Kyffin Williams: Digital Analysis of Paintings

Report Name	Progress Report
Author (User Id)	Alexander D. Brown (adb9)
Supervisor (User Id)	Hannah M. Dee (hmd1)
Module	CS39440
Date	November 18, 2012
Revision	0.4
Status	Draft
Word Count	3615

Contents

List of Figures

List of Tables

1 Project Summary

Sir John "Kyffin" Williams was a landscape painter from Wales whose work was predominantly based in Wales and Patagonia. He studied at the Slade, one of Britain's top art schools, after epilepsy ended his career in the army during the second world war. This epilepsy made Kyffin Williams sensitive to light and is the reason his work gets darker over time [3].

Gareth Lloyd Roderick, a PhD student in the National Library of Wales, has collected data such as the date or location, of these paintings. This data allows for some interesting analysis; particularly that of temporal or geographical classification of a given painting. That is being able to take a painting and decide the year or location in which it was painted from a database of existing, known, works by Kyffin Williams.

Temporal analysis will be the focus of this project as it allows for a diverse range of techniques; from statistical analysis of colour values of the paintings to looking at the length and style of paintbrush strokes. Geographical analysis would likely be very difficult, especially as the locations depicted were often sketched on-site then painted in a studio.

Whilst it would be nice to be able to predict the age of a painting with no known year, it is far more interesting to try to guess the year of paintings for which the date is known. This project will use leave-one-out cross-validation to help measure the effectiveness and validity of the analysis techniques employed in this project. Leave-one-out validation can be used with this project as the data set is small enough not to incur large performance overheads and the overall speed of the program is irrelevant so long as it completes within a decent amount of time.

One major limitation with this is it also includes the technique used for classification, some techniques might work better with K-Nearest Neighbour whilst other might benefit from more complex methods of classification. This means I will either have to stick with a single classification algorithm and hope it's a good one for all techniques. Or I could find the best machine learning technique for each individual analysis technique, then perform comparison. This does assume that the best machine learning technique for every analysis technique exists within the scope of the project.

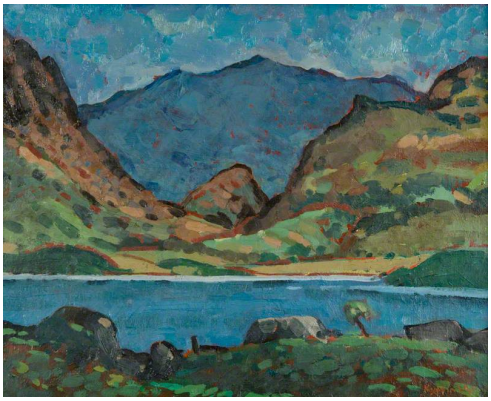
As there has been a lot of research into this topic there is an introductory paper to the literature [9], this paper specifies a lot of useful papers, including papers like the AUTHENTIC project [**?**].

Some examples of Kyffin Williams' work show the changes in his style over time; his early work as shown in figure 1 often contained a lot of strokes and quite a lot of bright colour. Comparing it to the work in around the middle of his career show in figure 2, there are visible differences; the colours used are a lot darker and the strokes on the canvas are a lot more confident, leading to a more "blocky" feel. His later work, shown in figure 3 retains this "blocky" feel and the darker colours, but has a lot more detail like some of the paintings from his earlier period.

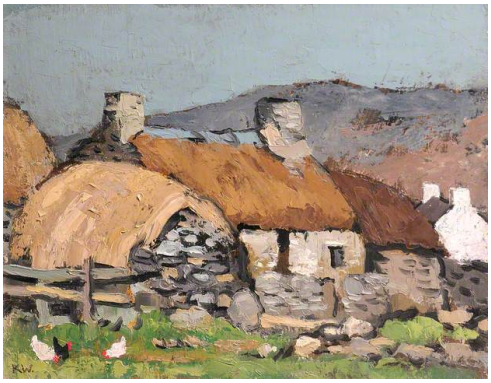
There is also a period in 1969 where Kyffin Williams visited Patagonia and did quite a few works in that area; these are visually very distinct as they use a lot of orange colours not common to Welsh landscapes. Some of this work is shown in figure 4.

At current I have had one meeting with both Lloyd and Hannah to discuss the current progress of the project and where we see the project progressing to. Lloyd was impressed by the current state of the project and was looking forward to see where it was leading to.

We also discussed the potential of getting a paper published out of this project.



(a) Snowdon from Llyn Nantle (1945)



(b) Swtan (1947)

Figure 1: Early works by Kyffin Williams

(a) Nant Ffrancon from Llandegfan (1970)

(b) Farm, Llanfairynghornwy (1974)

Figure 2: Works by Kyffin Williams in the middle of his career

(a) Pengwyryd (1999)

(b) Sunset, Anglesey (2004)

Figure 3: Works by Kyffin Williams towards the end of his career

(a) Paith, Patagonia (1969) (b) Patagonian Landscape (1969)

Figure 4: Works by Kyffin Williams from his trip to Patagonia

2 Current Progress

At current, there are three main areas in which progress has been made:

- Research into tools.
- Research into related work.
- Progress in development and experimentation.

2.1 Research into tools

2.1.1 Library and Language Research

The first step in the project was to research into the numerous image processing/computer vision libraries available for use with the aim to find the most suitable library to use constantly for the rest of the project. Because of this the library had to be easy to install and use, as well as having all the necessary features which would be used in this project.

The choice of library also has a knock-on affect on the choice of language this project would be written in. To this end I downloaded each of the libraries in turn, created a simple application to perform blur on a single image to test their use and documentation. From this I gained a good insight into how easy that library would be to perform more complex tasks and how confident I felt with the language(s) the library has bindings for.

Table 1 shows an overview of all the libraries I have currently considered and experimented with.

Library	**Platform**				**Language(s)**	**Example**
	Windows	Mac	Linux	Android		
OpenCV	✓	✓	✓	✓	C, C++, Python	Figure 5
FIJI	✓	✓	✓		Java	Figure 6
IVT	✓	✓	✓		C++	Figure 7

Table 1: Comparison of image processing/computer vision libraries.

Of these libraries OpenCV was the easiest to work with. OpenCV also boasts a wide range of features, all of which are well documented. FIJI provided a lot of high-level functionality, but for use as a library it quickly became unwieldy and was difficult to find the correct classes just for the simple task of blurring an image. On a side note, I only managed to get the blurring outputting a greyscale image in the short period of time I spent using FIJI.

IVT was somewhat similar to FIJI in that it had a good range of high-level features, but was less impressive as a library. Despite following the example code I struggled to compile my own example and eventually gave up trying to get a working binary due to time constraints.

Having weighed up these libraries I it became fairly apparent that OpenCV would be the best choice, not only did it act as an easy to use library, it is also seems that is is one of the most prevalent libraries for Computer Vision. For the Kyffin Williams project OpenCV provides a lot of pre-built helpers which allow for very rapid production of the early elements of the project. For example it is able to handle loading images in different colour spaces, generating histograms of images and even comes with its own machine learning libraries.

Having decided to go with OpenCV I was then faced with a choice of languages. OpenCV runs natively with C++ and has bindings for Python. Of these two languages I am slightly more familiar with C++ and I am also used to the syntax from programming Java continually for three years.

However, I wanted to take a rapid prototyping approach with this project; constantly adding new modules each week to build up a better and better system. C++ seemed too heavyweight for this approach, whilst Python seemed designed for it.

Another consideration I had was that the results of any analysis could take any form at all, from an array of numbers to a complex data structure returned by OpenCV. Trying to work around this with a statically typed language such as C++ could end up being difficult, especially when trying to keep a object-orientated approach. With a dynamically typed language such as Python it's a lot easier to pass these sorts of objects so long as the functions you pass to are expecting the right sort of object.

The Python OpenCV bindings also allow a powerful scientific library named numpy to be used for matrix manipulation, complex maths functions. Another related library I am using is

matplotlib, a library for plotting graphs and charts, which is designed to be used in conjunction with numpy. These two libraries together should help with some of the more complex elements of the project.

There may be some parts of the project which I might consider writing in C++ after prototyping in Python. An example of this would be if I wrote an algorithm which located strokes across the painting, as it may be used in other projects, not just my own.

2.1.2 Other Tools

Having used git for several personal projects I was keen to continue using it for my version control system. GitHub was a convenient hosting company to go with as they provide students with free private repositories and have very high uptime. It also means I am able to easily work on any system without difficulty but with the assurance of security. This is preferable to hosting my own repositories and running the risk of losing a lot of work.

Git is one of the nicer version control programs I have used and has some nice extensions (shell integration with zsh, git-flow for helping with the development process) and its decentralised nature allows me to easily work where-ever I feel most comfortable.

2.2 Technical Issues

Having engineered software before I know that the biggest issue with developing a piece of software is working with external APIs. With this project I can currently get away with only having to work with a few APIs; OpenCV being the main candidate. I have had some struggles trying to understand exactly what the API wants as inputs and returns as outputs, but that is more due to a slight lack of knowledge in the area of computer vision and image processing rather than the API being complex.

I am currently reading up on image processing and computer vision, using free, online resources [5] and knowledge within the department.

OpenCV seems like a good library to work with, though I have yet to use it for any specialised image processing (i.e. I have only used built-in methods OpenCV provides by default). I currently don't see any issue as the python bindings for OpenCV use numpy in the background. Both of these libraries are widely used and open source. If I do happen to have any problems with either of these libraries there is a substantial community behind them and the option to fix any problems and give them back to the projects.

Another technical issue associated with this project is the resolution of the paintings in the current database. Most of the images are fairly small as they are intended to be used on a website rather than in an image processing application. There is potential that I may be able to get a hold of larger images for certain paintings, but if the techniques could be made to work on the current size of image it would probably be better for the project as a whole.

A problem associated with this is memory limits; for smaller images only a small amount of data will need to be stored, but for more complex techniques, large matrices may be generated and memory might become a precious resource. At current all analysis is run before classification but it may be necessary to only hold the analysis data for the current example to classify and the training example for which we are currently measuring the distance to. This would not be a difficult change to make, but may slow the program down as a whole.

```
import cv

im = cv.LoadImageM("tux.png")
blur = cv.CreateMat(im.rows, im.cols, im.type)
cv.Smooth(im, blur, cv.CV_BLUR, 9)
cv.SaveImage("tux_blurred_opencv.png", blur)
```

Figure 5: Using OpenCV to blur an image

```
package fiji;

import ij.IJ;
import ij.io.FileSaver;
import imagescience.feature.Smoother;
import imagescience.image.ColorImage;
import imagescience.image.Image;

public final class Blur {
        public static final String IMG_PATH = "tux.png";
        public static final String OUTPUT_PATH = "tux_fiji.png";

        public static void main(String[] args) {
                final Image image = new ColorImage(IJ.openImage(IMG_PATH));
                final Smoother s = new Smoother();
                final Image blurred = s.gauss(image, 9f);
                final FileSaver fs = new FileSaver(blurred.imageplus());
                fs.saveAsPng(OUTPUT_PATH);
        }
}
```

Figure 6: Using FIJI to blur an image

```
#include <stdio.h>
#include "Image/ByteImage.h"
#include "Image/ImageProcessor.h"

int main(int argc, char **argv) {
        CByteImage in;
        CByteImage out;
        char* from = "tux.png";
        char* to = "tux_ivt.png";

        in.LoadFromFile(from)
        out = CByteImage(in);
        ImageProcessor::GaussianSmooth(&in, &out, 1.0f, 3);
        out.SaveToFile(to)
}
```

Figure 7: Using IVT to blur an image

2.3 Research into Existing Work

2.3.1 Research into 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 brushstrokes 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 brushstrokes 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 [**?**]. 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 brushstrokes. An interesting part of their findings was the ability to date some of Van Gogh's paintings to a known period in his career [4].

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 brushstroke 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 brushstroke.

For each of these labelled candidates, a heuristic function is used to threshold any brushstrokes 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 brushstroke 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 brushstrokes 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.

2.4 Progress in Development and Experimentation

2.4.1 Design

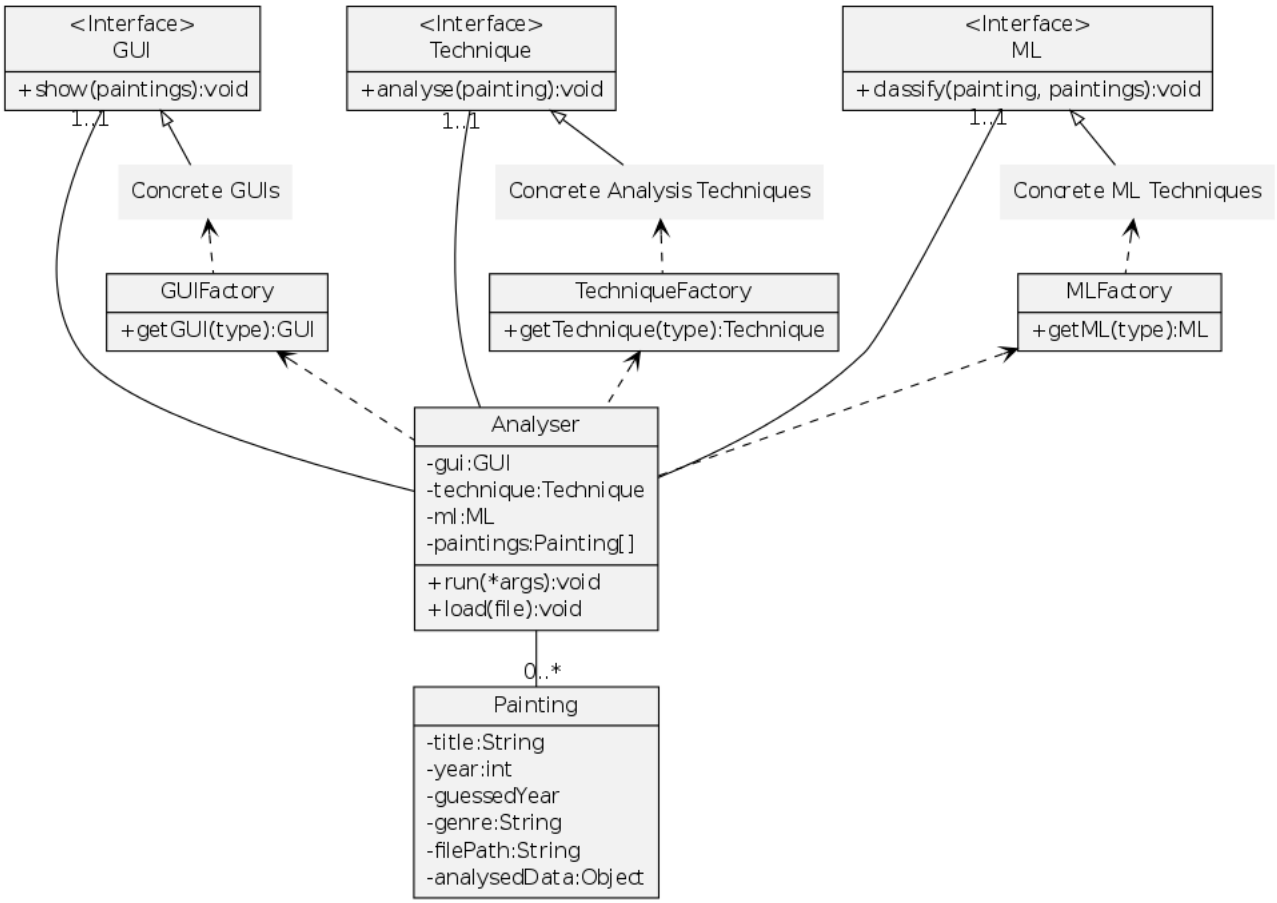


Figure 8: Initial Design

An initial UML class diagram is depicted by figure 8. This design will allow additional techniques, both analysis and machine learning based, to be added easily. Command line arguments, or later a GUI front-end, will be used to specify which techniques should be used.

The GUI elements depicted are used for visualising the analysed data or results of the machine learning algorithms. An example of this is a graphical representation of the colour space averages (see figure 10).

2.4.2 Colour Space Statistical Analysis

With the design solidified, I began to work on the statistical analysis of digital images. The easiest of these techniques is to read the image pixel by pixel, taking the various intensities at that pixel, then averaging them out over the whole image.

The most common colour space used is RGB (Red, Green, Blue), where each pixel holds three different intensities, one for each colour with a value of 0 to 255.

Using this analysis I plotted a graph of all paintings with a known year against the different intensities (see figure 9). Whilst the RGB values do all slowly increase over time and particularly the Red and Green values, the change may not be enough to classify new examples due to the fluctuations in the data.

Another popular colour space used in image processing is HSV (Hue, Saturation, Value) as it shows variations in colour, saturation and brightness separately (unlike RGB where all three

values are affected by changes in brightness). It was a simple matter of changing the existing code for RGB so that it would analyse HSV values instead.

Figure 10 shows a similar graph to the one in figure 9 only with HSV colour space instead of RGB. With this the Hue stays roughly the same throughout, whilst the saturation slowly declines and the value increases; this would make the colours brighter but with less colour to them. Whilst the reduction of colour over time is fairly obvious looking at Kyffin Williams' work, the increase in brightness was not something I would have necessarily have picked up on.

It's also quite interesting that both of these graphs has a dip during the Patagonia period, though both do have quite a high variance, meaning classification will tend to be unreliable. HSV does seem slightly less so than RGB.

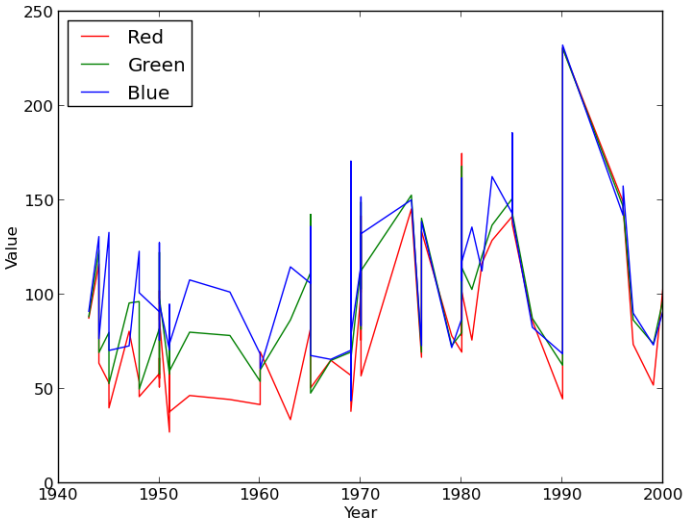


Figure 9: Mean RGB intensities by year.

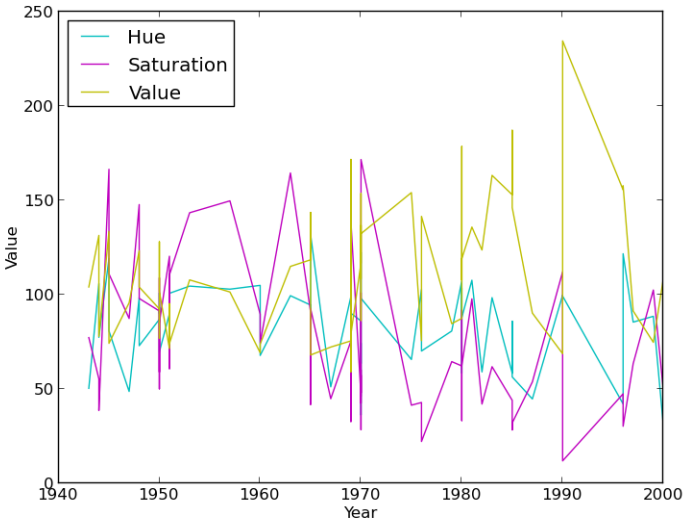


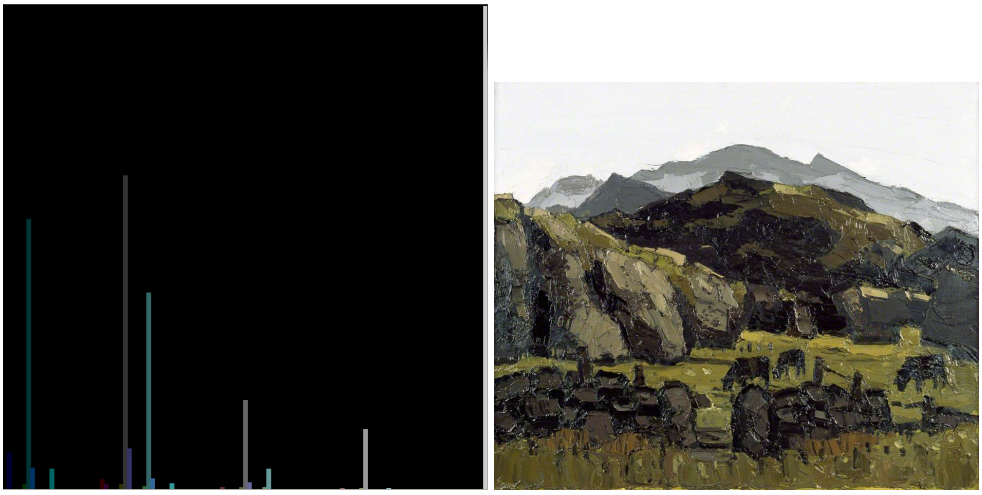
Figure 10: Mean HSV intensities by year. Note: value increases as time progresses showing Kyffin Williams' paintings became brighter as time went on.

2.4.3 Histogram Analysis

With colour-space analysis complete, the next sensible step was to start generating colour histograms from Kyffin Williams paintings.

A histogram is a nice way of representing the distribution of colour in a given image and is therefore a richer representation of a painting than colour-space averages.

As with before, histograms can be taken from different colour spaces, currently only the program can only generate RGB histograms (shown in figure 11), but it is trivial to include HSV histograms too.



(a) Histogram (b) Painting

Figure 11: Example of an RGB Histogram of Welsh Blacks Grazing on Snowdonia (1980).

From looking at the RGB histograms it becomes very apparent that Red is a very rare colour in isolation, except during the Patagonia period. Similarly green in isolation appears infrequently, where it is visible it's usually more of a teal colour. Very bright colours are also rare except in paintings with a lot of (Welsh) sky.

2.4.4 Nearest Neighbour Classification

Having looked at different representations of colour in paintings, the next part was to create a simple classification algorithm. The simplest of which is 1-Nearest Neighbour, that is; take the "Nearest" training example to the example to be classified and classify the year of this example with the year of the nearest example (see figure 12).

This simple algorithm can be expanded to be a K-Nearest Neighbour algorithm without too much effort (see figure 13). Instead of storing a single nearest neighbour you can simply store a list of them (making 1-Nearest Neighbour still work without issue) and then have some small amount of processing at the end to get a hold of the best neighbour. This processing will likely take the form of a simple statistic method such as the mean or the modal year.

$bestDistance \leftarrow \infty$
for $x = 1 \rightarrow X$ **do**
  if $distance(a, x) < bestDistance$ **then**
    $bestDistance \leftarrow distance(a, x)$
  end if
end for

Where:
a is the example to classify.
X is the list of all training examples.

Figure 12: 1-Nearest Neighbour

$kBest \leftarrow []$
for $x = 1 \rightarrow X$ **do**
  $cur \leftarrow distance(a, x)$
  for $k = 1 \rightarrow K$ **do**
    if $cur < kBest[k]$ **then**
      $temp \leftarrow kBest[k]$
      $kBest[k] \leftarrow cur$
      $cur \leftarrow temp$
    end if
  end for
end for

Where:
a is the example to classify.
X is the list of all training examples. K is the number of nearest neighbours to check.

Figure 13: K-Nearest Neighbour

2.4.5 Distance Measures

Working out the "Nearest" example to a given painting requires some way of measuring distance between the two.

One very simple way of doing this is Manhattan distance (equation 1). This distance measure can be improved further to Euclidean distance (equation 2) without too much effort. There are numerous other methods of measuring the distance of points in space, including those which focus on the distance between colour histograms, however Euclidean and Manhattan distance are both good starting points.

Manhattan Distance

$$d = \sum_{x=0}^{X} |a_x - b_x| \tag{1}$$

Where a is the first painting, b is the second, and X is all analysed values in both a and b

Euclidean Distance

$$d = \sqrt{\sum_{x=0}^{X} (a_x - b_x)^2} \tag{2}$$

Where a is the first painting, b is the second, and X is all analysed values in both a and b

2.4.6 Leave-one-out Cross Validation

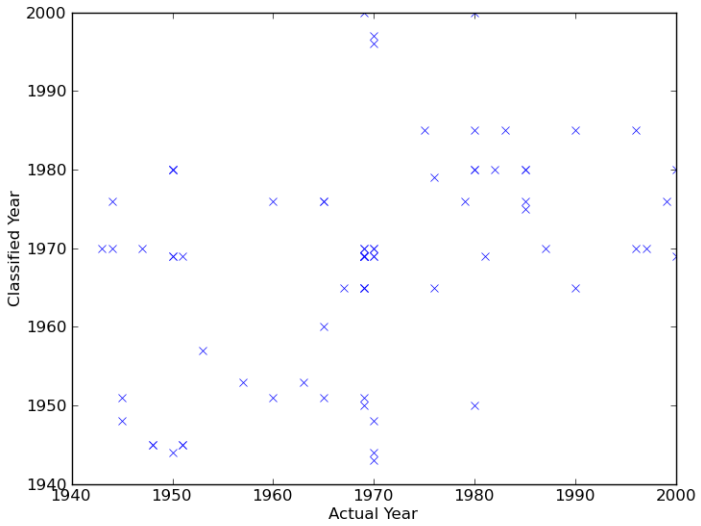


Figure 14: Leave-one-out cross validation of actual year against classified year using RGB statistical analysis and 1-nearest neighbour classification.

The idea of leave-one-out cross validation is to take the entire training set, remove a single example and then to classify this example against the rest of the training set. You can then do this for every example in the training set and be able to plot a graph of the actual year against the classified year, aiming for the line $x = y$, an example of this is shown in figure 14 and 15. With this information we can also work out the correlation between the two and have an indication of how each analysis technique performs, but as yet I have not had time to complete this step.

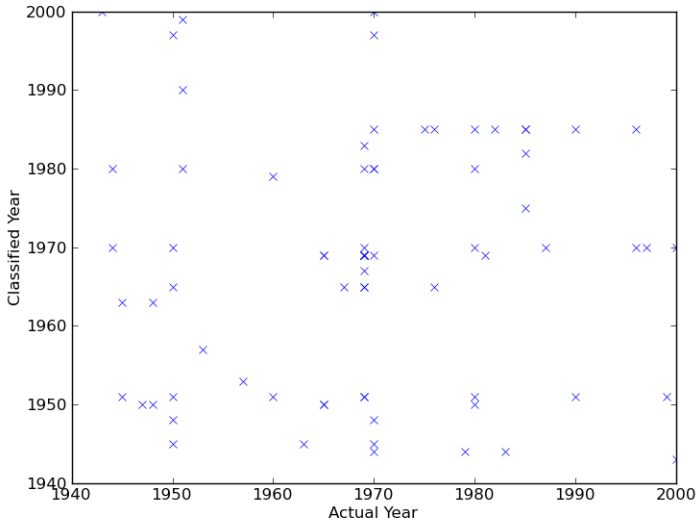


Figure 15: Leave-one-out cross validation of actual year against classified year using HSV statistical analysis and 1-nearest neighbour classification.

3 Planning

3.1 Methodology

The methodology I am taking as part of this project is a mix of iterative development with rapid prototyping. I think this approach will work well for this project as it is built up of a lot of small, isolated modules which perform discrete tasks.

Other models of development would likely slow the development of each of these modules down and are therefore less appropriate for this project. Some agile methodologies may also be a good approach, but as most agile methodologies are designed for teams rather than individual projects they are inappropriate.

Figure 16 depicts the Gantt chart I am going to try to follow for the course of this project. In this chart I have depicted some of the potential risks, such as other assignments and examinations. Unlike a lot of other people I don't have so much of a risk of having to travel to potential job interviews as I still have another year left in my degree.

3.2 Gantt Chart

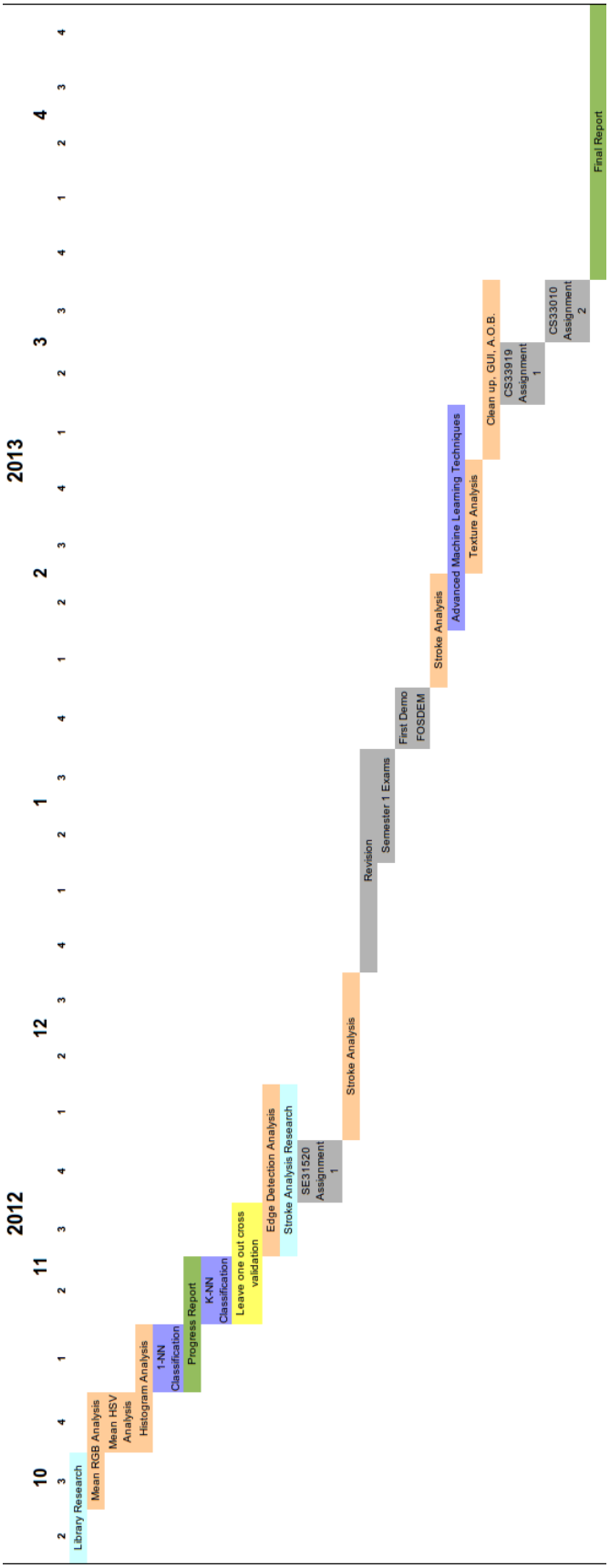


Figure 16: Gantt Chart for the Kyffin Williams project

3.3 Plans

I plan to have meetings with Gareth Lloyd Roderick around ever month or two to keep him informed of the progress of the project, to get his input into Kyffin's work and how a human might analyse paintings so that they might be converted into computerised ideas. I would also like to visit the National Library of Wales to view the large collection of Kyffin Williams' work in the stores there.

3.4 Mid-term Demonstration

For the mid-term demonstration I aim to be able to demonstrate a working, end-to-end, prototype which will generate correlations for each analysis technique and be able to rank them in some viewable way. Finally I want to be able to pass this technique an unseen example for which the date can be guessed relatively easy and see how if it can classify it as well as I can.

For this I would like to have the at least following techniques complete:

- RGB Statistical Analysis
- HSV Statistical Analysis
- RGB Histogram Analysis
- HSV Histogram Analysis
- Edge Orientation Analysis
- Simple Stroke Analysis

3.4.1 Final Demonstration

For the final demonstration I aim to have built on top of the prototype from the mid-term demonstration and to have added at least a complex stroke analysis technique as well as some other classification techniques other than K-Nearest Neighbour.

For the stroke analysis I would like to be able to visualise some of the process it takes to help explain how the algorithm works and the steps it takes to produce the final result.

I want to collect some paintings by a different artist and be able to compare it to Kyffin Williams' work, as well as seeing if the same algorithms can be applied to other artists.

Annotated Bibliography

[1] P. Azad, "The integrating vision toolkit (IVT)," Website, 2011. [Online]. Available: http://ivt.sourceforge.net/

Used documentation (http://ivt.sourceforge.net/doxygen/) and examples (http://ivt.sourceforge.net/examples.html) to create a simple image blurring application to test the capabilities and ease of use of the library. Used these resources from the 16 October 2012 to the 24 October 2012.

[2] 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.

[3] 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.

[4] 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.

[5] S. J. D. Prince, *Computer vision : models, learning, and inference*. Cambridge University Press, 2012. [Online]. Available: http://www.computervisionmodels.com/ 9781107011793.

Learning reference for Computer Vision and Machine Learning.

[6] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona, "Fiji: an open-source platform for biological-image analysis," *Nature Methods*, vol. 9, no. 7, pp. 676–682, June 2012. [Online]. Available: http://dx.doi.org/10.1038/nmeth.2019

Used as part of referencing FIJI documentation to test out of capabilities and ease of use of the library.

[7] SciJava, "Aggregator project for the fiji plugins 2.0.0-SNAPSHOT API." [Online]. Available: http://fiji.sc/javadoc/

Used as reference documentation to create a simple application to blur an image. Used from 11 October 2012 to 24 October 2012

[8] D. Stork and M. Duarte, "Computer vision, image analysis, and master art: Part 3," *IEEE Multimedia*, vol. 14, no. 1, pp. 14–18, 2007. [Online]. Available: http://dx.doi.org/10.1109/MMUL.2007.6

Defines a potential method of analysing regions in a painting, this could be interesting to apply to Kyffin Williams' work as the regions in his earlier work are a lot less well defined as those in his later work.

[9] D. G. Stork, "Computer vision and computer graphics analysis of paintings and drawings: An introduction to the literature computer analysis of images and patterns," ser. Lecture Notes in Computer Science, X. Jiang and N. Petkov, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2009, vol. 5702, ch. 2, pp. 9–24. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03767-2_2

Notes a lot of useful literature to look at and some useful terminology and ideas for analysis techniques too.