and an often preferred alternative.

11.4 Overall Success of the Project

The project has been successful in meeting its main goal and in meeting most of the requirements set out for it.

It is clear from the work conducted throughout this project that implementing a system that is able to correctly orientate images from several image categories is difficult. Each category of image will contain different patterns indicating their

orientation, although these patterns may not hold true for every image in that category, it is therefore difficult to implement a system flexible enough to account for each image category.

This project has proven the viability of a system capable of correctly orientating images automatically, and thoroughly researched into the area of feature extraction from digital images in general.

12.0 Bibliography

APPLE INC. Cocoa Reference. Updated 2007. Accessed 21 April 2007 <<http://developer.apple.com/reference/Cocoa/>>

FISCHER, R. Feature Detectors. Updated 2003. Accessed 10 April 2007 <<http://homepages.inf.ed.ac.uk/rbf/HIPR2/featops.htm>>

KELLER, Bill. "Software Design Lecture Notes." (2006) The Department of Informatics, The University of Sussex.

LISCHNER, Ray. *C++ in a Nutshell*. California, O'Reilly Media Inc., 2003.

LOUDON, Kyle. *C++ Pocket Reference*. California, O'Reilly Media Inc., 2003.

LYU, Siwei. "Automatic Image Orientation Detection with Natural Image Statistics." (2005). Accessed 31 March 2007 <<http://www.cns.nyu.edu/~lsw/files/mm05.pdf>>

MCGRATH, Mike. *C++ Programming...in easy steps*. Warwickshire, In easy steps, 2005.

NORMAN, Donald A.. *The Design of Everyday Things*. New York, Basic Books, 2002.

SHARP, Helen & ROGERS, Yvonne & PREECE, Jenny. *Interaction Design: Beyond Human Computer Interaction*. Indianapolis, Wiley Publishing Inc., 2007.

SU, Zhou. "Automatic Image Orientation Detection." (2004). Accessed 31 March 2007 <<http://www.dcs.shef.ac.uk/intranet/teaching/projects/archive/msc2004/pdf/m3zs2.pdf>>

TRENT, Michael & MCCORMACK, Drew. *Beginning Mac OS X Programming*. Indianapolis, Wiley Publishing Inc., 2005.

TSCHUMPERLÉ, David. The CImg Library Documentation. Updated 2007. Accessed 21 April 2007 <<http://cimg.sourceforge.net/reference/index.html>>

VAILAYA, Aditya. "Automatic Image Orientation Detection." (1999). Accessed 31 March 2007 <<http://citeseer.ist.psu.edu/223869.html>>

VAILAYA, Aditya et al. "Automatic Image Orientation Detection." (2000). Accessed 31 March 2007 <http://research.microsoft.com/asia/dload_files/group/mcomputing/2003P/11tip07-vailaya-proof.pdf>

ZHANG, Lei & LI, Mingjing & ZHANG, Hong-Jiang. "Boosting Image Orientation Detection with Indoor vs. Outdoor Classification." (2001). Accessed 31 March 2007 <<http://research.microsoft.com/users/leizhang/Paper/WACV02.pdf>>

13.0 References

- ¹ Film & Film Processing - US. Updated 2004. Accessed 21 March 2007 <<http://www.mindbranch.com/listing/product/R560-1061.html>>
- ² Industry statistics. Updated 2004. Accessed 21 March 2007 <http://www.bigplanet.com/corp/company/industry_statistics.shtml>
- ³ The British Computer Society. Updated 2007. Accessed 24 March 2007 <<http://www.bcs.org/>>
- ⁴ BCS Code of Conduct. Updated 2006. Accessed 24 March 2007 <<http://www.bcs.org/server.php?show=nav.6030>>
- ⁵ See [4]
- ⁶ BCS Code of Practice. Updated 2006. Accessed 24 March 2007 <<http://www.bcs.org/upload/pdf/cop.pdf>>
- ⁷ See [6]
- ⁸ See [6]
- ⁹ Adobe Photoshop product page. Updated 2006. Accessed 30 March 2007 <<http://www.adobe.com/products/photoshop/>>
- ¹⁰ GNU GIMP web-site. Updated 2006. Accessed 30 March 2007 <<http://www.gimp.org/>>
- ¹¹ SU, Zhou. "Automatic Image Orientation Detection." (2004). Accessed 31 March 2007 <<http://www.dcs.shef.ac.uk/intranet/teaching/projects/archive/msc2004/pdf/m3zs2.pdf>>
- ¹² LYU, Siwei. "Automatic Image Orientation Detection with Natural Image Statistics." (2005). Accessed 31 March 2007 <<http://www.cns.nyu.edu/~lsw/files/mm05.pdf>>
- ¹³ VAILAYA, Aditya. "Automatic Image Orientation Detection." (1999). Accessed 31 March 2007 <<http://citeseer.ist.psu.edu/223869.html>>
- ¹⁴ ZHANG, Lei & LI, Mingjing & ZHANG, Hong-Jiang. "Boosting Image Orientation Detection with Indoor vs. Outdoor Classification." (2001). Accessed 31 March 2007 <<http://research.microsoft.com/users/leizhang/Paper/WACV02.pdf>>
- ¹⁵ VAILAYA, Aditya et al. "Automatic Image Orientation Detection." (2000). Accessed 31 March 2007 <http://research.microsoft.com/asia/dload_files/group/mcomputing/2003P/11tip07-vailaya-proof.pdf>
- ¹⁶ See [11]

¹⁷ See [14]

¹⁸ Apple Cocoa technology. Updated 2007. Accessed 4 April 2007 <<http://developer.apple.com/cocoa/>>

¹⁹ Apple Inc. Updated 2007. Accessed 4 April 2007 <<http://www.apple.com>>

²⁰ The CImg C++ Library. Updated 2007. Accessed 5 April 2007 <<http://cimg.sourceforge.net/>>

²¹ Sun Microsystems - Java Technology. Updated 2007. Accessed 4 April 2007 <<http://java.sun.com>>

²² See [21]

²³ See [20]

²⁴ Feature Detectors. Updated 2003. Accessed 10 April 2007 <<http://homepages.inf.ed.ac.uk/rbf/HIPR2/featops.htm>>

²⁵ The Software Development Process. Updated 2007. Accessed 24 April 2007. <http://en.wikipedia.org/wiki/Software_development_process>

²⁶ Nielsens Heuristics Principles. Updated 2007. Accessed 16 April 2007 <http://www.useit.com/papers/heuristic/heuristic_list.html>

²⁷ See [18]

²⁸ Apple Developer Connection: Mac OS X. Updated 2007. Accessed 4 April 2007 <<http://developer.apple.com/macosx>>

²⁹ WANDSCHNEIDER, Marc. Rotating an NSImage object in Cocoa. Updated 2006. Accessed 22 April 2007 <<http://www.chipmunkninja.com/article/nsimagerotate>>

³⁰ See [20]

³¹ See [18]

³² White box testing. Updated 2005. Accessed 19 April 2007 <<http://openseminar.org/se/modules/46/index/screen.do>>

³³ Black box testing. Updated 2005. Accessed 19 April 2007 <<http://openseminar.org/se/modules/47/index/screen.do>>

³⁴ See [32]

14.0 Appendices

14.1 Appendix A - Testing Results

14.1.1 Usability Surveys

**Rotation User Interface
Usability Survey**

The premise of the **Rotation** application is as follows:
The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being "strongly agree" ranging to 5 "strongly disagree".

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Why not these variables here?

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Very easy to do, nice preview image, program got a bit slow here though

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Very easy, just wished some online processing.

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Take a minute to figure out how to click - to make just one. Confusing. Right?

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

Thank you for completing the survey.

**Rotation User Interface
Usability Survey**

The premise of the **Rotation** application is as follows:
The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being "strongly agree" ranging to 5 "strongly disagree".

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

This is called add in the menu though

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Don't tell you if the preview is rotated or not.

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Can't stop the processing if you don't it without closing same.

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

Thank you for completing the survey.

Rotation User Interface Usability Survey

The premise of the **Rotation** application is as follows:

The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being 'strongly agree' ranging to 5 'strongly disagree'.

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

☐ 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

⊕ button, not very helpful, I picked (choose) button first and was
gk stuck. Had to be shown, so what was the point by made to be.

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

easy

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

error msg came up which didn't tell me how to
sort out what I had done wrong. Variables weren't
right. Easy to use though

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

'info' - state image "show image"
pink overlay

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

Easy peasy

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

Just exploratory, once realised that the 2 icons meant +, -
I didn't right not realistic and just select 'remove all'

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

Help file, or just a description of buttons.
Otherwise succinct and well thought out structure.
Very impressed!

Thank you for completing the survey.

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

☐ 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

Show/Hide info it's not clear enough. Perhaps show/hide preview
would be better. Need to state what the preview image has been done.

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

☐ 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

Perhaps save using a V1 panel, having to shift
forward find is inconvenient. No feedback to the user.

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5

Any comments/improvements/suggestions:

⊕ ⊖ are slightly less intuitive, I only realised
their function as they were next to the 'remove all'
button.

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

Very intuitive, great UI!

Thank you for completing the survey.

Rotation User Interface Usability Survey

The premise of the **Rotation** application is as follows:

The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being 'strongly agree' ranging to 5 'strongly disagree'.

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5

Any comments/improvements/suggestions:

Call 'open an image' - 'open', not 'Add'.

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

Any comments/improvements/suggestions:

Rotation User Interface Usability Survey

The premise of the **Rotation** application is as follows:

The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being "strongly agree" ranging to 5 "strongly disagree".

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

one I found out to use the "+" button it was easy but it should be open all the time

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

easy + obvious (I used the shortcut **Ctrl+P**)

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Easy enough. Although it's not obvious what the variables actually do

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

very obvious button + keyboard shortcut

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

ctrl type into save key
It doesn't save but then deleted before the processing
re complete the processing

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

just click remove

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

Make this into a game

Thank you for completing the survey.

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

just clicked "show/hide" info

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

maybe instead of process button and save
pick box use "save and process" button.
Told me I needed to choose a location to save to. Good.

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

"=" button confusing.

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

change the naming of buttons slightly

Thank you for completing the survey.

Rotation User Interface Usability Survey

The premise of the **Rotation** application is as follows:

The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being "strongly agree" ranging to 5 "strongly disagree".

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

I had problems as the button to open an image is "+" and in the menu it is called "Add..."

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

very easy, just clicked "process"

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

I don't know what those variables do though

Rotation User Interface Usability Survey

The premise of the **Rotation** application is as follows:

The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being "strongly agree" ranging to 5 "strongly disagree".

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Just clicked "+" button

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Just clicked "Process" button. The error box did not give enough info though.

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

What do these variables do?

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Easy but maybe show if image is rotated in preview or not more clearly.

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Change "-" to "remove"

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

Remove variables changers + make automated.
Improve error messages.

Thank you for completing the survey.

Rotation User Interface Usability Survey

The premise of the **Rotation** application is as follows:

The application is designed to provide a simple method of determining the correct orientation of any images whose orientations are undetermined. The user will provide an image(s) - generally photographs - the application then processes these images to analyse their correct orientation, with the ability to save the rotated images.

Please answer the following questions with a 1 to 5 rating, 1 being "strongly agree" ranging to 5 "strongly disagree".

1. Please try to perform the following operation: "Open an image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Easy yes

2. Please try to perform the following operation: "Process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Easy

3. Please try to perform the following operation: "Change the variables and process the image(s)".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

Didn't change anything

4. Please try to perform the following operation: "Preview the image rotation using the info drawer".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

5. Please try to perform the following operation: "Process the image(s), also saving them".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

6. Please try to perform the following operation: "Remove the image(s) from the list".
This operation is easy to perform.

1 2 3 4 5

Any comments/improvements/suggestions:

How do I remove just one?

7. Are there any improvements you can suggest that you think would ease use of the application, or do you have any other comments?

No

Thank you for completing the survey.

14.1.2 Time Performance

200x150 grayscale	
	0.6953
	0.3388
	0.3060
	0.2783
	0.2869
	0.2892
	0.3766
	0.3103
	0.2718
	0.2846
	0.3438 seconds

800x600 grayscale	
	1.889
	1.913
	1.904
	1.895
	1.874
	2.042
	2.049
	1.909
	1.939
	1.892
	1.931 seconds

1600x1200 grayscale	
	3.823
	3.713
	3.749
	3.723
	3.700
	3.724
	3.707
	3.725
	3.739
	3.697
	3.730 seconds

2800x2100 grayscale	
	5.685
	5.556
	5.567
	5.583
	5.368
	5.339
	5.266
	5.368
	5.268
	5.482
	5.448 seconds

200x150 colour	
	0.7217
	0.3652
	0.4199
	0.3392
	0.3474
	0.3591
	0.3516
	0.3622
	0.3462
	0.3843
	0.3997 seconds

800x600 colour	
	2.037
	2.060
	2.074
	1.987
	2.057
	2.051
	1.947
	2.023
	2.062
	2.045
	2.034 seconds

1600x1200 colour	
	3.845
	3.807
	3.834
	3.797
	3.907
	3.805
	3.811
	3.822
	3.817
	4.040
	3.849 seconds

2800x2100 colour	
	5.872
	5.440
	5.489
	5.469
	5.468
	5.637
	5.449
	5.434
	5.457
	5.667
	5.538 seconds

14.1.3 CPU Performance

Idle

0.1
0.1
0.1
0.1
0.1 %

Previewing

6.8
7.3
7.3
7.4
7.2 %

Processing

37.1
-
43.0
24.2
35 %

14.1.4 Memory Performance

Idle

7.25
7.23
7.23
7.24
7.24 MB

Previewing

11.11
13.09
14.54
15.79
13.6 MB

Processing

16.03
17.50
19.20
18.71
17.86 MB

14.2 Appendix B - Experiment Results

14.2.1 Rotation Results

Nick in the gallery.jpg					
DSC01434.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	270	0	361	0
5	0	270	5	361	0
10	0	270	10	361	0
15	0	270	15	361	0
20	0	270	20	361	0
25	0	270	25	361	0
30	0	270	30	361	0
35	0	270	35	361	0
40	0	270	40	361	0
45	270	270	45	361	0
50	270	270	50	361	0

DSC00145.jpg					
DSCF0636.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	180	0	361	0
5	0	180	5	361	0
10	0	270	10	361	0
15	180	270	15	361	0
20	180	270	20	361	0
25	180	270	25	361	0
30	180	270	30	361	0
35	180	270	35	361	0
40	180	270	40	361	0
45	180	270	45	361	0
50	180	270	50	361	0

DSCF0326.jpg					
DSCF0071.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	361	0	361	90
5	0	361	5	361	90
10	0	361	10	361	90
15	0	361	15	361	90
20	0	361	20	361	90
25	0	361	25	361	90
30	0	361	30	361	90
35	0	361	35	361	90
40	0	361	40	361	90
45	0	361	45	361	90
50	0	361	50	361	90

DSCF1258.jpg					
DSC00841.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	270	0	361	0
5	0	270	5	361	0
10	0	270	10	361	0
15	0	270	15	361	0
20	0	270	20	361	0
25	0	270	25	361	0
30	0	270	30	361	0
35	0	270	35	361	0
40	0	270	40	361	0
45	0	270	45	361	0
50	0	270	50	361	0

DSCF0618.jpg					
DSC01840.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	361	0	361	270
5	0	361	5	361	270
10	0	361	10	361	270
15	0	361	15	361	270
20	0	361	20	361	270
25	0	361	25	361	270
30	0	361	30	361	270
35	0	361	35	361	270
40	0	361	40	361	270
45	0	361	45	361	270
50	0	361	50	361	270

Counts			
variable value	correct rotation	incorrect rotation	Rejected
0	1	2	2
5	1	2	2
10	1	2	2
15	1	2	2
20	1	2	2
25	1	2	2
30	1	2	2
35	1	2	2
40	1	2	2
45	1	2	2
50	2	1	2

Rotation Indoor Colour Determination Results

DSCF0163.jpg					
DSCF1282.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	270	270
5	0	0	5	270	270
10	0	0	10	270	270
15	0	0	15	270	270
20	0	0	20	270	270
25	0	0	25	270	270
30	0	0	30	270	270
35	0	0	35	270	270
40	0	0	40	270	270
45	0	0	45	270	270
50	0	0	50	270	270

DSCF1116.jpg					
DSCF0127.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	90	0	0	0
5	0	90	5	0	0
10	0	90	10	0	0
15	0	90	15	0	0
20	0	90	20	0	0
25	0	90	25	0	0
30	0	90	30	0	0
35	0	90	35	0	0
40	0	90	40	0	0
45	0	90	45	0	0
50	0	90	50	0	0

DSCF0153.jpg					
Sunset from my room 2.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	270	0	180	0
5	0	270	5	180	0
10	0	270	10	180	0
15	0	270	15	180	0
20	0	270	20	180	0
25	0	270	25	180	0
30	0	270	30	180	0
35	0	270	35	180	0
40	0	270	40	180	0
45	0	270	45	180	0
50	0	270	50	180	0

DSC01884.jpg					
View onto Dolomites.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	90	0	270	270
5	0	90	5	270	270
10	0	90	10	270	270
15	0	90	15	270	270
20	0	90	20	270	270
25	0	90	25	270	270
30	0	90	30	270	270
35	0	90	35	270	270
40	0	90	40	270	270
45	0	90	45	270	270
50	0	90	50	270	270

DSC01173.jpg					
DSC00703.jpg					
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	270	0	0	0
5	0	270	5	0	0
10	0	270	10	0	0
15	0	270	15	0	0
20	0	270	20	0	0
25	0	270	25	0	0
30	0	270	30	0	0
35	0	270	35	0	0
40	0	270	40	0	0
45	0	270	45	0	0
50	0	270	50	0	0

Counts			
variable value	Correct rotation	Incorrect rotation	Rejected
0	10	0	0
5	10	0	0
10	9	0	0
15	9	0	0
20	9	0	0
25	9	0	0
30	9	0	0
35	9	0	0
40	9	0	0
45	9	0	0
50	9	0	0

Rotation Outdoor Colour Determination Results

DSC01840.jpg				Nick in the gallery.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	270	0	270	270	270
10	180	0	270	10	180	270	270
20	180	0	270	20	180	270	270
30	180	0	270	30	180	270	270
40	180	0	270	40	90	270	270
50	180	0	270	50	90	270	270
60	180	0	270	60	90	270	270
70	180	0	270	70	90	270	270
80	180	0	270	80	90	270	270
90	180	0	270	90	361	270	270
100	180	0	270	100	361	270	270

DSC01496.jpg				DSCF0071.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	90	0	90	90	90
10	180	0	90	10	0	90	90
20	180	0	90	20	0	90	90
30	180	0	90	30	0	90	90
40	180	0	90	40	0	90	90
50	180	0	90	50	0	90	90
60	180	0	90	60	90	90	90
70	180	0	90	70	90	90	90
80	180	0	90	80	90	90	90
90	180	0	90	90	90	90	90
100	180	0	90	100	90	90	90

DSCF0218.jpg				DSCF0326.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	10	180	0	0	0
10	180	0	10	180	0	10	180
20	180	0	20	180	0	20	180
30	180	0	30	180	0	30	180
40	180	0	40	180	0	40	180
50	180	0	50	180	0	50	180
60	180	0	60	180	0	60	180
70	180	0	70	180	0	70	180
80	180	0	80	180	0	80	180
90	180	0	90	180	0	90	180
100	180	0	100	180	0	100	180

DSCF0636.jpg				DSC01312.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	10	0	90	90	90
10	0	0	10	0	270	90	90
20	0	0	20	0	90	90	90
30	0	0	30	0	90	90	90
40	0	0	40	0	90	90	90
50	0	0	50	0	90	90	90
60	0	0	60	0	90	90	90
70	0	0	70	0	90	90	90
80	0	0	80	0	90	90	90
90	0	0	90	0	90	90	90
100	0	0	100	0	90	90	90

DSC00145.jpg				A140255			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	270	0	0	270	270
10	270	0	270	10	180	270	270
20	270	0	270	20	180	270	270
30	270	0	270	30	180	270	270
40	270	0	270	40	180	270	270
50	361	0	270	50	180	270	270
60	361	0	270	60	180	270	270
70	361	0	270	70	180	270	270
80	361	0	270	80	180	270	270
90	361	0	270	90	180	270	270
100	361	0	270	100	180	270	270

Hugo Interviews Paul_2.jpg				DSC00841.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	10	90	0	0	0
10	90	0	10	90	0	10	90
20	90	0	20	90	0	20	90
30	90	0	30	90	0	30	90
40	90	0	40	90	0	40	90
50	90	0	50	90	0	50	90
60	90	0	60	90	0	60	90
70	90	0	70	90	0	70	90
80	90	0	80	90	0	80	90
90	90	0	90	90	0	90	90
100	90	0	100	90	0	100	90

DSCF0618.jpg				DSC01434.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	270	0	0	0	0
10	0	0	270	10	90	0	0
20	270	0	270	20	90	0	0
30	270	0	270	30	90	0	0
40	270	0	270	40	90	0	0
50	270	0	270	50	90	0	0
60	270	0	270	60	90	0	0
70	270	0	270	70	90	0	0
80	270	0	270	80	90	0	0
90	270	0	270	90	90	0	0
100	270	0	270	100	90	0	0

DSCF1258.jpg			
variable value	rotation	correct rotation	variable value
0	0	0	270
10	0	0	270
20	270	0	270
30	270	0	270
40	361	0	270
50	361	0	270
60	361	0	270
70	361	0	270
80	361	0	270
90	361	0	270
100	361	0	270

Counts			
variable value	Correct rotation	Incorrect rotation	Rejected
0	4	0	0
10	5	10	0
20	0	0	0
30	7	8	0
40	6	8	1
50	5	8	0
60	6	7	2
70	7	6	2
80	7	6	2
90	7	5	3
100	7	5	3

Rotation Indoor Complexity Determination Results

DSCF0153.jpg			DSC00703.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	270	0	0	0
10	0	180	10	0	0
20	0	90	20	0	0
30	0	0	30	0	0
40	0	90	40	0	0
50	0	180	50	0	0
60	0	270	60	0	0
70	361	270	70	361	0
80	361	270	80	361	0
90	361	270	90	361	0
100	361	270	100	361	0

A bay on the ring of Kerry, Ireland.jpg			DSC01637.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	270	270
10	0	0	10	270	270
20	0	0	20	270	270
30	0	0	30	270	270
40	0	0	40	270	270
50	0	0	50	270	270
60	0	0	60	270	270
70	0	0	70	270	270
80	0	0	80	270	270
90	0	0	90	270	270
100	0	0	100	270	270

View onto Dolomites.jpg			DSCF1116.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	270	0	90	90
10	0	270	10	90	90
20	180	270	20	90	90
30	90	270	30	90	90
40	0	270	40	90	90
50	0	270	50	90	90
60	180	270	60	361	90
70	361	270	70	361	90
80	361	270	80	361	90
90	361	270	90	361	90
100	361	270	100	361	90

DSCF0127.jpg			Sunset from my room 2.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	0
10	0	0	10	0	0
20	0	0	20	90	0
30	0	0	30	90	0
40	0	0	40	0	0
50	0	0	50	0	0
60	90	0	60	361	0
70	361	0	70	361	0
80	361	0	80	361	0
90	361	0	90	361	0
100	361	0	100	361	0

DSCF0163.jpg			A lake in southern Ireland.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	90	270
10	0	0	10	270	270
20	0	0	20	270	270
30	0	0	30	270	270
40	0	0	40	270	270
50	361	0	50	270	270
60	361	0	60	270	270
70	361	0	70	361	270
80	361	0	80	361	270
90	361	0	90	361	270
100	361	0	100	361	270

DSC01684.jpg			DSCF1282.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	270	90	0	270	270
10	270	90	10	270	270
20	0	90	20	270	270
30	0	90	30	270	270
40	0	90	40	270	270
50	361	90	50	270	270
60	361	90	60	361	270
70	361	90	70	361	270
80	361	90	80	361	270
90	361	90	90	361	270
100	361	90	100	361	270

DSC01173.jpg			Dolomites snow.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	270	270	0	270	270
10	270	270	10	270	270
20	270	270	20	270	270
30	270	270	30	270	270
40	0	270	40	180	270
50	90	270	50	270	270
60	90	270	60	270	270
70	90	270	70	270	270
80	361	270	80	270	270
90	361	270	90	270	270
100	361	270	100	270	270

DSCF0147.jpg		
variable value	rotation	correct rotation
0	0	0
10	0	0
20	0	0
30	0	0
40	0	0
50	90	0
60	90	0
70	90	0
80	90	0
90	90	0
100	90	0

Counts			
	Correct rotation	Incorrect rotation	Rejected
0	13	2	0
10	13	2	0
20	11	4	0
30	11	4	0
40	10	4	1
50	8	4	1
60	5	5	5
70	10	1	10
80	3	1	11
90	3	1	11
100	3	1	11

Rotation outdoor Complexity Determination Results

DSC01840.jpg			DSCF0618.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	270
10	0	0	10	0	270
20	0	0	20	0	270
30	0	0	30	0	270
40	0	0	40	0	270
50	0	0	50	0	270
60	0	0	60	0	270
70	0	0	70	0	270
80	0	0	80	0	270
90	0	0	90	0	270
100	0	0	100	0	270

DSCF0218.jpg			Hugo Interviews Paul_2.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	90
10	0	0	10	0	90
20	0	0	20	0	90
30	0	0	30	0	90
40	0	0	40	0	90
50	0	0	50	0	90
60	0	0	60	0	90
70	0	0	70	0	90
80	0	0	80	0	90
90	0	0	90	0	90
100	0	0	100	0	90

DSCF0324.jpg			DSCF0071.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	90
10	0	0	10	0	90
20	0	0	20	0	90
30	0	0	30	0	90
40	0	0	40	0	90
50	0	0	50	0	90
60	0	0	60	0	90
70	0	0	70	0	90
80	0	0	80	0	90
90	0	0	90	0	90
100	0	0	100	0	90

DSCF0636.jpg			DSCF1258.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	270
10	0	0	10	0	270
20	0	0	20	0	270
30	0	0	30	0	270
40	0	0	40	0	270
50	0	0	50	0	270
60	0	0	60	0	270
70	0	0	70	0	270
80	0	0	80	0	270
90	0	0	90	0	270
100	0	0	100	0	270

DSC01312.jpg			Nick in the Gallery.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	270
10	0	0	10	0	270
20	0	0	20	0	270
30	0	0	30	0	270
40	0	0	40	0	270
50	0	0	50	0	270
60	0	0	60	0	270
70	0	0	70	0	270
80	0	0	80	0	270
90	0	0	90	0	270
100	0	0	100	0	270

DSC01496.jpg			DSC00145.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	270
10	0	0	10	0	270
20	0	0	20	0	270
30	0	0	30	0	270
40	0	0	40	0	270
50	0	0	50	0	270
60	0	0	60	0	270
70	0	0	70	0	270
80	0	0	80	0	270
90	0	0	90	0	270
100	0	0	100	0	270

DSC1434.jpg			DSC00841.jpg		
variable value	rotation	correct rotation	variable value	rotation	correct rotation
0	0	0	0	0	0
10	0	0	10	0	0
20	0	0	20	0	0
30	0	0	30	0	0
40	0	0	40	0	0
50	0	0	50	0	0
60	0	0	60	0	0
70	0	0	70	0	0
80	0	0	80	0	0
90	0	0	90	0	0
100	0	0	100	0	0

A140255.jpg		
variable value	rotation	correct rotation
0	0	90
10	0	90
20	0	90
30	0	90
40	0	90
50	0	90
60	0	90
70	0	90
80	0	90
90	0	90
100	0	90

Counts			
variable value	Correct rotation	Incorrect rotation	Rejected
0	4	10	0
10	5	10	0
20	5	10	0
30	5	10	0
40	5	10	0
50	5	10	0
60	5	10	0
70	5	10	0
80	4	11	0
90	4	11	0
100	5	10	0

Rotation indoor edge Determination Results

DSCF0163.jpg				A bay on the ring or Kerry, Ireland.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	0	0	0	0	0
10	90 or 270	0	10	0	0	10	0
20	90 or 270	0	20	0	0	20	0
30	90 or 270	0	30	0	0	30	0
40	90 or 270	0	40	0	0	40	0
50	90 or 270	0	50	0	0	50	0
60	90 or 270	0	60	0	0	60	0
70	90 or 270	0	70	0	0	70	0
80	90 or 270	0	80	0	0	80	0
90	90 or 270	0	90	0	0	90	0
100	90 or 270	0	100	0	0	100	0

DSCF01637.jpg				DSC01684.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	0	0	0	0	0
10	0	270	10	0	270	10	0
20	0	270	20	90 or 270	90	20	0
30	0	270	30	90 or 270	90	30	0
40	0	270	40	90 or 270	90	40	0
50	0	270	50	90 or 270	90	50	0
60	0	270	60	90 or 270	90	60	0
70	0	270	70	90 or 270	90	70	0
80	0	270	80	90 or 270	90	80	0
90	0	270	90	90 or 270	90	90	0
100	90 or 270	0	100	90 or 270	90	100	0

A lake in Southern Ireland.jpg				DSCF1116.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	0	0	0	0	0
10	0	270	10	0	0	10	0
20	0	270	20	0	0	20	0
30	0	270	30	0	0	30	0
40	0	270	40	0	0	40	0
50	0	270	50	90 or 270	90	50	0
60	0	270	60	90 or 270	90	60	0
70	0	270	70	90 or 270	90	70	0
80	0	270	80	90 or 270	90	80	0
90	0	270	90	90 or 270	90	90	0
100	0	270	100	90 or 270	90	100	0

DSCF1282.jpg				DSCF0153.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	0	0	0	0	0
10	0	270	10	0	0	10	0
20	0	270	20	0	0	20	0
30	0	270	30	0	0	30	0
40	0	270	40	0	0	40	0
50	0	270	50	0	0	50	0
60	0	270	60	0	0	60	0
70	0	270	70	0	0	70	0
80	0	270	80	0	0	80	0
90	0	270	90	0	0	90	0
100	90 or 270	0	100	90 or 270	0	100	0

DSCF0147.jpg				DSC00703.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	0	0	0	0	0
10	90 or 270	0	10	90 or 270	0	10	0
20	90 or 270	0	20	90 or 270	0	20	0
30	90 or 270	0	30	90 or 270	0	30	0
40	90 or 270	0	40	90 or 270	0	40	0
50	90 or 270	0	50	90 or 270	0	50	0
60	90 or 270	0	60	90 or 270	0	60	0
70	90 or 270	0	70	90 or 270	0	70	0
80	90 or 270	0	80	90 or 270	0	80	0
90	90 or 270	0	90	90 or 270	0	90	0
100	90 or 270	0	100	90 or 270	0	100	0

DSCF0127.jpg				Dolomites snow.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	0	0	0	0	0
10	90 or 270	0	10	90 or 270	270	10	0
20	90 or 270	0	20	90 or 270	270	20	0
30	90 or 270	0	30	90 or 270	270	30	0
40	90 or 270	0	40	90 or 270	270	40	0
50	90 or 270	0	50	90 or 270	270	50	0
60	90 or 270	0	60	90 or 270	270	60	0
70	90 or 270	0	70	90 or 270	270	70	0
80	90 or 270	0	80	90 or 270	270	80	0
90	90 or 270	0	90	90 or 270	270	90	0
100	90 or 270	0	100	90 or 270	270	100	0

DSCF0173.jpg				Sunset from my room 2.jpg			
variable value	rotation	correct rotation	variable value	rotation	correct rotation	variable value	rotation
0	0	0	0	0	0	0	0
10	0	270	10	90 or 270	0	10	0
20	0	270	20	90 or 270	0	20	0
30	0	270	30	90 or 270	0	30	0
40	0	270	40	90 or 270	0	40	0
50	0	270	50	90 or 270	0	50	0
60	0	270	60	90 or 270	0	60	0
70	0	270	70	90 or 270	0	70	0
80	0	270	80	90 or 270	0	80	0
90	0	270	90	90 or 270	0	90	0
100	0	270	100	90 or 270	0	100	0

View onto Dolomites.jpg			
variable value	rotation	correct rotation	variable value
0	0	0	0
10	0	270	10
20	0	270	20
30	0	270	30
40	0	270	40
50	0	270	50
60	0	270	60
70	0	270	70
80	0	270	80
90	0	270	90
100	0	270	100

Counts			
variable value	Correct rotation	Incorrect rotation	Rejected
0	1	0	1
10	1	14	0
20	1	13	0
30	2	13	0
40	2	13	0
50	3	12	0
60	3	12	0
70	3	12	0
80	3	12	0
90	3	12	0
100	1	0	0

Rotation outdoor edge Determination Results

14.2.2 Confidence Results

<p>DSC00145.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>61</td></tr> <tr><td>5</td><td>66</td></tr> <tr><td>10</td><td>63</td></tr> <tr><td>15</td><td>61</td></tr> <tr><td>20</td><td>61</td></tr> <tr><td>25</td><td>60</td></tr> <tr><td>30</td><td>59</td></tr> <tr><td>35</td><td>52</td></tr> <tr><td>40</td><td>48</td></tr> <tr><td>45</td><td>44</td></tr> <tr><td>50</td><td>42</td></tr> </table>	variable value	confidence	0	61	5	66	10	63	15	61	20	61	25	60	30	59	35	52	40	48	45	44	50	42	<p>DSC01840.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>-</td></tr> <tr><td>5</td><td>-</td></tr> <tr><td>10</td><td>-</td></tr> <tr><td>15</td><td>-</td></tr> <tr><td>20</td><td>-</td></tr> <tr><td>25</td><td>-</td></tr> <tr><td>30</td><td>-</td></tr> <tr><td>35</td><td>-</td></tr> <tr><td>40</td><td>-</td></tr> <tr><td>45</td><td>-</td></tr> <tr><td>50</td><td>-</td></tr> </table>	variable value	confidence	0	-	5	-	10	-	15	-	20	-	25	-	30	-	35	-	40	-	45	-	50	-
variable value	confidence																																																
0	61																																																
5	66																																																
10	63																																																
15	61																																																
20	61																																																
25	60																																																
30	59																																																
35	52																																																
40	48																																																
45	44																																																
50	42																																																
variable value	confidence																																																
0	-																																																
5	-																																																
10	-																																																
15	-																																																
20	-																																																
25	-																																																
30	-																																																
35	-																																																
40	-																																																
45	-																																																
50	-																																																
<p>DSC0071.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>-</td></tr> <tr><td>5</td><td>-</td></tr> <tr><td>10</td><td>-</td></tr> <tr><td>15</td><td>-</td></tr> <tr><td>20</td><td>-</td></tr> <tr><td>25</td><td>-</td></tr> <tr><td>30</td><td>-</td></tr> <tr><td>35</td><td>-</td></tr> <tr><td>40</td><td>-</td></tr> <tr><td>45</td><td>-</td></tr> <tr><td>50</td><td>-</td></tr> </table>	variable value	confidence	0	-	5	-	10	-	15	-	20	-	25	-	30	-	35	-	40	-	45	-	50	-	<p>DSC0326.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>-</td></tr> <tr><td>5</td><td>-</td></tr> <tr><td>10</td><td>-</td></tr> <tr><td>15</td><td>-</td></tr> <tr><td>20</td><td>-</td></tr> <tr><td>25</td><td>-</td></tr> <tr><td>30</td><td>-</td></tr> <tr><td>35</td><td>-</td></tr> <tr><td>40</td><td>-</td></tr> <tr><td>45</td><td>-</td></tr> <tr><td>50</td><td>-</td></tr> </table>	variable value	confidence	0	-	5	-	10	-	15	-	20	-	25	-	30	-	35	-	40	-	45	-	50	-
variable value	confidence																																																
0	-																																																
5	-																																																
10	-																																																
15	-																																																
20	-																																																
25	-																																																
30	-																																																
35	-																																																
40	-																																																
45	-																																																
50	-																																																
variable value	confidence																																																
0	-																																																
5	-																																																
10	-																																																
15	-																																																
20	-																																																
25	-																																																
30	-																																																
35	-																																																
40	-																																																
45	-																																																
50	-																																																
<p>DSC0618.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>-</td></tr> <tr><td>5</td><td>-</td></tr> <tr><td>10</td><td>-</td></tr> <tr><td>15</td><td>-</td></tr> <tr><td>20</td><td>-</td></tr> <tr><td>25</td><td>-</td></tr> <tr><td>30</td><td>-</td></tr> <tr><td>35</td><td>-</td></tr> <tr><td>40</td><td>-</td></tr> <tr><td>45</td><td>-</td></tr> <tr><td>50</td><td>-</td></tr> </table>	variable value	confidence	0	-	5	-	10	-	15	-	20	-	25	-	30	-	35	-	40	-	45	-	50	-	<p>DSC0636.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>-</td></tr> <tr><td>5</td><td>-</td></tr> <tr><td>10</td><td>-</td></tr> <tr><td>15</td><td>-</td></tr> <tr><td>20</td><td>-</td></tr> <tr><td>25</td><td>-</td></tr> <tr><td>30</td><td>-</td></tr> <tr><td>35</td><td>-</td></tr> <tr><td>40</td><td>-</td></tr> <tr><td>45</td><td>-</td></tr> <tr><td>50</td><td>-</td></tr> </table>	variable value	confidence	0	-	5	-	10	-	15	-	20	-	25	-	30	-	35	-	40	-	45	-	50	-
variable value	confidence																																																
0	-																																																
5	-																																																
10	-																																																
15	-																																																
20	-																																																
25	-																																																
30	-																																																
35	-																																																
40	-																																																
45	-																																																
50	-																																																
variable value	confidence																																																
0	-																																																
5	-																																																
10	-																																																
15	-																																																
20	-																																																
25	-																																																
30	-																																																
35	-																																																
40	-																																																
45	-																																																
50	-																																																
<p>DSC1258.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>99</td></tr> <tr><td>5</td><td>99</td></tr> <tr><td>10</td><td>99</td></tr> <tr><td>15</td><td>99</td></tr> <tr><td>20</td><td>99</td></tr> <tr><td>25</td><td>99</td></tr> <tr><td>30</td><td>99</td></tr> <tr><td>35</td><td>99</td></tr> <tr><td>40</td><td>99</td></tr> <tr><td>45</td><td>99</td></tr> <tr><td>50</td><td>99</td></tr> </table>	variable value	confidence	0	99	5	99	10	99	15	99	20	99	25	99	30	99	35	99	40	99	45	99	50	99	<p>Nick in the Gallery.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>78</td></tr> <tr><td>5</td><td>78</td></tr> <tr><td>10</td><td>78</td></tr> <tr><td>15</td><td>73</td></tr> <tr><td>20</td><td>71</td></tr> <tr><td>25</td><td>68</td></tr> <tr><td>30</td><td>68</td></tr> <tr><td>35</td><td>64</td></tr> <tr><td>40</td><td>62</td></tr> <tr><td>45</td><td>64</td></tr> <tr><td>50</td><td>64</td></tr> </table>	variable value	confidence	0	78	5	78	10	78	15	73	20	71	25	68	30	68	35	64	40	62	45	64	50	64
variable value	confidence																																																
0	99																																																
5	99																																																
10	99																																																
15	99																																																
20	99																																																
25	99																																																
30	99																																																
35	99																																																
40	99																																																
45	99																																																
50	99																																																
variable value	confidence																																																
0	78																																																
5	78																																																
10	78																																																
15	73																																																
20	71																																																
25	68																																																
30	68																																																
35	64																																																
40	62																																																
45	64																																																
50	64																																																

<p>DSC00841.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>-</td></tr> <tr><td>5</td><td>-</td></tr> <tr><td>10</td><td>-</td></tr> <tr><td>15</td><td>-</td></tr> <tr><td>20</td><td>-</td></tr> <tr><td>25</td><td>-</td></tr> <tr><td>30</td><td>-</td></tr> <tr><td>35</td><td>-</td></tr> <tr><td>40</td><td>-</td></tr> <tr><td>45</td><td>-</td></tr> <tr><td>50</td><td>-</td></tr> </table>	variable value	confidence	0	-	5	-	10	-	15	-	20	-	25	-	30	-	35	-	40	-	45	-	50	-	<p>DSC01434.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>-</td></tr> <tr><td>5</td><td>-</td></tr> <tr><td>10</td><td>-</td></tr> <tr><td>15</td><td>-</td></tr> <tr><td>20</td><td>-</td></tr> <tr><td>25</td><td>-</td></tr> <tr><td>30</td><td>-</td></tr> <tr><td>35</td><td>-</td></tr> <tr><td>40</td><td>-</td></tr> <tr><td>45</td><td>-</td></tr> <tr><td>50</td><td>-</td></tr> </table>	variable value	confidence	0	-	5	-	10	-	15	-	20	-	25	-	30	-	35	-	40	-	45	-	50	-
variable value	confidence																																																
0	-																																																
5	-																																																
10	-																																																
15	-																																																
20	-																																																
25	-																																																
30	-																																																
35	-																																																
40	-																																																
45	-																																																
50	-																																																
variable value	confidence																																																
0	-																																																
5	-																																																
10	-																																																
15	-																																																
20	-																																																
25	-																																																
30	-																																																
35	-																																																
40	-																																																
45	-																																																
50	-																																																
<p>Count</p> <table> <tr><th></th><th>avg confidence</th></tr> <tr><td>0</td><td>85</td></tr> <tr><td>5</td><td>81</td></tr> <tr><td>10</td><td>79</td></tr> <tr><td>15</td><td>78</td></tr> <tr><td>20</td><td>77</td></tr> <tr><td>25</td><td>76</td></tr> <tr><td>30</td><td>74</td></tr> <tr><td>35</td><td>72</td></tr> <tr><td>40</td><td>70</td></tr> <tr><td>45</td><td>69</td></tr> <tr><td>50</td><td>68</td></tr> </table>		avg confidence	0	85	5	81	10	79	15	78	20	77	25	76	30	74	35	72	40	70	45	69	50	68																									
	avg confidence																																																
0	85																																																
5	81																																																
10	79																																																
15	78																																																
20	77																																																
25	76																																																
30	74																																																
35	72																																																
40	70																																																
45	69																																																
50	68																																																

Confidence indoor colour Results

<p>DSC01613.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>100</td></tr> <tr><td>5</td><td>100</td></tr> <tr><td>10</td><td>100</td></tr> <tr><td>15</td><td>100</td></tr> <tr><td>20</td><td>100</td></tr> <tr><td>25</td><td>99</td></tr> <tr><td>30</td><td>99</td></tr> <tr><td>35</td><td>99</td></tr> <tr><td>40</td><td>99</td></tr> <tr><td>45</td><td>99</td></tr> <tr><td>50</td><td>99</td></tr> </table>	variable value	confidence	0	100	5	100	10	100	15	100	20	100	25	99	30	99	35	99	40	99	45	99	50	99	<p>DSC0127.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>100</td></tr> <tr><td>5</td><td>100</td></tr> <tr><td>10</td><td>100</td></tr> <tr><td>15</td><td>100</td></tr> <tr><td>20</td><td>100</td></tr> <tr><td>25</td><td>100</td></tr> <tr><td>30</td><td>100</td></tr> <tr><td>35</td><td>100</td></tr> <tr><td>40</td><td>99</td></tr> <tr><td>45</td><td>97</td></tr> <tr><td>50</td><td>93</td></tr> </table>	variable value	confidence	0	100	5	100	10	100	15	100	20	100	25	100	30	100	35	100	40	99	45	97	50	93
variable value	confidence																																																
0	100																																																
5	100																																																
10	100																																																
15	100																																																
20	100																																																
25	99																																																
30	99																																																
35	99																																																
40	99																																																
45	99																																																
50	99																																																
variable value	confidence																																																
0	100																																																
5	100																																																
10	100																																																
15	100																																																
20	100																																																
25	100																																																
30	100																																																
35	100																																																
40	99																																																
45	97																																																
50	93																																																
<p>DSC00703.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>100</td></tr> <tr><td>5</td><td>100</td></tr> <tr><td>10</td><td>100</td></tr> <tr><td>15</td><td>100</td></tr> <tr><td>20</td><td>99</td></tr> <tr><td>25</td><td>99</td></tr> <tr><td>30</td><td>85</td></tr> <tr><td>35</td><td>74</td></tr> <tr><td>40</td><td>63</td></tr> <tr><td>45</td><td>53</td></tr> <tr><td>50</td><td>42</td></tr> </table>	variable value	confidence	0	100	5	100	10	100	15	100	20	99	25	99	30	85	35	74	40	63	45	53	50	42	<p>DSC01684.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>100</td></tr> <tr><td>5</td><td>100</td></tr> <tr><td>10</td><td>100</td></tr> <tr><td>15</td><td>99</td></tr> <tr><td>20</td><td>99</td></tr> <tr><td>25</td><td>99</td></tr> <tr><td>30</td><td>99</td></tr> <tr><td>35</td><td>99</td></tr> <tr><td>40</td><td>99</td></tr> <tr><td>45</td><td>99</td></tr> <tr><td>50</td><td>99</td></tr> </table>	variable value	confidence	0	100	5	100	10	100	15	99	20	99	25	99	30	99	35	99	40	99	45	99	50	99
variable value	confidence																																																
0	100																																																
5	100																																																
10	100																																																
15	100																																																
20	99																																																
25	99																																																
30	85																																																
35	74																																																
40	63																																																
45	53																																																
50	42																																																
variable value	confidence																																																
0	100																																																
5	100																																																
10	100																																																
15	99																																																
20	99																																																
25	99																																																
30	99																																																
35	99																																																
40	99																																																
45	99																																																
50	99																																																
<p>DSC1282.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>100</td></tr> <tr><td>5</td><td>100</td></tr> <tr><td>10</td><td>100</td></tr> <tr><td>15</td><td>100</td></tr> <tr><td>20</td><td>100</td></tr> <tr><td>25</td><td>100</td></tr> <tr><td>30</td><td>100</td></tr> <tr><td>35</td><td>100</td></tr> <tr><td>40</td><td>100</td></tr> <tr><td>45</td><td>99</td></tr> <tr><td>50</td><td>99</td></tr> </table>	variable value	confidence	0	100	5	100	10	100	15	100	20	100	25	100	30	100	35	100	40	100	45	99	50	99	<p>DSC1116.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>93</td></tr> <tr><td>5</td><td>90</td></tr> <tr><td>10</td><td>89</td></tr> <tr><td>15</td><td>90</td></tr> <tr><td>20</td><td>89</td></tr> <tr><td>25</td><td>90</td></tr> <tr><td>30</td><td>90</td></tr> <tr><td>35</td><td>89</td></tr> <tr><td>40</td><td>88</td></tr> <tr><td>45</td><td>83</td></tr> <tr><td>50</td><td>77</td></tr> </table>	variable value	confidence	0	93	5	90	10	89	15	90	20	89	25	90	30	90	35	89	40	88	45	83	50	77
variable value	confidence																																																
0	100																																																
5	100																																																
10	100																																																
15	100																																																
20	100																																																
25	100																																																
30	100																																																
35	100																																																
40	100																																																
45	99																																																
50	99																																																
variable value	confidence																																																
0	93																																																
5	90																																																
10	89																																																
15	90																																																
20	89																																																
25	90																																																
30	90																																																
35	89																																																
40	88																																																
45	83																																																
50	77																																																
<p>DSC01173.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>100</td></tr> <tr><td>5</td><td>100</td></tr> <tr><td>10</td><td>100</td></tr> <tr><td>15</td><td>100</td></tr> <tr><td>20</td><td>100</td></tr> <tr><td>25</td><td>100</td></tr> <tr><td>30</td><td>100</td></tr> <tr><td>35</td><td>100</td></tr> <tr><td>40</td><td>99</td></tr> <tr><td>45</td><td>98</td></tr> <tr><td>50</td><td>97</td></tr> </table>	variable value	confidence	0	100	5	100	10	100	15	100	20	100	25	100	30	100	35	100	40	99	45	98	50	97	<p>Sunset from my room 2.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>96</td></tr> <tr><td>5</td><td>90</td></tr> <tr><td>10</td><td>90</td></tr> <tr><td>15</td><td>83</td></tr> <tr><td>20</td><td>83</td></tr> <tr><td>25</td><td>82</td></tr> <tr><td>30</td><td>80</td></tr> <tr><td>35</td><td>73</td></tr> <tr><td>40</td><td>63</td></tr> <tr><td>45</td><td>50</td></tr> <tr><td>50</td><td>32</td></tr> </table>	variable value	confidence	0	96	5	90	10	90	15	83	20	83	25	82	30	80	35	73	40	63	45	50	50	32
variable value	confidence																																																
0	100																																																
5	100																																																
10	100																																																
15	100																																																
20	100																																																
25	100																																																
30	100																																																
35	100																																																
40	99																																																
45	98																																																
50	97																																																
variable value	confidence																																																
0	96																																																
5	90																																																
10	90																																																
15	83																																																
20	83																																																
25	82																																																
30	80																																																
35	73																																																
40	63																																																
45	50																																																
50	32																																																

<p>DSC0153.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>100</td></tr> <tr><td>5</td><td>100</td></tr> <tr><td>10</td><td>97</td></tr> <tr><td>15</td><td>97</td></tr> <tr><td>20</td><td>97</td></tr> <tr><td>25</td><td>97</td></tr> <tr><td>30</td><td>96</td></tr> <tr><td>35</td><td>96</td></tr> <tr><td>40</td><td>96</td></tr> <tr><td>45</td><td>96</td></tr> <tr><td>50</td><td>96</td></tr> </table>	variable value	confidence	0	100	5	100	10	97	15	97	20	97	25	97	30	96	35	96	40	96	45	96	50	96	<p>View onto Dolomites.jpg</p> <table> <tr><th>variable value</th><th>confidence</th></tr> <tr><td>0</td><td>95</td></tr> <tr><td>5</td><td>95</td></tr> <tr><td>10</td><td>89</td></tr> <tr><td>15</td><td>89</td></tr> <tr><td>20</td><td>89</td></tr> <tr><td>25</td><td>89</td></tr> <tr><td>30</td><td>89</td></tr> <tr><td>35</td><td>86</td></tr> <tr><td>40</td><td>82</td></tr> <tr><td>45</td><td>76</td></tr> <tr><td>50</td><td>66</td></tr> </table>	variable value	confidence	0	95	5	95	10	89	15	89	20	89	25	89	30	89	35	86	40	82	45	76	50	66
variable value	confidence																																																
0	100																																																
5	100																																																
10	97																																																
15	97																																																
20	97																																																
25	97																																																
30	96																																																
35	96																																																
40	96																																																
45	96																																																
50	96																																																
variable value	confidence																																																
0	95																																																
5	95																																																
10	89																																																
15	89																																																
20	89																																																
25	89																																																
30	89																																																
35	86																																																
40	82																																																
45	76																																																
50	66																																																
<p>Count</p> <table> <tr><th></th><th>avg confidence</th></tr> <tr><td>0</td><td>96</td></tr> <tr><td>5</td><td>97</td></tr> <tr><td>10</td><td>96</td></tr> <tr><td>15</td><td>96</td></tr> <tr><td>20</td><td>96</td></tr> <tr><td>25</td><td>95</td></tr> <tr><td>30</td><td>94</td></tr> <tr><td>35</td><td>93</td></tr> <tr><td>40</td><td>89</td></tr> <tr><td>45</td><td>85</td></tr> <tr><td>50</td><td>80</td></tr> </table>		avg confidence	0	96	5	97	10	96	15	96	20	96	25	95	30	94	35	93	40	89	45	85	50	80																									
	avg confidence																																																
0	96																																																
5	97																																																
10	96																																																
15	96																																																
20	96																																																
25	95																																																
30	94																																																
35	93																																																
40	89																																																
45	85																																																
50	80																																																

Confidence outdoor colour Results

DSCF0326.jpg

variable value	confidence
0	1
10	66
20	63
30	62
40	60
50	69
60	69
70	69
80	69
90	69
100	69

DSC00145.jpg

variable value	confidence
0	14
10	20
20	42
30	62
40	75
50	-
60	-
70	-
80	-
90	-
100	-

DSC00841.jpg

variable value	confidence
0	24
10	73
20	100
30	100
40	100
50	100
60	100
70	100
80	100
90	100
100	100

DSC01840.jpg

variable value	confidence
0	27
10	50
20	48
30	71
40	88
50	98
60	100
70	100
80	100
90	100
100	100

DSC01496.jpg

variable value	confidence
0	0
10	68
20	94
30	99
40	100
50	100
60	100
70	100
80	100
90	100
100	100

Nick in the Gallery.jpg

variable value	confidence
0	0
10	54
20	53
30	54
40	62
50	68
60	60
70	48
80	100
90	100
100	100

DSCF0218.jpg

variable value	confidence
0	11
10	98
20	98
30	98
40	98
50	98
60	98
70	98
80	98
90	98
100	98

DSCF0071.jpg

variable value	confidence
0	8
10	75
20	68
30	51
40	40
50	77
60	100
70	100
80	100
90	100
100	100

Hugo interviews Paul_2.jpg

variable value	confidence
0	57
10	57
20	65
30	73
40	81
50	85
60	93
70	96
80	100
90	100
100	100

DSCF1258.jpg

variable value	confidence
0	15
10	52
20	65
30	73
40	100
50	-
60	-
70	-
80	-
90	-
100	-

DSC01434.jpg

variable value	confidence
0	23
10	43
20	54
30	70
40	93
50	96
60	93
70	97
80	97
90	97
100	97

A140255.jpg

variable value	confidence
0	4
10	44
20	46
30	41
40	37
50	50
60	71
70	81
80	81
90	81
100	81

DSCF0636.jpg

variable value	confidence
0	14
10	43
20	99
30	99
40	99
50	99
60	99
70	99
80	99
90	99
100	99

DSCF0618.jpg

variable value	confidence
0	60
10	91
20	96
30	98
40	99
50	99
60	99
70	99
80	99
90	99
100	99

DSC01312.jpg

variable value	confidence
0	23
10	43
20	64
30	86
40	84
50	84
60	84
70	84
80	84
90	84
100	84

Count

variable value	avg confidence
0	34
10	72
20	37
30	36
40	33
50	32
60	30
70	29
80	28
90	28
100	24

Confidence indoor complexity Results

View onto Dolomites.jpg

variable value	confidence
0	30
10	46
20	72
30	69
40	87
50	88
60	92
70	-
80	-
90	-
100	-

DSCF0153.jpg

variable value	confidence
0	14
10	36
20	86
30	97
40	100
50	100
60	100
70	-
80	-
90	-
100	-

DSC0703.jpg

variable value	confidence
0	83
10	89
20	83
30	90
40	91
50	97
60	100
70	-
80	-
90	-
100	-

DSC01684.jpg

variable value	confidence
0	10
10	71
20	97
30	100
40	-
50	-
60	-
70	-
80	-
90	-
100	-

DSCF0147.jpg

variable value	confidence
0	41
10	90
20	97
30	99
40	100
50	100
60	100
70	100
80	100
90	100
100	100

DSCF1282.jpg

variable value	confidence
0	18
10	85
20	81
30	74
40	73
50	-
60	-
70	-
80	-
90	-
100	-

DSCF0163.jpg

variable value	confidence
0	58
10	95
20	92
30	87
40	95
50	-
60	-
70	-
80	-
90	-
100	-

DSCF1116.jpg

variable value	confidence
0	49
10	80
20	83
30	91
40	90
50	97
60	-
70	-
80	-
90	-
100	-

DSCF01637.jpg

variable value	confidence
0	51
10	90
20	100
30	100
40	100
50	100
60	100
70	100
80	100
90	100
100	100

DSC01173.jpg

variable value	confidence
0	57
10	89
20	82
30	61
40	13
50	83
60	97
70	97
80	-
90	-
100	-

DSCF0127.jpg

variable value	confidence
0	63
10	94
20	90
30	95
40	95
50	95
60	60
70	-
80	-
90	-
100	-

A bay on the ring of kerry, Ireland.jpg

variable value	confidence
0	30
10	72
20	78
30	80
40	80
50	80
60	82
70	75
80	75
90	100
100	100

A lake in Southern Ireland.jpg

variable value	confidence
0	23
10	72
20	99
30	99
40	100
50	100
60	100
70	-
80	-
90	-
100	-

Dolomites snow.jpg

variable value	confidence
0	37
10	65
20	74
30	72
40	81
50	100
60	100
70	100
80	100
90	100
100	100

Sunset from my room 2.jpg

variable value	confidence
0	43
10	53
20	50
30	53
40	90
50	95
60	100
70	-
80	-
90	-
100	-

Count

variable value	avg confidence
0	30
10	37
20	37
30	30
40	33
50	33
60	30
70	29
80	28
90	28
100	24

Confidence outdoor complexity Results

variable value	confidence
0	89
10	95
20	76
30	69
40	64
50	57
60	52
70	48
80	45
90	43
100	37

variable value	confidence
0	74
10	65
20	57
30	50
40	46
50	40
60	35
70	34
80	32
90	28
100	23

variable value	confidence
0	53
10	43
20	38
30	32
40	30
50	28
60	23
70	22
80	20
90	19
100	14

variable value	confidence
0	5
10	7
20	10
30	7
40	4
50	5
60	6
70	9
80	8
90	6
100	13

variable value	confidence
0	0
10	0
20	0
30	0
40	0
50	0
60	0
70	0
80	0
90	0
100	0

variable value	confidence
0	21
10	16
20	10
30	7
40	4
50	5
60	4
70	4
80	4
90	0
100	2

variable value	confidence
0	14
10	17
20	20
30	14
40	11
50	11
60	9
70	7
80	8
90	7
100	0

variable value	confidence
0	8
10	20
20	24
30	24
40	24
50	22
60	21
70	18
80	15
90	18
100	15

variable value	confidence
0	20
10	13
20	9
30	8
40	7
50	5
60	5
70	5
80	4
90	4
100	0

variable value	confidence
0	15
10	7
20	0
30	0
40	4
50	5
60	6
70	6
80	4
90	6
100	10

variable value	confidence
0	41
10	39
20	35
30	31
40	29
50	26
60	24
70	22
80	21
90	20
100	14

variable value	confidence
0	14
10	10
20	10
30	5
40	4
50	3
60	3
70	3
80	3
90	3
100	1

variable value	confidence
0	53
10	38
20	25
30	17
40	11
50	5
60	2
70	0
80	0
90	0
100	2

variable value	confidence
0	21
10	38
20	21
30	17
40	20
50	18
60	17
70	16
80	15
90	14
100	5

variable value	confidence
0	38
10	34
20	25
30	21
40	19
50	10
60	15
70	14
80	13
90	12
100	5

Count	avg confidence
0	31
5	19
10	23
15	20
20	14
25	16
30	15
35	14
40	13
45	12
50	10

Confidence indoor edge Results

variable value	confidence
0	76
10	82
20	83
30	84
40	81
50	78
60	75
70	73
80	71
90	69
100	64

variable value	confidence
0	13
10	22
20	20
30	22
40	27
50	26
60	25
70	23
80	22
90	21
100	21

variable value	confidence
0	20
10	20
20	18
30	12
40	12
50	8
60	7
70	6
80	6
90	6
100	4

variable value	confidence
0	0
10	14
20	22
30	26
40	27
50	25
60	24
70	23
80	22
90	21
100	23

variable value	confidence
0	28
10	48
20	60
30	62
40	64
50	60
60	58
70	58
80	58
90	55
100	52

variable value	confidence
0	9
10	11
20	16
30	20
40	22
50	23
60	24
70	23
80	23
90	22
100	23

variable value	confidence
0	77
10	66
20	54
30	45
40	46
50	33
60	29
70	25
80	24
90	24
100	17

variable value	confidence
0	25
10	23
20	16
30	13
40	12
50	8
60	6
70	6
80	5
90	5
100	2

variable value	confidence
0	81
10	84
20	76
30	75
40	66
50	61
60	57
70	54
80	51
90	49
100	44

variable value	confidence
0	20
10	12
20	5
30	2
40	0
50	0
60	0
70	1
80	1
90	1
100	0

variable value	confidence
0	19
10	20
20	17
30	16
40	15
50	13
60	11
70	10
80	10
90	9
100	4

variable value	confidence
0	0
10	19
20	17
30	16
40	15
50	13
60	11
70	10
80	10
90	9
100	30

variable value	confidence
0	38
10	32
20	25
30	21
40	19
50	16
60	15
70	14
80	13
90	12
100	5

variable value	confidence
0	92
10	87
20	78
30	71
40	66
50	58
60	55
70	51
80	48
90	45
100	40

variable value	confidence
0	3
10	18
20	31
30	25
40	37
50	35
60	33
70	32
80	31
90	30
100	30

Count	avg confidence
0	34
5	17
10	37
15	36
20	35
25	32
30	30
35	29
40	29
45	26
50	24

Confidence outdoor edge Results

Colour	Indoor	Outdoor
	0	80
	5	81
	10	79
	15	78
	20	77
	25	76
	30	74
	35	72
	40	70
	45	69
	50	68

Complexity	Indoor	Outdoor
	0	34
	5	37
	10	37
	15	36
	20	35
	25	32
	30	30
	35	29
	40	28
	45	26
	50	24

Edge	Indoor	Outdoor
	0	31
	5	28
	10	23
	15	20
	20	19
	25	16
	30	15
	35	14
	40	13
	45	12
	50	10

Confidence overall Results

14.2.3 Collaborative Results

Outdoor images for N=5%, T=20%, S=100%			
Image	Confidence	Rotation	Correct rotation
A bay on the ring of keyy, Ireland	46	0	0
A lake in Southern Ireland	60	0	270
Bridge.jpg	90	0	0
Dolomites snow.jpg	42	270	270
DSC00703.jpg	91	0	0
DSC01070.jpg	99	270	270
DSC01173.jpg	91	270	270
DSC01631.jpg	99	0	0
DSC01637.jpg	50	270	270
DSC01684.jpg	51	90	90
DSC00098.jpg	70	90	90
DSC0127.jpg	97	0	0
DSC0147.jpg	97	0	0
DSC0153.jpg	100	270	270
DSC0163.jpg	96	0	0
DSC0166.jpg	94	270	270
DSC0329.jpg	51	90	90
DSC0711.jpg	92	0	0
DSC1116.jpg	58	90	90
DSC1282.jpg	61	270	270
DSC1389.jpg	19	270	270
DSC1413.jpg	100	270	270
Hugo.jpg	78	90	270
Sunset from my room 2.jpg	90	0	0
View onto Dolomites.jpg	90	270	270

Indoor images for N=5%, T=20%, S=100%			
Image	Confidence	Rotation	Correct rotation
A149255.jpg	23	0	90
A140259.jpg	96	0	0
DSC00145.jpg	34	0	270
DSC00841.jpg	53	270	0
DSC01189.jpg	76	0	0
DSC01312.jpg	64	90	90
DSC01434.jpg	54	90	0
DSC01496.jpg	94	0	90
DSC01762.jpg	87	0	90
DSC01840.jpg	48	0	270
DSC00071.jpg	36	0	90
DSC0081.jpg	89	90	90
DSC0087.jpg	12	0	0
DSC0137.jpg	83	90	90
DSC0218.jpg	101	361	0
DSC0251.jpg	35	0	90
DSC0326.jpg	32	270	0
DSC0618.jpg	96	270	270
DSC0636.jpg	99	0	0
DSC0732.jpg	44	90	270
DSC0804.jpg	31	90	0
DSC1258.jpg	93	270	270
Hugo Interviews Paul_2.jpg	33	90	90
Nick in the Gallery.jpg	39	0	270
Richard at the vision mixing des	79	90	270

Counts			
	Correct rotation	Incorrect rotation	Rejected
Indoor	10	14	1
Outdoor	23	2	0

14.3 Appendix C - Source Code

14.3.1 Interface Controller

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
#import "RotationController.h"

@implementation RotationController

/* Called on interface 'waking'. Initialises variables */
- (void)awakeFromNib
{
    //Create data structure for list of images
    records = [[NSMutableArray alloc] init];

    //Timer to control refresh of preview information
    timer = [NSTimer scheduledTimerWithTimeInterval: 0.1
        target: self
        selector: @selector(handleTimer:)
        userInfo: nil
        repeats: YES];

    //Set progress indicator to update on a separate
    //thread to the application loop
    [progressInd setUsesThreading:YES];

    //Set minimum confidence level
    minConfidence = 25;
}

/* Method to add files to the datastructure */
- (IBAction)addFiles:(id)sender
{
    /* Open a file selector window with a limit of available file types */
    NSOpenPanel *panel = [NSOpenPanel openPanel];
    [panel setCanChooseDirectories:NO];
    [panel setCanChooseFiles:YES];
    [panel setAllowsMultipleSelection:YES];
    NSArray *fileTypes = [NSArray arrayWithObjects:@"*.jpg", @"*.JPG", @"*.gif", @"*.GIF", @"*.png", @"*.PNG", nil];
    [panel setTitle:@"Please Choose Some Images"];
    [panel setPrompt:@"Choose"];

    //Variables to store in
    NSMutableArray *fileList;
    int chosen;

    //Run the file selector, storing in variables
    if (chosen = [panel runModalForTypes:fileTypes]) {
        fileList = [panel filenames];
    }

    //Check if the cancel button was hit
    if (chosen != 0) {
        NSEnumerator *enumerator = [fileList objectEnumerator];
        id object;

        //Enumerate over array
    }
}

```

Printed: 24/04/2007 09:43

Page 1

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
while (object = [enumerator nextObject]) {
    //Get file path out
    NSString *path = object;

    //Create a record (multi-part 'dictionary')
    NSMutableDictionary *record = [[NSMutableDictionary alloc] init];

    //Set objects in the dictionary for: the image path, codes that it's
    //unchecked, //that there's no valid preview.
    [record setObject:path forKey:@"ImagePath"];
    [record setObject:[NSNumber numberWithInt:101] forKey:@"Confidence"];
    [record setObject:[NSNumber numberWithInt:361] forKey:@"Rotation"];
    [record setObject:[NSNumber numberWithInt:0] forKey:@"previewValid"];

    //Add record to array
    [records addObject:record];
}

[files reloadData]; //Call for reload of the data to the table

[files deselectAll:self]; //Deselect all rows in the file list
}

/* Method to remove selected files from the datastructure */
- (IBAction)deleteFiles:(id)sender
{
    //Get the selected rows of the file list
    NSIndexPath *indexPathSet = [files selectedRowIndexes];
    unsigned int numberSelected = [indexPathSet count];

    //Check for no files selected
    if (numberSelected == 0) {
        NSRunAlertPanel(@"Error", @"No files selected to remove.", @"OK", NULL, NULL);
        return;
    }

    //Move selected indexes into buffer
    unsigned int indexBuffer[numberSelected];
    unsigned limit = [indexPathSet getIndexes:indexBuffer maxCount:[indexPathSet count]
        inIndexRange:NSMakeRange(0, [indexPathSet count])];
    unsigned idx;

    //Create temp variables to save to
    NSMutableArray *tempArray = [NSMutableArray array];
    id tempObject;

    //Store each object to be removed in a temporary array
    for (idx = 0; idx < limit; idx++) {
        tempObject = [records objectAtIndex:indexBuffer[idx]];
        [tempArray addObject:tempObject];
    }
}

```

Printed: 24/04/2007 09:43

Page 2

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm

//Now remove all these objects
[records removeObjectAtIndex:numberOfFiles];

//Call to reload the data to the table
[files reloadData];

//Deselect all rows in the file list
[files deselectAll:self];
}

/* Overwrite abstract method for datastructure of a table, returns number of table
rows */
- (int)numberOfRowsInTableView:(NSTableView *)aTableView
{
    return [records count];
}

/* Overwrite abstract method for datastructure of a table, returns information about
the cell specified */
- (id)tableView:(NSTableView *)aTableView
    objectValueForTableColumn:(NSTableColumn *)aTableColumn
    row:(int)rowIndex
{
    //Store data from datastructure about specified cell
    id theRecord, theValue;

    theRecord = [records objectAtIndex:rowIndex];
    NSString *ident = [aTableColumn identifier];

    theValue = [theRecord objectForKey:ident];

    //Look for confidence error code
    //Return "Unchecked" if value is 101, or value + "%" if not
    if ([ident compare:@"Confidence"] == NSOrderedSame) {
        if ([theValue intValue] == 101)
            return @"Unchecked";
        else
            return [NSString stringWithFormat:@"%d%%", [theValue intValue],
                @"%"];
    }

    //Look for rotation error code
    //Return "-" if value is 361, or value + "" if not
    if ([ident compare:@"Rotation"] == NSOrderedSame) {
        if ([theValue intValue] == 361)
            return @"";
        else
            return [NSString stringWithFormat:@"%d%%", [theValue intValue],
                @""];
    }

    //Otherwise just return the value
}

```

Printed: 24/04/2007 09:43

Page 3

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
return theValue;
}

/* Method to process the opened files */
/* The datastructure is queried, the number of files checked for 0 result.
* The buttons disabled
* Data inputs on the interface and the saving checked
* For each file in the datastructure the path is passed to be processed and the
progress tracked
* on the interface progress bar
* Errors are detected from the returned values
* The interface is told to refresh and the buttons are enabled again.
*/
- (IBAction)processList:(id)sender
{
    //Get number of files in data structure
    int numberOfFiles = [records count];

    //Check for files existing
    if (numberOfFiles != 0) {
        //Disable all the buttons
        [self disableButtons];

        //Set progress bar max to number of images
        [progressInd setMaxValue:numberOfFiles];

        //Get slider values
        int segments = [segmentsSlider intValue];
        int threshold = [thresholdSlider intValue];
        int spread = [spreadValueSlider intValue];

        //Get save checkbox value
        int save = [saveTick intValue];

        //Get save location
        NSString *savePath = [saveLocation stringValue];

        //Ensure checkbox value is 1 or 0
        //Adjust values for sliders at 0
        if (save != 1 && save != 0) { save = 1; }
        if (segments == 0) { segments = 1; }
        if (threshold == 0) { threshold = 1; }
        if (spread < 4) { spread = 4; }

        //Check for no save path supplied when save checkbox is checked
        //Check for valid directory URL
        if ([savePath length] == 0 && save == 1) {
            //Run error alert
            NSRunAlertPanel(@"Error", @"Please choose a location to save if you wish to
save the images on processing.", @"OK", NULL, NULL);
        }

        //Enable all the buttons and return out
        [self enableButtons];
    }
}

```

Printed: 24/04/2007 09:43

Page 4

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
return;
} else if([savePath length] != 0 && save == 1) {
//Check for valid directory path with NSURL
NSURL *url = [NSURL URLWithString:savePath];
BOOL isValid = [url isFileURL];

//If invalid
if(!isValid) {
//Run error alert
NSRunAlertPanel(@"Error",@"There was a problem with the directory you
specified to save. Please check this directory is valid",@"OK",NULL,NULL);

//Enable all the buttons and return out
[self enableButtons];
return;
}

//Set low confidence tracker
bool lowConf = NO;

//Loop over all files
for(int n = 0; n < numberOffiles; n++) {
//Get record to look at
id theRecord = [records objectAtIndex:n];

//Store file path
NSString *filePath = [theRecord objectForKey:@"ImagePath"];

//Increment progress indicator
[progressInd incrementBy:1];

//Redraw the window to update progress indicator
[[NSRunLoop currentRunLoop] runMode:NSEventTrackingRunLoopMode
beforeDate:[NSDate date]];

//Convert file path to string for C++
unsigned int filePathLength = [filePath length];
char temp[filePathLength + 1];
strcpy(temp, [filePath cString]);

//Create temporary variables and create C++ object
int isDone;
int rotation;
int confidence;
Image* myCpImage;
myCpImage = new Image(temp);

//Do this to save
if(save == 1) {
//Get filename
NSArray *separatedPath = [filePath componentsSeparatedByString:@"\"];
NSString *fileNameString = [separatedPath
Printed: 24/04/2007 09:43
Page 5

```

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
objectAtIndex:([separatedPath count] - 1));

//Concatenate file name onto save path
NSString *fullSavePath = [NSString stringWithFormat:@"%@%@",
savePath,
fileNameString];

//Covert the save path to C++ char array
unsigned int savePathLength = [fullSavePath length];
char temp2[savePathLength + 1];
strcpy(temp2, [fullSavePath cString]);

//Run processing method on C++ object with parameters
isDone = myCpImage -> processImage(segments, threshold, spread, save
, temp2);
} else
isDone = myCpImage -> processImage(segments, threshold, spread, save
, NULL);

//Check if an error was encountered (0 == error)
if(isDone == 1) {
//Get determined rotation and confidence
confidence = myCpImage -> getConfidence();
rotation = myCpImage -> getRotation();

//Store determined confidence and rotation, store that no valid
preview
[theRecord setObject:[NSNumber numberWithInt:confidence]
forKey:@"Confidence"];
[theRecord setObject:[NSNumber numberWithInt:rotation]
forKey:@"Rotation"];
[theRecord setObject:[NSNumber numberWithInt:0]
forKey:@"previewValid"];
} else {
//Run error alert
NSString *errPath = [NSString stringWithFormat:@"%@%@",
@"There was an error i
n processing the file at path: ",
filePath,
@" This could be due
to a system error or that no determination could be made with the set variables."];
NSRunAlertPanel(@"Error",errPath,@"OK",NULL,NULL);

//Delete C++ object as it's no longer needed
delete myCpImage;

//Check for low confidence
if(confidence < minConfidence)
lowConf = YES;
}

//On finish reset progress bar
Printed: 24/04/2007 09:43
Page 6

```

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
[progressInd setDoubleValue:0];

//Call to reload the data to the table
[files reloadData];

//Deselect all rows in the file list
[files deselectAll:self];

//Enable all buttons
[self enableButtons];

//Run alert if any images had low confidence
if(lowConf)
NSRunAlertPanel(@"Warning",@"Some files have a low confidence of rotation
",@"OK",NULL,NULL);
} else
NSRunAlertPanel(@"Error",@"There are no files to process",@"OK",NULL,NULL);
}

/* Method to disable all buttons on the interface */
- (void)disableButtons
{
[segmentsSlider setEnabled:NO];
[thresholdSlider setEnabled:NO];
[spreadValueSlider setEnabled:NO];

[addButton setEnabled:NO];
[minusButton setEnabled:NO];
[removeAllButton setEnabled:NO];
[toggleInfoButton setEnabled:NO];

[chooseButton setEnabled:NO];
[saveTick setEnabled:NO];
[processButton setEnabled:NO];
}

/* Method to enable all buttons on the interface */
- (void)enableButtons
{
[segmentsSlider setEnabled:YES];
[thresholdSlider setEnabled:YES];
[spreadValueSlider setEnabled:YES];

[addButton setEnabled:YES];
[minusButton setEnabled:YES];
[removeAllButton setEnabled:YES];
[toggleInfoButton setEnabled:YES];

[chooseButton setEnabled:YES];
[saveTick setEnabled:YES];
[processButton setEnabled:YES];
}
Printed: 24/04/2007 09:43
Page 7

```

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
/* Method called to choose a saving location */
- (IBAction)chooseSave:(id)sender
{
//Set file selector options
NSOpenPanel *panel = [NSOpenPanel openPanel];
[panel setCanChooseDirectories:YES];
[panel setCanChooseFiles:NO];
[panel setTitle:@"Please Choose A Folder"];
[panel setPrompt:@"Choose"];

//Variable to check for cancel button hit
int chosen;

//Run panel
if(chosen = [panel runModalForTypes:nil]) {
//Check for cancel button being hit
if(chosen != 0) {
//Get the directory path, append a "/", set it in text field
NSString *path = [[panel filenames] objectAtIndex:0];
NSString *absPath = [NSString stringWithFormat:@"%@/",
path];
[saveLocation setStringValue:absPath];
}
}

/* Tick/untick the save tick box */
- (IBAction)tickSave:(id)sender
{
[saveTick setState:[saveTick state]];
}

/* Increment the segments slider */
- (IBAction)incrementN:(id)sender
{
[segmentsSlider setIntValue:[segmentsSlider intValue] + 1];
}

/* Increment the threshold slider */
- (IBAction)incrementT:(id)sender
{
[thresholdSlider setIntValue:[thresholdSlider intValue] + 1];
}

/* Increment the spread slider */
- (IBAction)incrementS:(id)sender
{
[spreadValueSlider setIntValue:[spreadValueSlider intValue] + 1];
}

/* Decrement the segments slider */
- (IBAction)decrementN:(id)sender
{
Printed: 24/04/2007 09:43
Page 8

```

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
[segmentsSlider setValue:[(segmentsSlider intValue) - 1]];
}

/* Decrement the threshold slider */
- (IBAction)decrement:(id)sender
{
    [thresholdSlider setValue:[(thresholdSlider intValue) - 1]];
}

/* Decrement the spread slider */
- (IBAction)decrement:(id)sender
{
    [spreadValueSlider setValue:[(spreadValueSlider intValue) - 1]];
}

/* Method to remove all files */
- (IBAction)clearFiles:(id)sender
{
    //Get number of files in lsit
    int total = [records count];

    //Loop over files removing each
    for(int n = 0; n < total; n++) {
        [records removeObjectAtIndex:0];
    }

    //Call to reload the data to the table
    [files reloadData];

    //Deselect all rows in the file list
    [files deselectAll:self];
}

/* Toggle the info drawer */
- (IBAction)toggleDrawer:(id)sender
{
    [drawer toggle:self];
}

/* This method is called by the timer refreshing the info drawer contents */
- (void)handleTimer:(NSTimer *)timer
{
    //Check that the drawer is not closed
    if([drawer state] != 0) {
        //Count number of selected files
        NSIndexSet *indexSet = [files selectedRowIndexes];
        unsigned int numberSelected = [indexSet count];

        //Set "No files selected"
        if(numberSelected == 0)
            [drawer setContentView:NOSelectionView];

        //Set "Multiple files selected"
    }
}

```

Printed: 24/04/2007 09:43 Page 9

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
if(numberSelected > 1)
    [drawer setContentView:tableView];

//Set info about selected file
if(numberSelected == 1) {
    //Get the record from the datastructure of the selected file
    unsigned int indexBuffer[numberSelected];
    unsigned limit = [indexSet getIndexes:indexBuffer maxCount:[indexSet count]
    inIndexRange:NULL];
    NSMutableDictionary *record = [records objectAtIndex:indexBuffer[0]];

    //Get values from the record
    int confidence = [[record objectForKey:@"Confidence"] intValue];
    int rotation = [[record objectForKey:@"Rotation"] intValue];
    int previewValid = [[record objectForKey:@"previewValid"] intValue];
    NSString *path = [record objectForKey:@"ImagePath"];

    //This is added if confidence is below the set level
    NSString *append = @"";
    if(confidence < minConfidence)
        append = @" (!!)";

    /* SET FILE NAME */
    NSArray *separatedPath = [path componentsSeparatedByString:@" / "];
    NSString *fileNameString = [separatedPath objectAtIndex:[separatedPath count] - 1];
    [imageFileName setStringValue:fileNameString];

    /* SET FILE PATH */
    [imagePath setStringValue:path];

    /* SET CONFIDENCE */
    NSString *conf;
    if(confidence == 101)
        conf = @"Not processed";
    else
        conf = [NSString stringWithFormat:@"%d%%", confidence,
        @"%",
        append];

    [imageConfidence setStringValue:conf];

    /* SET ROTATION */
    NSString *rota;
    if(rotation == 361)
        rota = @"Not processed";
    else
        rota = [NSString stringWithFormat:@"%d° CW", rotation,
        append];

    [imageRotation setStringValue:rota];

    /* SET FILE TYPE */
    NSString *extension = [path substringFromIndex:[path length]-4];
    if([extension compare:@" .jpg"] == NSOrderedSame || [extension compare:@" .

```

Printed: 24/04/2007 09:43 Page 10

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
JPG" == NSOrderedSame)
    [imageKind setStringValue:@"JPG"];
    if([extension compare:@" .gif"] == NSOrderedSame || [extension compare:@" .
GIF" == NSOrderedSame)
    [imageKind setStringValue:@"GIF"];
    if([extension compare:@" .png"] == NSOrderedSame || [extension compare:@" .
PNG" == NSOrderedSame)
    [imageKind setStringValue:@"PNG"];

/* SET IMAGE PREVIEW */
UIImage *previewImage;

//Check is image has valid preview
if(previewValid == 0) {
    //Get NSURL of file path
    NSURL *fileURL = [NSURL URLWithString:path];

    //Read in the image as an UIImage
    previewImage = [[UIImage alloc] initWithContentsOfURL:fileURL];

    //Check for rotation of 90 or 270
    if(rotation == 90 || rotation == 270) {
        //Call to rotate the preview image
        //correct rotation by +180 as rotate method does CCW when we want
        CW
        previewImage = [self rotateIndividualImage:previewImage
        by:(rotation+180)];
    }

    //Set UIImage in data structure
    [record setObject:previewImage forKey:@"previewImage"];

    //Set that there is a valid preview
    [record setObject:[NSNumber numberWithInt:1] forKey:@"previewValid"];
} else {
    //Set stored UIImage
    previewImage = [record objectForKey:@"previewImage"];
}

//Set image in frame on info drawer
[imagePreview setImageScaling:NISScaleProportionally];
[imagePreview setImage:previewImage];

/* SET CONTENT VIEW TO DRAWER */
[drawer setContentView:drawerView];
}

}

/* This method is to rotate an image by 90 or 270 degrees
 *
 * This code taken from: http://www.chipmunkninja.com/article/nsimagerotat
 * modified very slightly

```

Printed: 24/04/2007 09:43 Page 11

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.mm
/*
- (UIImage *)rotateIndividualImage:(UIImage *)image by:(int)alpha
{
    CGSize existingSize = [image size];
    CGSize newSize = CGSizeMake(existingSize.height, existingSize.width);

    UIImage *rotatedImage = [[UIImage alloc] initWithSize:newSize];
    CGAffineTransform *rotateTF = [CGAffineTransform transform];

    [rotatedImage lockFocus];
    CGPoint centerPoint = CGPointMake(newSize.width / 2, newSize.height / 2);
    [rotateTF translateXBy: centerPoint.x yBy: centerPoint.y];
    [rotateTF rotateByDegrees: alpha];
    [rotateTF translateXBy: -centerPoint.x yBy: -centerPoint.x];
    [rotateTF concat];
    [image drawAtPoint:NSZeroPoint
    fromRect:NSMakeRect(0, 0, existingSize.width, existingSize.height)
    operation:NSCompositeSourceOver
    fraction:1.0];
    [rotatedImage unlockFocus];

    return rotatedImage;
}

@end

```

Printed: 24/04/2007 09:43 Page 12

14.3.2 Image Processing

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp
#include "Image.h"

/* Constructor for a new image, which can be processed */
Image::Image(const char* filePath)
{
    //The image path is stored and the value of PI is set.
    filePath = filePath;
    pi = 3.141592654;
}

/* Process the image with the passed parameters */
int Image::processImage(int N, int S, int save, const char* savePath)
{
    //Load the image at filePath
    CImg<unsigned char> image(filePath);

    //Create temp grayscale image
    CImg<unsigned char> grayscale;

    //Get number of colour channels
    int v = image.dimv();

    //If image is grayscale skip colour determination,
    //if not run colour determination then convert to grayscale
    if(v != 1) {
        colour(image, N); //Perform colour determination
        grayscale = toGrayscale(image); //Convert image to grayscale
    } else {
        //Store invalid values for colour determination
        colourRotation = 361;
        colourConfidence = 101;
        grayscale = image;
    }

    complexity(grayscale, T); //Perform complexity determination
    edge(grayscale, S); //Perform edge determination

    //Run decision merging
    //returns 0 if no determination made (error), 1 if else (OK)
    int found = calculateRotationsAndConfidences();

    //Check for no error and wishing to save.
    if((found == 1) && (save == 1)) {
        //Rotate image to amount
        image.rotate(rotation);

        //Save rotated image
        image.save(savePath);
    }

    //Return 1 or 0 (error or not)
    return found;
}

```

Printed: 24/04/2007 09:46

Page 1

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp

/* Convert the parameter image to grayscale and return the converted image */
CImg<unsigned char> Image::toGrayscale(CImg<unsigned char> image) {
    //Create grayscale image of same dimensions
    CImg<unsigned char> grayscale(image.dimx(), image.dimy());

    //Convert original image to grayscale, storing in new image
    cimg_mapXY(image,x,y)
    grayscale(x,y) = 0.3 * image(x,y,0) + 0.59 * image(x,y,1) + 0.11 *
    image(x,y,2);

    //Return grayscale converted image
    return grayscale;
}

/* Run decision merging to find the winning rotation and confidence */
int Image::calculateRotationsAndConfidences() {
    //edgeRotation = 1" is 0 or 180 degrees, "edgeRotation = 2" is 90 or 270 degrees
    //rotation = 361" or "confidence = 101" indicates error code

    //Set up counters and toggle
    int zeroCounter = 0;
    int zeroToggle = 0;
    int ninetyCounter = 0;
    int ninetyToggle = 0;
    int oneEightyCounter = 0;
    int oneEightyToggle = 0;
    int twoSeventyCounter = 0;
    int twoSeventyToggle = 0;

    //Check for colour error code
    //Store the colour determination and confidence
    if(colourConfidence != 101) {
        if(colourRotation == 0) {
            zeroCounter = zeroCounter + colourConfidence;
            zeroToggle++;
        } else if(colourRotation == 90) {
            ninetyCounter = ninetyCounter + colourConfidence;
            ninetyToggle++;
        } else if(colourRotation == 180) {
            oneEightyCounter = oneEightyCounter + colourConfidence;
            oneEightyToggle++;
        } else if(colourRotation == 270) {
            twoSeventyCounter = twoSeventyCounter + colourConfidence;
            twoSeventyToggle++;
        }
    }

    //Check for complexity error code
    //Store the complexity determination and confidence
    if(complexityConfidence != 101) {
        if(complexityRotation == 0) {
            zeroCounter = zeroCounter + complexityConfidence;

```

Printed: 24/04/2007 09:46

Page 2

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp

    zeroToggle++;
} else if(complexityRotation == 90) {
    ninetyCounter = ninetyCounter + complexityConfidence;
    ninetyToggle++;
} else if(complexityRotation == 180) {
    oneEightyCounter = oneEightyCounter + complexityConfidence;
    oneEightyToggle++;
} else if(complexityRotation == 270) {
    twoSeventyCounter = twoSeventyCounter + complexityConfidence;
    twoSeventyToggle++;
}

//Check for edge error code
//Store the edge determination and confidence
if(edgeConfidence != 101) {
    int temp = 0;
    if(edgeConfidence != 0) {
        temp = (edgeConfidence/2);
    }
    if(edgeRotation == 1) {
        zeroCounter = zeroCounter + temp;
        zeroToggle++;
        oneEightyCounter = oneEightyCounter + temp;
        oneEightyToggle++;
    } else if(edgeRotation == 2) {
        ninetyCounter = ninetyCounter + temp;
        ninetyToggle++;
        twoSeventyCounter = twoSeventyCounter + temp;
        twoSeventyToggle++;
    }
}

//Check if no determinations were made
//Find the counter that is the highest, this indicates the most common rotation
determined
//Set final confidence and rotation
if((zeroToggle != 0) || (ninetyToggle != 0) || (oneEightyToggle != 0) ||
(twoSeventyToggle != 0)) {
    if((zeroCounter > ninetyCounter) && (zeroCounter > oneEightyCounter) && (zero
Counter > twoSeventyCounter)) {
        confidence = zeroCounter / zeroToggle;
        rotation = 0;
    } else if((ninetyCounter > zeroCounter) && (ninetyCounter > oneEightyCounter)
&& (ninetyCounter > twoSeventyCounter)) {
        confidence = ninetyCounter / ninetyToggle;
        rotation = 90;
    } else if((oneEightyCounter > zeroCounter) && (oneEightyCounter > ninetyCou
nter) && (oneEightyCounter > twoSeventyCounter)) {
        confidence = oneEightyCounter / oneEightyToggle;
        rotation = 180;
    } else if((twoSeventyCounter > zeroCounter) && (twoSeventyCounter > ninetyCou
nter) && (twoSeventyCounter > oneEightyCounter)) {
        confidence = twoSeventyCounter / twoSeventyToggle;

```

Printed: 24/04/2007 09:46

Page 3

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp

    rotation = 270;
} else {
    //Store error codes, return error code 0
    rotation = 361;
    confidence = 101;
    return 0;
}

//Set result of 180 to be 0 as we never rotate 180
if(rotation == 180)
    rotation = 0;

//Return OK code
return 1;
}

/* Perform colour determination on the parameter image */
void Image::colour(CImg<unsigned char> image, int segments) {
    //Get image height and width
    int width = image.dimx();
    int height = image.dimy();

    //Get pixel size of segments
    int widthN = width * (segments/100.0);
    int heightN = height * (segments/100.0);

    //Variable to keep track of the side being examined
    int side = 0;

    //Set up counter arrays
    int redCount[4] = {0,0,0,0};
    int greenCount[4] = {0,0,0,0};
    int blueCount[4] = {0,0,0,0};

    //Loop over image, border of heightN (top and bottom)
    cimg_for_borderXY(image,x,y,heightN) {
        //Check if we're in top or bottom
        //Set "side" to 1 if top or 0 for bottom
        if((side = (y < heightN)) || (y > height - heightN)){
            //Get RGB values
            int red = image(x,y,0);
            int green = image(x,y,1);
            int blue = image(x,y,2);

            //Increment counts for dominant colours
            if((red > green) && (red > blue))
                redCount[side]++;

```

Printed: 24/04/2007 09:46

Page 4

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp
    if((green > red) && (green > blue))
        greenCount[side]++;
    if((blue > red) && (blue > green))
        blueCount[side]++;
    }
}

//Loop over image, border of widthN (left and right)
cimg_for_borderXY(image,x,y,widthN) {
    //Check if we're in left or right
    //Set "side" to 0 if left or 1 if right
    if((side = (x < widthN)) || (x > width - widthN)) {
        //Adjust side
        side = side + 2;

        //Get RGB values
        int red = image(x,y,0);
        int green = image(x,y,1);
        int blue = image(x,y,2);

        //Increment counts for dominant colours
        if((red > green) && (red > blue))
            redCount[side]++;
        if((green > red) && (green > blue))
            greenCount[side]++;
        if((blue > red) && (blue > green))
            blueCount[side]++;
    }
}

//Create a confidences array
int confidences[4] = {0,0,0,0};

//Loop over counters
for(int idx = 0; idx < 4; idx++) {
    //Check for blue counter being largest
    if((blueCount[idx] > redCount[idx]) && (blueCount[idx] > greenCount[idx])) {
        //Deal with infinity (don't divide by 0) and store confidence
        float temp = ((redCount[idx] + greenCount[idx])/2.0);
        if(temp == 0)
            confidences[idx] = 100;
        else
            confidences[idx] = (1-(temp/blueCount[idx]))*100;
    }
}

//Find the max confidence for which side
int maxIndex = 4;
int maxVal = confidences[0];
for(int idx = 1; idx < 4; idx++) {
    if(confidences[idx] > maxVal) {
        maxVal = confidences[idx];
        maxIndex = idx;
    }
}

Printed: 24/04/2007 09:46
Page 5

```

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp
    }
}

//Set rotation amount
if(maxIndex == 4) {
    if(confidences[0] != 0) {
        colourRotation = 180;
        colourConfidence = confidences[0];
    } else {
        //not found
        colourRotation = 361;
        colourConfidence = 101;
    }
} else {
    if(maxIndex == 1) colourRotation = 0;
    if(maxIndex == 2) colourRotation = 270;
    if(maxIndex == 3) colourRotation = 90;
    colourConfidence = maxVal;
}

/* Perform complexity determination on the parameter image */
void Image::complexity(Cimg_unsigned char* image, int threshold) {
    //Get image width and height
    int width = image.dimx();
    int height = image.dimy();

    //Calculate threshold value
    threshold = (255 * threshold) / 100;

    //Define 3x3 neighbourhood, I
    Cimg_3x3(I,float);

    //Define counters
    float topComplexCount=0;
    float botComplexCount=0;
    float leftComplexCount=0;
    float rightComplexCount=0;

    //Calculate half of height and width
    int heightOverTwo = height / 2;
    int widthOverTwo = width / 2;

    //Loop over image using 3x3 neighbourhood
    cimg_map3x3(image,x,y,0,0,I) {
        //Get current pixel value (grayscale so 0 to 255)
        int pixVal = image(x,y);

        //Calculate the magnitude difference between the current pixel and its
        //neighbours
        //If any magnitude exceeds the threshold value increment a counter
        if((fabs(pixVal - Ipp) > threshold) || (fabs(pixVal - Ipc) > threshold) ||
            (fabs(pixVal - Ipn) > threshold) || (fabs(pixVal - Icp) > threshold) || (fabs(pixVal
Printed: 24/04/2007 09:46
Page 6

```

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp
- Icn) > threshold) || (fabs(pixVal - Ipn) > threshold) || (fabs(pixVal - Inc) >
threshold) || (fabs(pixVal - Inn) > threshold)) {
    //Check in top or bottom half
    if(y < heightOverTwo)
        topComplexCount++;
    else
        botComplexCount++;

    //Check in left or right half
    if(x < widthOverTwo)
        leftComplexCount++;
    else
        rightComplexCount++;
}

//Determine the greatest counter, set rotation to this and calculate confidence
if((topComplexCount > botComplexCount) && (topComplexCount > leftComplexCount) &&
(topComplexCount > rightComplexCount)) {
    complexityRotation = 180;
    complexityConfidence = (1-(botComplexCount/topComplexCount))*100;
} else if((botComplexCount > topComplexCount) && (botComplexCount > leftComplexCo
unt) && (botComplexCount > rightComplexCount)) {
    complexityRotation = 0;
    complexityConfidence = (1-(topComplexCount/botComplexCount))*100;
} else if((leftComplexCount > topComplexCount) && (leftComplexCount > botComplexC
ount) && (leftComplexCount > rightComplexCount)) {
    complexityRotation = 270;
    complexityConfidence = (1-(rightComplexCount/leftComplexCount))*100;
} else if((rightComplexCount > topComplexCount) && (rightComplexCount >
topComplexCount) && (rightComplexCount > botComplexCount)) {
    complexityRotation = 90;
    complexityConfidence = (1-(leftComplexCount/rightComplexCount))*100;
} else {
    //Store error codes if no determination
    complexityRotation = 361;
    complexityConfidence = 101;
}

/* Perform edge determination on the parameter image */
void Image::edge(Cimg_unsigned char* image, int spread) {
    //S comes in as 4 to 100, convert to 0.01 to 0.25
    float S = spread/400.0;

    //Define 3x3 neighbourhood, I
    Cimg_3x3(I,float);

    //Variables for horizontal and vertical edge counts
    float horizlines = 0;
    float vertlines = 0;

    //Loop over the image using 3x3 neighbourhood
Printed: 24/04/2007 09:46
Page 7

```

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.cpp
    cimg_map3x3(image,x,y,0,0,I) {
        //Calculate Gx and Gy using the correct neighbour positions for sobel edge
        //detection
        float gx = Ipp + Inc*2 + Inn - Ipp - Ipc*2 - Ipn;
        float gy = Ipp + Icp*2 + Ipn - Ipn - Icn*2 - Inn;

        //Find result = arctan(Gy / Gx)
        float result = atan2(gy,gx);

        //Determine if the edge direction is in one of the two ranges
        //The spread of these ranges is given by S
        if((((pi*2) < result) && (result < -(pi*(2-S)))) || ((-(pi*(1+S)) <= result
) && (result < -(pi*(1-S)))) || ((-(pi*(0+S)) <= result) && (result < pi*(0+S))) ||
((pi*(1-S) <= result) && (result < pi*(1+S))) || ((pi*(2-S) <= result) && (result <=
pi*2)))
            vertlines++;
        if((((-(pi*(1.5+S)) <= result) && (result < -(pi*(1.5-S)))) || ((-(pi*(0.5+S))
<= result) && (result < -(pi*(0.5-S)))) || ((pi*(0.5-S) <= result) && (result < pi*(0
.5+S))) || ((pi*(1.5-S) <= result) && (result < pi*(1.5+S))))
            horizlines++;
    }

    //Determine the greatest counter and so rotation determined
    if(vertlines > horizlines) {
        //rotate 0 or 180
        edgeRotation = 1;
        edgeConfidence = (1-(horizlines/vertlines))*100;
    } else if(vertlines < horizlines) {
        //rotate 90 or 270
        edgeRotation = 2;
        edgeConfidence = (1-(vertlines/horizlines))*100;
    } else {
        //error codes
        edgeRotation = 361;
        edgeConfidence = 101;
    }
}

/* Return the total confidence */
int Image::getConfidence()
{
    return confidence;
}

/* Return the total rotation */
int Image::getRotation()
{
    return rotation;
}

Printed: 24/04/2007 09:46
Page 8

```

14.3.3 Header Files

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.h
/* RotationController */

#import <Cocoa/Cocoa.h>
#include "Image.h"

/* Define controller object references */
@interface RotationController : NSObject
{
    //Interface buttons
    IBOutlet NSButton *addButton;
    IBOutlet NSButton *minusButton;
    IBOutlet NSButton *removeAllButton;
    IBOutlet NSButton *chooseButton;
    IBOutlet NSButton *processButton;
    IBOutlet NSButton *toggleInfoButton;

    //More complex interface objects
    IBOutlet NSTableView *files;
    IBOutlet NSProgressIndicator *progressInd;
    IBOutlet NSTextField *saveLocation;
    IBOutlet NSButton *saveTick;

    //Variable sliders
    IBOutlet NSSlider *segmentsSlider;
    IBOutlet NSSlider *spreadValueSlider;
    IBOutlet NSSlider *thresholdSlider;

    //Preview Drawers
    IBOutlet NSDrawer *drawer;
    IBOutlet NSView *drawerView;
    IBOutlet NSView *noSelectionView;
    IBOutlet NSView *multipleSelectionView;

    //Preview drawer componenets
    IBOutlet NSImageView *imagePreview;
    IBOutlet NSTextField *imageFileName;
    IBOutlet NSTextField *imageKind;
    IBOutlet NSTextField *imagePath;
    IBOutlet NSTextField *imageRotation;
    IBOutlet NSTextField *imageConfidence;

    //Reference to to controller itself
    IBOutlet id rotation;

    //Data structure for list of images
    NSMutableArray *records;

    //Low confidence setting variable
    int minConfidence;

    //Timer for refreshes
    NSTimer *timer;
}

```

Printed: 24/04/2007 09:45

Page 1

```

/Users/admin/Dev/3rd Year Project/Rotation/RotationController.h

/* Methods triggered by interface controls */
- (IBAction)addFiles:(id)sender;
- (IBAction)deleteFiles:(id)sender;
- (IBAction)processList:(id)sender;
- (IBAction)chooseSave:(id)sender;
- (IBAction)tickSave:(id)sender;
- (IBAction)incrementN:(id)sender;
- (IBAction)incrementT:(id)sender;
- (IBAction)incrementS:(id)sender;
- (IBAction)decrementN:(id)sender;
- (IBAction)decrementT:(id)sender;
- (IBAction)decrementS:(id)sender;
- (IBAction)decrementI:(id)sender;
- (IBAction)decrement5:(id)sender;
- (IBAction)clearFiles:(id)sender;
- (IBAction)toggleDrawer:(id)sender;

@end

```

Printed: 24/04/2007 09:45

Page 2

Interface controller header file

```

/Users/admin/Dev/3rd Year Project/Rotation/Image.h
/*
 * Image.h
 */

//Define CImg properties and include library
#include <cstdlib>
#include <iostream>
#define cimg_convert_path "/usr/local/bin/convert"
#define cimg_display_type 0
#include "Cimg.h"
using namespace cimg_library;
using namespace std;

/* Define methods and variables of the class */
class Image {
public:
    Image(const char* fileLoc);
    int processImage(int N, int T, int S, int save, const char* savePath);
    CImg<unsigned char> toGrayscale(CImg<unsigned char> image);
    int calculateRotationsAndConfidences();
    void colour(CImg<unsigned char> image, int segments);
    void complexity(CImg<unsigned char> image, int threshold);
    void edge(CImg<unsigned char> image, int spread);
    int getConfidence();
    int getRotation();
private:
    float pi;
    const char* filePath;
    int colourRotation;
    int complexityRotation;
    int edgeRotation;
    int colourConfidence;
    int complexityConfidence;
    int edgeConfidence;
    int confidence;
    int rotation;
};

```

Printed: 24/04/2007 09:46

Page 1

Image processing header file