Kyffin Williams: Digital Analysis of Paintings

Final Report for CS39440 Major Project

Author: Alexander David Brown (adb9@aber.ac.uk)
Supervisor: Hannah Dee (hmd1@aber.ac.uk)

22nd April 2012
Version: 0.0.792 (Draft)

This report was submitted as partial fulfilment of a MEng degree in Software Engineering (G601)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature ....

Date ....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature ....

Date ....

Acknowledgements

Abstract

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

Chapter 1

Background & Objectives

1.1 Sir John "Kyffin" Williams

Sir John "Kyffin" Williams (1918-2006) was a Welsh painter and printmaker, widely regarded as the defining artist of Wales during the 20th century [7]. He was advised to take up art by a doctor after failing a British Army medical examination because of an 'abnormality' (epilepsy) as something which would not tax his brain.

He studied at the Slade School of Fine Art and taught art in Highgate School, after which he retired to Anglesey until he died in 2006 after a long battle with cancer.

His most characteristic pictures are of Welsh landscapes, painted with thick layers of oil paint applied with a palette knife [4]. Most of his paintings are highly textural; to the point of being 3-dimensional.

As his life progressed Kyffin's 'abnormality' grew steadily worse, especially when exposed to bright light. As a result most of his paintings are of overcast Welsh landscapes and tend to become visibly darker over time [10]. By eye it is generally quite easy to approximate the time period in which a painting was created.

In 1969 he won a scholarship to study and paint in Y Wladfa; the Welsh settlement in Patagonia. This period of his life is very obvious from his paintings as there is a complete contrast in colour between Patagonian and Welsh landscapes.

1.2 Interdisciplinary work with the National Library of Wales

This project was initially suggested through a conversation between Hannah Dee and Gareth "Llyod" Roderick about image processing and art. Llyod is a PhD student at the National Library of Wales (NLW). Their initial idea was to try to geolocate a Kyffin painting on a map to build up a geographical representation of Kyffin's work.

Hannah started to create a prototype for performing geographical analysis, this proved to be a difficult task and one which is still being researched.

However, the nature of Kyffin's illness and painting style allows for a second form of analysis: temporal. As previously stated it is fairly easy to judge by eye a good approximation of the period in which a Kyffin painting was created. It should, therefore, follow that this process can be performed digitally.

When I started this project I was given a "database" (in reality this was just a spreadsheet) Llyod had produced, containing information of Kyffin Williams' paintings, including: title, year, category (landscape, portrait, etc.), canvas size and a few additional details which aren't so relevant to the project.

The first meeting held was between Llyod, Hannah and I, in which we discussed the current state of the project, what our aims for the project were and what form of help Llyod could provide to us. As one of the objectives of this project is to, eventually, get a paper published, the relevant details of the process we would need to go through if we wanted to do so.

The second meeting was between Hannah, Llyod, Lorna M. Hughes (Llyod's supervisor) and I. Again we discussed the state of the project. Llyod had also produced a better version of his "database" to be more machine readable and succinct. A lot of information came from this meeting;

- The "cut-off" point between early and late is around 1973.
- The size of the canvas might be a useful data point to use in classification, as Kyffin sold more paintings he would have had the money available for larger canvases and the paint for said canvas.
- It is a little dubious as to whether some dates can be trusted. One painting owned by the NLW was stated to be his last painting, but Lorna believes it was painted much earlier and claimed to be his last to improve the sale price.
- Llyod may have found date markings on some paintings. These again may not be accurate, but may prove to increase the sample size.
- It should be easy to provide a "no later than" estimate for each painting from the art historians.
- Paul (?) should be able to produce some exemplars for us as a ground truth.
- Llyod may be able to find more paintings in the hands of private collectors to increase the sample size.
- Llyod had been playing around with ImageJ to do some basic graph plotting. This might be useful to look at further to expand my own work.

There were also more detailed discussions about publications, particularly in a digital humanities journal.

1.2.1 Continuation of the Kyffin Project

There are several projects that could continue on from the Kyffin Project.

One was to use the Learning/Teaching development fund to produce a web-based front-end for of some of my analysis.

Another venture was to look into PhD funding to build up a 3D map of some of Kyffin's paintings and being able to display it (perhaps via HTML5 and WebGL) so they can explore the painting digitally how it is meant to be in real life.

1.3 Existing Work

1.3.1 Edge-Orientated Gradients

As Kyffin Williams' work is highly texturally, looking at the edge orientation of the image is likely to be a valuable technique to use.

One technique recommended by Hannah was to look at Histogram of Orientated Gradients (HOG). The suggested paper outlined the use of grids of HOG descriptors to improve the feature set for robust visual object recognition [6]. As it significantly improves the feature set it seems sensible to try and implement it as a technique without the Kyffin project to experiment with a non colour-based approach.

The approach involves quite a few separate steps, only some of which are relevant to the project:

1. Gamma and colour normalization. Grayscale, Red, Green, Blue (RGB) and L, a, b Colour Space (LAB) spaces were used. RGB and LAB give similar results. Grayscale reduced performance less that square root gamma compression, but not as much as log compression.

2. Compute gradients. Often the simplest are the better here; Gaussian smoothing followed by discrete derivative masks (e.g.: figure 1.1, figure 1.2), etc. For colour this was done for each channel, and take the one with the largest norm.

3. Spatial and Orientation binning:

   - Spatial binning is done by splitting the images into cells which can be rectangle or radial.
   - Orientation binning are spaced equally between either 0-180 ”unsigned” or 0-360 ”signed” bins.

4. Normalisation and Descriptor Blocks. Gradients vary over foreground/background, etc. Typically the blocks were overlapped so that each scalar response contains several components.

5. Pass a detector window across the image.

6. Run through a Linear Support Vector Machine (SVM) to classify the image.

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Figure 1.1: Example of a 1 by 2 Discrete Derivative Mask

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Figure 1.2: Example of a 1 by 3 Discrete Derivative Mask

Obviously, when applying this as an analysis technique to paintings, there are some points which are completely irrelevant. Passing a detector window and running through a Linear SVM are the obvious two. Normalisation is another unneeded step; computing performance isn't likely to be a large issue for this project; so long as the techniques complete within a semi-reasonable amount of time.

The leaves the act of computing the gradients (again, this can be done without Gaussian smoothing as that reduces accuracy) which should a simple matter implemented by earlier techniques. Binning by Rectangular Histogram of Orientated Gradients (R-HOG) or Circular Histogram of Orientated Gradients (C-HOG) descriptors, which may prove to the one of the more difficult parts of implementing this technique.

R-HOG descriptors have similarities to Scale-Invariant Feature Transform (SIFT) descriptors, but are used quite differently; SIFT descriptors are optimised for sparse baseline matching whilst R-HOG descriptors are optimised for the dense and robust coding of a spatial form. The size of the descriptor affects performance when using R-HOG, for paintings it may turn out that a size relating to the original size of the painting is a good way of getting around this problem.

C-HOG descriptors become more complex still. They are similar to Shape Context [**?**], but differ in one key aspect: in C-HOG descriptors each spatial cell holds a stack of gradient-weighted orientation cells over an orientation-independent edge-presence count which Shape Contexts use.

According to the author it is better to think of C-HOG descriptors as an advanced form of centre-surround coding as small descriptors with very few radial bins gave the best results.

Local contrast normalisation can be performed to help against local variations in the illumination of foreground and background.

It would seem that both R-HOG and C-HOG descriptors are designed more for the detection window rather than analysis technique. This may make them less useful and result in an implemented technique being just a simple histogram of edge orientations.

1.3.2 Brush-stroke Analysis

Stroke analysis is one of the main goals for this project. It is quite apparent from looking at Kyffin Williams' paintings that his brush-strokes change over time, his early work having lots of smaller strokes over the canvas to large bold strokes in his later work.

The first paper I found relating to the analysis of brush-strokes involved moving a circular filter across the whole painting to find the ridges of strokes, then filling any unbroken areas. They then shrunk these areas to a single pixel line and fitted a n^{th} order polynomial to this line [2]. This method seems fairly simplistic, but could be an interesting first step, but as it is more focused on authenticating paintings it may be of limited use.

Another method for stroke analysis has been published in the IEEE Transactions on Pattern Analysis and Machine Learning journal. This method is far more complex, but is able to extract and label individual brush-strokes. An interesting part of their findings was the ability to date some of Van Gogh's paintings to a known period in his career [12].

This method involves performing edge detection of the painting followed by an edge linking algorithm which aims to remove small, noisy edges and to trace every edge. With this they then perform enclosing, as strokes may not be complete this stage also aims to fill in missing gaps of strokes and to fill these in within a certain tolerance.

The algorithm then decides if a stroke really is a painted stroke, if the stroke is completely enclosed, isolated from other non-edge pixels and forms a connected component then it is likely that it is a proper brush-stroke and is extracted. The edge pixels are used as the background and the non-edge pixels as the foreground, this is the process of labelling the brush-stroke.

For each of these labelled candidates, a heuristic function is used to threshold any brush-strokes that are either too long or too short, these strokes are discarded. These strokes are then considered to be candidates if they are not significantly branched, the stroke is not too wide (this may change for Kyffin Williams as he used a pallet knife rather than a brush) and the brush-stroke is not too big or small.

Separately, the image is then segmented using k-means clustering by RGB values. This clustering algorithm is applied several times, lowering the tolerances for distance within a cluster. Connected components as a result of this clustering and have noise reduction performed upon them. Finally, the two types of brush-strokes are combined.

This technique may need some changing to account for Kyffin Williams' use of a pallet knife, but the overall principals of this technique should work with Kyffin's paintings.

1.4 Analysis Objectives

Analysis is one of the biggest sections of this project and involves creating techniques which will allow comparison of paintings in a way which will allow some form of classification to be performed on them.

Typically I would expect this to produce some form of high-dimension state space in which each painting is a point in the state space. From this state space the distance between one painting and another can be easily resolved using a distance measure like Manhattan distance (1), euclidean distance (2) or a distance measure more specific to the state space should it be needed (e.g.: chi-squared for histograms).

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=0}^{n} |p_i - q_i| \tag{1}$$

$$d_1(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=0}^{n} (q_i - p_i)^2} \tag{2}$$

1.4.1 Colour-space Analysis

The simplest way of analysing a digital image is to look at the colours which it consists of. Doing this is relatively simple; each pixel has a set of values defining the colour of that point, getting something meaningful from this is less simple.

The simplest strategy is to perform some form of statistical analysis on each painting then use this for classification. Several good and computationally cheap options exist for this; mean (3) and standard deviation (4), are some good examples which often come predefined in image processing and computer vision libraries.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2} \tag{4}$$

The representation of colour is another important factor, an RGB representation will have all three values change if there are many changes in brightness of the colours whilst a Hue, Saturation, Value (HSV) representation will only have a single value change.

Therefore, an object of this section should be to explore different colour models and statistical methods which can be applied to them.

Another useful technique which should be investigated early into the project are image histograms. These histograms plot the distribution of colour across an image and are therefore a very powerful method of analysing an image, especially for comparison. As with statistical analysis, histograms will be largely effective by colour model.

1.4.2 Texture Analysis

As Kyffin Williams' work is very textural, it follows that a main part of the analysis should focus around the texture of his paintings. Unfortunately for this section, it seems unlikely that I will be able to get any 3-dimensional models of Kyffin's paintings. This would have been a nice, if rather large, section of the project.

Instead it is more sensible to look at the orientation of edges in Kyffin's work. Some useful pre-existing techniques have already been discussed in section 1.3.1. Histograms of edge orientation [6] seem like a promising concept which may prove relatively simple to implement.

This section may also help with any work into brush-stroke analysis (see section 1.4.3).

1.4.3 Brush-stroke Analysis

With Kyffin's distinctive style and how obviously this style changes over time, the ultimate aim of this project is to be able to analyse the brush-strokes[1] in a painting.

From looking at the paintings it is very apparent that in his earlier work he made a lot more strokes than in his later works[2]. The strokes in his later work tend to have larger areas and span more of the canvas.

If it is possible to calculate a rough amount and size of strokes made in a given painting it should be a reasonable piece of data to classify on. As previously discussed in section 1.3.2 there has already been a decent amount of research into determining brush-strokes in a painting.

It would be preferable to try and take one of the techniques discussed in that research and change it to suit the needs of the project rather than attempting to create a whole new method of brush-stroke recognition.

1.4.4 Ensemble Techniques

With some of the aforementioned analysis techniques it makes sense to combine two or more techniques together; a good example would be colour histograms and histograms of edge orientation.

This form of analysis is inspired by the concept of the same name in statistics and machine learning which tend to obtain better predictive performance. It may also be worth while trying to weight different techniques so that the techniques which give the best performance affect the result of the ensemble technique more.

1.5 Classification Objectives

The overall objective of classification is to be able to label a painting by Kyffin Williams as being painted in a given year based on analysis performed on all other paintings with known years.

[1] A slight misnomer as Kyffin used a palette knife to paint with rather than a traditional brush

[2] Although this isn't quite true as the canvases he worked on in his later life tended to be larger

This ties in with the main aim of this project of being able to classify any Kyffin Williams painting, whether it has a known or unknown year, as being from a given year. Evidently for paintings with an unknown year it is difficult to know how accurately the system has been, so, for the most part, these paintings have been ignored and those paintings with a known year have made up the training and validation set.

Because of the small size of paintings with known years it should be computationally viable to perform leave-one-out cross validation (figure 1.3).

```
function LOOCV($data$)                                   ▷ $data$ is a set of all data points
    for all $item \in data$ do
        $classified_{item} \leftarrow$ CLASSIFY($item, data \setminus \{item\}$)
    end for
return $classified$
end function
```

Figure 1.3: Pseudocode for Leave-One-Out Cross Validation

This can be used to evaluate the performance of the analysis technique and classification algorithm. Pearson's product-moment correlation coefficient (5) between actual year and classified year has been suggested to be a good performance measure for this project.

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{5}$$

1.5.1 Classification

One of the simplest methods of classification is k-Nearest Neighbour (figure 1.4) from this one can take a poll of the years for each neighbour and assign the year of the painting to classify to be the average of these years.

Depending which form of average you take (mathematical mean (3), median or mode) will alter the result; although it should be noted that median is very unlikely to give a result on its own due to the sparseness of the data.

```
Require: $0 < k \le |data|$                              ▷ $data$ is a set of all data points
  function KNEARESTNEIGHBOUR($k, data$)
      for $i = 1 \rightarrow k$ do
          $nn_i \leftarrow$ NEAREST($data$)
          $data \leftarrow data \setminus \{nn_i\}$
          $i = i + 1$
      end for
  return $nn$
  end function
```

Figure 1.4: Pseudocode for k-Nearest Neighbour

There are other techniques which could be applied to this problem, but the rewards for implementing them is not likely to be outweighed by the time it would take to implement such techniques. There is a workaround for this; there are several machine learning tool-kits which provide pre-implemented version of these techniques.

One of the most popular tool-kits available for general use is Weka [9], which is discussed in more detail in section 1.5.1.1.

Another technique suggested by Julie Greensmith is to use Learning Classifier Systems (LCS) [1], which has an implementation for Weka. This may prove to give very good results for the kind of analysis being performed on Kyffin's work.

1.5.1.1 Use of Weka

1.5.2 Exemplars

The use of exemplar images would be another way of performing classification. The idea of an exemplar is that a painting is the most representative of a given time period. With the help of Llyod, Lorna and the NLW a list of exemplars which can be used as a ground truth to classify against has been produced.

The initial idea for digitally producing exemplars is to take the middle painting for a time period, as would be expected. These can then be compared to the ground truths to see how correct the analysis technique performed.

However, there is also the potentially to generate a theoretical exemplar from the analysis. This might be hard to perform validation against the ground truth upon, but will give some useful data on Kyffin Williams' style and how it changed.

These theoretical exemplars would likely be produced using some form of Gaussian mixture model.

Chapter 2

Development Process

2.1 Introduction

2.2 Modifications

Chapter 3

Design

3.1 Overall Architecture

The basic architecture for any system like this is to load the data in from a source of some form, apply an analysis technique to each data point then pass this data into the classification system.

From the classification system you should then be able to get the classified and actual year for each data point which can then have validation performed on it. This architecture is summed up in figure 3.1.

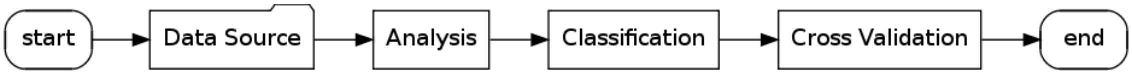

Figure 3.1: Basic Overall Architecture

Building up from this it is apparent that to implement the analysis and classification steps that there is a need to implement the factory method design pattern [8, p. 93-100]. Reading from a data source should be a simple matter of reading from a file, and cross validation has already been decided to use leave-one-out cross validation.

Figure 3.2 shows the design after adding in the factory methods.

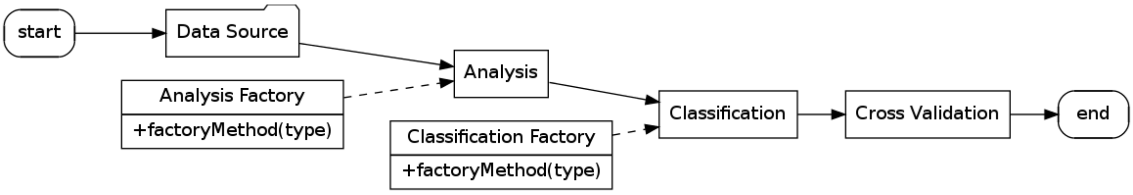

Figure 3.2: Overall Architecture with Factory Methods

From this we then need the two top-level interfaces `Analyser` and `Classifier`. The `Analyser` interface should have a single method which runs analysis on a painting and return some form of object which represents the analysed data.

The `Classifier` class should have a method which takes a single painting and a set of paintings, returning a year which is the classified year of the single painting based on the set of paintings.

At this point it is also required that there is a class to store meta-data of a painting. Figure 3.3 depicts the design after adding in these parts.

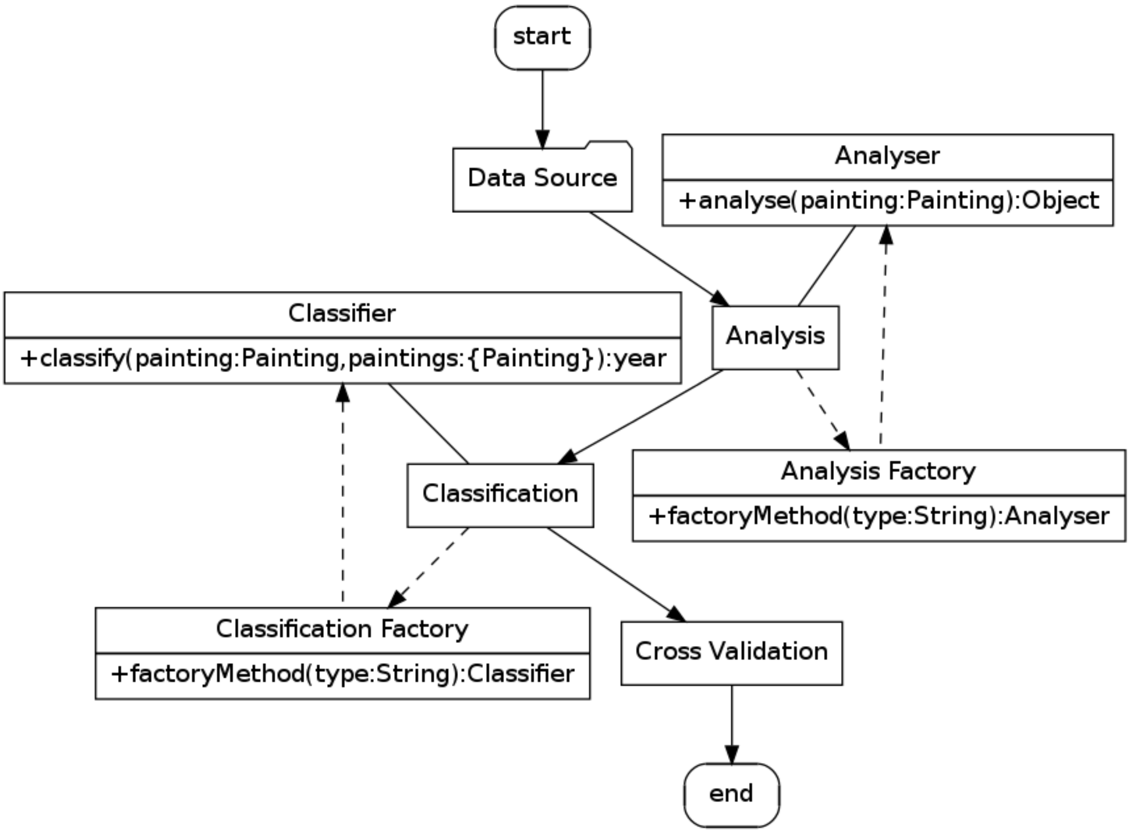
start
Data Source
Analyser
+analyse(painting:Painting):Object
Classifier
+classify(painting:Painting,paintings:{Painting}):year
Analysis
Classification
Analysis Factory
+factoryMethod(type:String):Analyser
Classification Factory
+factoryMethod(type:String):Classifier
Cross Validation
end

Figure 3.3: Overall architecture with Interfaces for Analysers and Classifiers

Chapter 4

Implementation

4.1 Colour Space Analysis

Colour space analysis involves performing statistical analysis on different colour models (RGB, HSV, etc.). This gives a very simplistic view of the entire image.

OpenCV offers the `Avg` method to perform the average across the image, however with a further look into the documentation there is also the `AvgStd` method which performs both mean and standard deviation on an image.

The analysed data was just the tuple returned by the `AvgStd` method. The distance measure was defined to be the sum of all elements in the tuple (in the case of an RGB colour model the mean red, green and blue and the standard deviation of red, green and blue).

4.1.1 Colour Models

There are many colour models to consider with digital image processing. RGB is one of the better know colour spaces as it is often how images are captured. It does have a problem in that all three values can change when the brightness changes.

As one of the main principals of this project is that Kyffin Williams' work darkened over time, it should follow that RGB may not be the best colour model to use.

To account for this it was decided to also use a HSV colour model to compare and contrast to RGB.

OpenCV handles colour spaces slightly oddly. Initially it uses `LoadImageM` to load the image, which uses an integer argument as a flag to define whether the image should be loaded in colour or grayscale.

From this image you then can use `CvtColor` to convert the colour model of an image, which uses an integer argument as a flag to define a number of different colour spaces.

Once converted, all methods act exactly the same as they would on a RGB image.

4.1.2 Colour Histograms

4.2 Texture Analysis

4.2.1 Edge Orientation

4.2.1.1 Histogram of Edge Orientation

4.3 Brush-Stroke Analysis

4.4 Classification and Validation

4.4.1 K-Nearest Neighbour

4.4.2 Leave-One-Out Cross Validation

4.4.3 Weka 3

4.4.3.1 Attribute-Relation File Format (ARFF)

4.4.4 Exemplars

4.4.4.1 Nearest Exemplar Classification

To implement Nearest Exemplar Classification was a fairly easy task: Llyod (with help from members of the NLW) provided a secondary spreadsheet which contained all the necessary information of exemplar by year (see table D.1 for the full document).

The spreadsheet was arranged in the format described in table 4.1, from there it was a simple matter of saving the spreadsheet as a Comma-Separated Values (CSV) file and taking some of the existing code for parsing CSV files. This caused a slight problem in that the parsed data didn't have enough information to create a full `Painting` object, yet all the analysis techniques worked from these objects.

Filename	ID	Title	Catalogue Entry	Year
154.jpg	154	Landscape at Llanaelhaearn	1947	1947
etc.				

Table 4.1: Layout of the Exemplar Spreadsheet

This was solved easily thanks to Python's dynamic typing. A simple class which implemented all the necessary elements of `Painting` could be passed to the analysis techniques without any complaints. With a statically typed language this would have been harder to complete, but there would have been ways around using sub-classes and so on.

With the exemplars loaded and analysed, the program could continue as normal, until the classification step.

The idea of Nearest Exemplar Classification is to classify the unknown example using the nearest exemplar to that example in the feature space. This acts as a k-Nearest Neighbour with $k = 1$ and the space of neighbours only including the exemplars, rather than every other example. The psuedocode for this is shown in figure 4.1.

Initially this was implemented so that the examples that were exemplars were also classified, but this is a pointless exercise which only skews the results. Additional logic was added to skip any example which was an exemplar itself.

```
function NEARESTEXEMPLARCLASSIFICATION($example$, $exemplars$)
    if $example \in exemplars$ then
        return $\varnothing$
    end if
    return NEAREST($example, exemplars$)
end function
```

Figure 4.1: Nearest Exemplar Classification Psuedocode

This proved to give slightly worse correlation per technique than k-Nearest Neighbour. This result is to be expected; for a start an artistically classified exemplar is unlikely to be the same as a statistically classified exemplar (see section 4.4.5). Also, with fewer examples to classify against, any variance in the data set (of which there is a lot) will likely be magnified.

Lastly, a painting may be picked as an exemplar by an expert for different reasons than any analysis technique that currently exists can give; emotional connections and knowledge of the artists history can be very subjective and may not relate to anything put down in paint.

4.4.5 Statistically Classified Exemplars

Another approach to exemplars is to work out a theoretical exemplar for a given period; the centroid of paintings within the given feature space for a single year, for example.

The simplest way of working this out is showing in figure 4.2, this works by taking the mean (equation 3) of each feature in the set of paintings for a single year, this will give the point in feature space that is most central. This is the same technique used to generate centroids in a clustering algorithm (k-Means Clustering, for example).

```
function STATISTICALCLASSIFYEXEMPLAR($examples$)
    for all $example \in examples$ do
        for all $feature \in example_{features}$ do
            $average_{feature} \leftarrow average_{feature} + example_{feature}$
        end for
    end for
    for all $feature \in average$ do
        $average_{feature} \leftarrow \frac{average_{feature}}{\text{LENGTH}(examples)}$
    end for
    return $average$
end function
```

Figure 4.2: Psuedocode for generating a Statistically Classified Exemplar

This may become a long operation depending on how many dimensions the feature space has. A technique like Principal Component Analysis (PCA) may be useful to help cut down the number of dimensions needed that this algorithm uses.

4.5 3rd Party Libraries and Tools

4.5.1 Python

4.5.1.1 Python setuptools

4.5.2 OpenCV

4.5.3 scipy & numpy

4.5.4 matplotlib

4.5.5 Weka 3

4.5.5.1 liac-arff

4.5.6 git & github

Chapter 5

Testing

5.1 Overall Approach to Testing

As this is a research-based project, it is difficult to perform any for of unit, functional or requirements testing. Most testing is based on validating the results of analysis and classification techniques.

5.2 Validation

5.2.1 Leave-One-Out Cross Validation

5.2.2 Validation using Weka

Chapter 6

Evaluation

6.1 Evaluation of Requirements

6.2 Evaluation of Design

6.3 Evaluation of Tools

6.3.1 Programming Language

6.3.1.1 Dependency Management

6.3.2 Image Processing/Computer Vision Libraries

6.3.3 Machine Learning Libraries

6.3.3.1 File Formats

6.3.4 Scientific and Numeric Libraries

Appendices

Appendix A

3rd Party Libraries and Tools

1.1 Python 2.7

1.1.1 setuptools

`http://pypi.python.org/pypi/setuptools`

1.1.2 scipy

`http://www.scipy.org/` [11]

1.1.3 numpy

`http://www.numpy.org/` [11]

1.1.4 matplotlib

`http://matplotlib.org/`

1.1.5 liac-arff

`https://github.com/renatopp/liac-arff`

1.2 OpenCV

`http://opencv.org/` [3]

1.2.1 OpenCV Python

`http://opencv.willowgarage.com/wiki/PythonInterface` [3]

1.3 Weka 3

`http://www.cs.waikato.ac.nz/ml/weka/` [9]

1.4 git

1.4.1 github

Appendix B

Equations

2.1 Statistical Equations

2.1.1 Mean

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

2.1.2 Standard Deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

2.1.3 Pearson's product-moment coefficient

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

2.2 Distance Equations

2.2.1 Manhattan Distance

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=0}^{n} |p_i - q_i|$$

2.2.2 Euclidean Distance

$$d_1(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=0}^{n}(q_i - p_i)^2}$$

2.3 Filter Equations

2.3.1 Gradient Direction

$$\theta = \operatorname{atan2}\left(\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}\right)$$

2.3.2 Discrete Derivative Masks

$$\begin{bmatrix}-1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1\\0\\1\end{bmatrix}$$

2.3.3 Gabor Filter

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right) \tag{1}$$

where:

$$\begin{aligned}x' &= x\cos\theta + y\sin\theta\\ y' &= x\sin\theta + y\cos\theta\end{aligned}$$

Appendix C

Code Samples

3.1 Gabor Filter Example Implementation

Listing C.1: Example implementation of a Gabor Filter in MATLAB from wikipedia [5]

```
function gb=gabor_fn(sigma,theta,lambda,psi,gamma)

sigma_x = sigma;
sigma_y = sigma/gamma;

% Bounding box
nstds = 3;
xmax = max(abs(nstds*sigma_x*cos(theta)),abs(nstds*sigma_y*sin(
    theta)));
xmax = ceil(max(1,xmax));
ymax = max(abs(nstds*sigma_x*sin(theta)),abs(nstds*sigma_y*cos(
    theta)));
ymax = ceil(max(1,ymax));
xmin = -xmax; ymin = -ymax;
[x,y] = meshgrid(xmin:xmax,ymin:ymax);

% Rotation
x_theta=x*cos(theta)+y*sin(theta);
y_theta=-x*sin(theta)+y*cos(theta);

gb= exp(-.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*cos(2*
    pi/lambda*x_theta+psi);
```

3.2 OpenCV Histogram Example Code

Listing C.2: Example Histogram calculation and displaying code from OpenCV [3].

```
# Taken from: http://opencv.willowgarage.com/documentation/
    python/imgproc_histograms.html#calchist
```

```
# Calculating and displaying 2D Hue-Saturation histogram of a
   color image

import sys
import cv

def hs_histogram(src):
    # Convert to HSV
    hsv = cv.CreateImage(cv.GetSize(src), 8, 3)
    cv.CvtColor(src, hsv, cv.CV_BGR2HSV)

    # Extract the H and S planes
    h_plane = cv.CreateMat(src.rows, src.cols, cv.CV_8UC1)
    s_plane = cv.CreateMat(src.rows, src.cols, cv.CV_8UC1)
    cv.Split(hsv, h_plane, s_plane, None, None)
    planes = [h_plane, s_plane]

    h_bins = 30
    s_bins = 32
    hist_size = [h_bins, s_bins]
    # hue varies from 0 (~0 deg red) to 180 (~360 deg red again
       */
    h_ranges = [0, 180]
    # saturation varies from 0 (black-gray-white) to
    # 255 (pure spectrum color)
    s_ranges = [0, 255]
    ranges = [h_ranges, s_ranges]
    scale = 10
    hist = cv.CreateHist([h_bins, s_bins], cv.CV_HIST_ARRAY,
       ranges, 1)
    cv.CalcHist([cv.GetImage(i) for i in planes], hist)
    (_, max_value, _, _) = cv.GetMinMaxHistValue(hist)

    hist_img = cv.CreateImage((h_bins*scale, s_bins*scale), 8,
       3)

    for h in range(h_bins):
        for s in range(s_bins):
            bin_val = cv.QueryHistValue_2D(hist, h, s)
            intensity = cv.Round(bin_val * 255 / max_value)
            cv.Rectangle(hist_img,
                         (h*scale, s*scale),
                         ((h+1)*scale - 1, (s+1)*scale - 1),
                         cv.RGB(intensity, intensity, intensity)
                            ,
                         cv.CV_FILLED)
    return hist_img
```

```
if __name__ == '__main__':
    src = cv.LoadImageM(sys.argv[1])
    cv.NamedWindow("Source", 1)
    cv.ShowImage("Source", src)

    cv.NamedWindow("H-S␣Histogram", 1)
    cv.ShowImage("H-S␣Histogram", hs_histogram(src))

    cv.WaitKey(0)
```

Appendix D

Spreadsheet Data

Filename	ID	Title	Catalogue entry BBC YP	exemplar year
154.jpg	154	Landscape at Llanaelhaearn	1947	1947
258.jpg	258	Snowdon, the Traeth and the Frightened Horse	1948	1948
155.jpg	155	Landscape with Cattle	c.1950	1950
286.jpg	286	The Dark Lake	1951	1951
288.jpg	288	The Moelwyns from Aberglasyn	c.1952	1952
238.jpg	238	Self Portrait	c.1953	1953
051.jpg	51	Cottages, Llanddona	c.1955	1955
185.jpg	185	Mountain Landscape	1957	1957
023.jpg	23	Capel Carmel	c.1958	1958
246.jpg	246	Snow above Beddgelert	c.1959	1959
206.jpg	206	Nant Ffrancon from Llandegfan	c.1960	1960
260.jpg	260	Snowstorm off Penmon	1961	1961
053.jpg	53	Cottages, Mynydd Bodafon	c.1962	1962
120.jpg	120	German Girl	c.1963	1963
243.jpg	243	Sir David Hughes Parry	1964	1964
242.jpg	242	Sir Charles Evans	c.1965	1965
087.jpg	87	Farm below Crib Goch	1967	1967
248.jpg	248	Snow on Siabod	c.1968	1968
158.jpg	158	Lle Cul, Patagonia	1969	1969
073.jpg	73	Deserted Farm, Llanrhuddlad	c.1970	1970
275.jpg	275	Sun and Cloud on Lliwedd	1973	1973
093.jpg	93	Farm, Llanfairynghornwy	c.1975	1975
018.jpg	18	Blaen Nant	1976	1976
237.jpg	237	Sea at Trearddur	c.1976	1976
017.jpg	17	Blaen Ffrancon No.1	c.1978	1978
107.jpg	107	Farmers on the Carneddau	c.1980	1980
105.jpg	105	Farmer below the Ridge	c.1983	1983
202.jpg	202	Mrs Rowlands	c.1984	1984
003.jpg	3	Above Carneddi, No.2	c.1985	1985
264.jpg	264	Storm at Trearddur	1987	1987
165.jpg	165	Llyn-y-Cau, Cader Idris	c.1990	1990
267.jpg	267	Storm, Porth Cwyfan	1995	1995
268.jpg	268	Storm, Trearddur	1996	1996
214.jpg	214	Pengwyryd	c.1999	1999
020.jpg	20	Bryn Cader Faner	2000	2000
278.jpg	278	Sunset, Anglesey	2004	2004

Table D.1: Exemplar Spreadsheet Data

Annotated Bibliography

[1] J. Bacardit and X. Llorà, "Large-scale data mining using genetics-based machine learning," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 1, pp. 37–61, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1002/widm.1078

> A paper recommended by Julie Greensmith for information on a Learning Classifier System (LCS) which Julie believes will yield good results with the Kyffin Williams project.

[2] I. E. Berezhnoy, E. O. Postma, and H. J. van den Herik, "Authentic: Computerized brushstroke analysis," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on.* IEEE, July 2005, pp. 1586–1588. [Online]. Available: http://dx.doi.org/10.1109/icme.2005.1521739

> Defines a method of analysing brushstrokes by applying a circular filter across a digital image to pick up the ridges of a brushstroke. This can then be used to pick out individual brushstrokes in order to be able to fit a nth order polynomial to them. Though this paper focuses on authenticating Van Gogh's paintings, it could easily be applied to the work of Kyffin Williams and may allow for some interesting analysis.

[3] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

> Used Python (http://opencv.willowgarage.com/documentation/python) and C++ (http://opencv.willowgarage.com/documentation) documentation for library reference and some learning on image processing/computer vision. Used since 11 October 2012.

[4] I. Chilvers, J. Glaves-Smith, and I. Chilvers, *A dictionary of modern and contemporary art.* Oxford University Press, 2009. [Online]. Available: http://www.worldcat.org/isbn/0199239665 0199239665.

[5] W. Contributors, "Gabor filter," Online, Oct. 2012. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Gabor_filter&#38;oldid=517342109

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, ser. CVPR '05, vol. 1. Washington, DC, USA: IEEE, June 2005, pp. 886–893 vol. 1. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2005.177

> Describes a method of producing histograms of edge orientations which may prove to be a useful analysis technique for Kyffin Williams' art. The most interesting part of this paper is the use of segmentation and binning of gradients

which seems like it could be useful to differentiate different parts of the image which may be painted in different styles.

[7] J. Davies and A. Gymreig, *The Welsh Academy encyclopaedia of Wales*. University of Wales Press, 2008, pp. 957–958. [Online]. Available: http://www.worldcat.org/isbn/9780708319536 9780708319536.

[8] E. Gamma, *Entwurfsmuster : Elemente wiederverwendbarer objektorientierter Software*, ser. Addison-Wesley professional computing series. Addison-Wesley, 1996. [Online]. Available: http://www.worldcat.org/isbn/9780201633610 9780201633610.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://dx.doi.org/10.1145/1656274.1656278

Citation for the Weka data mining software. Weka is a Java based tool which can be used to run a lot of classifiers to a dataset, making it a very useful tool to apply to the Kyffin Williams project. Weka allows the application of complex machine learning techniques without having to spend a lot of time learning, understand and implementing said techniques.

[10] R. Harris, "How rolf learnt to paint like sir kyffin williams," BBC Broadcast, Feb. 2011. [Online]. Available: http://www.bbc.co.uk/programmes/p00f6nyt

A video on the BBC by Rolf Harris about some of Kyffin Williams' life and about his interesting style of painting.

[11] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for python," 2001.

[12] J. Li, L. Yao, E. Hendriks, and J. Z. Wang, "Rhythmic brushstrokes distinguish van gogh from his contemporaries: Findings via automated brushstroke extraction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 6, pp. 1159–1176, June 2012. [Online]. Available: http://dx.doi.org/10.1109/tpami.2011.203

Defines a complex method for analysing individual brush strokes which has been used to classify the period of two paintings by Van Gogh. This technique could be very powerful when applied to Kyffin Williams' work. This could be one of the most important techniques for the whole of the Kyffin Williams project.