2014

Feature Extraction and Texture Analysis in the Classification of Paintings

Jacob Fauber

Feature Extraction and Texture Analysis in the Classification of Paintings

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Jacob Fauber

Annandale-on-Hudson, New York
December, 2014

Abstract

Painting classification is a novel subfield of research in computer vision and image processing that focuses on empirically defining a painting's authorship, date of creation, or genre. The applications of the subfield have increased in response to changing art markets and expanding and emerging research. Basic features like color and line in and of themselves prove inadequate in this study. Texture analysis provides one of the promising additional features. In this paper, I specifically investigate two different measures of texture analysis, Gabor filters and Haralick features, in addition to basic features, and their applications in a specific painting classification problem. I also consider different methods of collecting this data, and the impact of each on classification.

Acknowledgments

My deepest thanks and gratitude must be paid to the following people:

- My senior project advisor, Keith O'Hara, for his continual academic support, openness, and helpfulness in bringing my rather silly idea to fruition.
- The other members of the Bard Computer Science department, including Rebecca Thomas, Sven Anderson, and Bob McGrail, for the pivotal role each played in my education over these four years.
- My family for their unending emotional (and financial) support
- All my extraordinarily understanding friends. You know who you are.
- Any of these artists whose albums are among my senior project playlist, which has been on constant rotation these past few months:

  Beth Orton's Central Reservation
  The Go-Between's Oceans Apart
  The Weepies' Say I Am You
  Antonio Carlos Jobim's Stone Flower
  Sam Phillips' Don't Do Anything
  Mountains' Choral
  Mouse On Mars' Niun Niggung
  Horace Andy's Skylarking
  Nick Lowe's The Convincer

  Might they serve you as well as they have me.
- And of course, you. Anyone who takes time from their day to read an undergraduate thesis really deserves an award of some kind, so I guess a dedication will have to do.

Contents

List of Figures

1 Introduction

1.1 Motivation

The art market is one of the largest unregulated markets in the world. As the economic and academic uses of art continue to expand, computer analysis of these images can become very useful. Increasingly accurate categorization may eventually lead to forgery detection and completed catalogue raisonnés for artists with dubious history or provenance. As the art market moves online, automated categorization and analysis can help market paintings to interested collectors and expand the market.

Though research into general image analysis is plentiful, little work looks specifically at paintings, and little is understood on how to best categorize them. A suggested but little studied part of the painting categorization literature is texture analysis, which is used to capture brushstroke [13]. A more sophisticated knowledge of effectiveness of texture analysis is useful in these efforts.

1.2 The Image: Foundations

It is an unfortunate but utterly necessary element of image representation on computers that the image be digital. Most images used in classification are representations of a non-digital object, and their representation is digitized. This is an inherently flawed process (cf. 1.3.3), but serves well enough in representing most images that image processing may successfully take place.

The following explains what can be expected of a digitized image, and the notation this paper uses to represent them.

1.2.1 Representation

What does it mean, fundamentally, for an image to be digital? It means the image is discrete. As such, the image can be broken down into fundamental component parts, in this case a *pixel*, and a collection of these represents the image.

There are many ways of representing a pixel. If the image is grayscale, a pixel is almost universally represented by a byte, an integer value between 0 and 255, which are absolute black and white, respectively.

To represent color, pixels are generally structured as a tuple, containing various values which, taken collectively, represent a color value. The significance of these values changes with respect to the *color space*. The precise method of pixel representation used for painting classification varies, and more information can be found in section 3.1.1. There are various ways of representing a pixel in an image. However, every pixel must have the same method of representation.

Pixel location will here be represented with simple Cartesian coordinates.

$$\mathbf{p} = (x, y) \in \mathcal{R}^2 \tag{1.2.1}$$

In regards to location, if $\mathbf{p}$ = (0,0), that pixel will be located at the top left corner of the image.

Taken together, at their proper locations with their proper locations, pixels generate the illusion of a continuous image. The computer, of course, has no concept of continuity; it understands the image only with respect to contiguousness.

1.2.2 Convolution

After obtaining a digitized image, certain information can be gathered by iterating over the image matrix and performing calculations on the data. Information like color palette, mean color value, and color variance can all be understood by performing calculations and summations over the image proper.

Rather than examining the image itself, a filtering process known as *convolution* can occur instead. After this process, there is a new but still relevant image, from which new information can be learned about the original, such as edge counts, corner locations, and information density. The act of convolution, in this context, is also known as *neighborhood filtering*, and means to generate a new pixel value with respect to that pixel and its neighbors. The neighboring pixels used depend on the size of the smaller convolution matrix, called a *kernel.*

The act of convolution itself will be represented by $*$. The following is the convolution formula for a 2D signal.

$$g(x, y) = F * I(x, y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i, j)I(x - i, y - j) \tag{1.2.2}$$

The following example shows the appearance of the matrices during convolution of a single bold data entry $\mathbf{p}$ = (1, 1) in the image matrix F.

$$F(1,1) * I = g(1,1)$$

$$\begin{bmatrix} \mathbf{45} & \mathbf{60} & \mathbf{98} & 127 & 132 \\ \mathbf{46} & \mathbf{65} & \mathbf{98} & 123 & 126 \\ \mathbf{47} & \mathbf{65} & \mathbf{96} & 255 & 119 \\ 47 & 255 & 91 & 107 & 113 \\ 50 & 59 & 80 & 97 & 110 \end{bmatrix} * \begin{bmatrix} 0.087 & 0.121 & 0.087 \\ 0.121 & 0.168 & 0.121 \\ 0.087 & 0.121 & 0.087 \end{bmatrix} = \mathbf{68}$$

Figure 1.2.1: Convolution about one pixel in image F

The appearance of the convolved image matrix for all points (x, y):

$$\begin{array}{ccccc} F & * & I & = & g \\ \begin{bmatrix} 45 & 60 & 98 & 127 & 132 \\ 46 & 65 & 98 & 123 & 126 \\ 47 & 65 & 96 & 255 & 119 \\ 47 & 255 & 91 & 107 & 113 \\ 50 & 59 & 80 & 97 & 110 \end{bmatrix} & * & \begin{bmatrix} 0.087 & 0.121 & 0.087 \\ 0.121 & 0.168 & 0.121 \\ 0.087 & 0.121 & 0.087 \end{bmatrix} & = & \begin{bmatrix} 50 & 67 & 95 & 118 & 128 \\ 51 & 68 & 106 & 132 & 136 \\ 68 & 91 & 125 & 133 & 134 \\ 75 & 97 & 121 & 120 & 123 \\ 69 & 86 & 98 & 98 & 107 \end{bmatrix} \end{array}$$

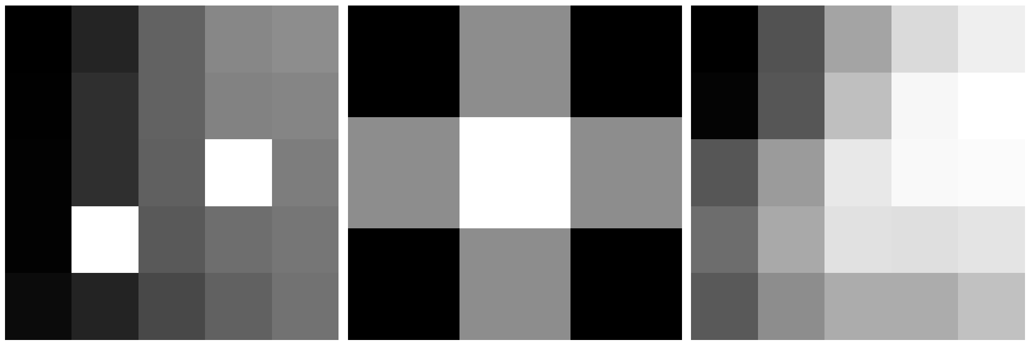


Figure 1.2.2: Representations of F, g, and convolution kernel I

1.2.2.1 The Border Case

The one small caveat to convolution is that when faced with the border, where there are not enough pixel neighbors to fit the standard $I(x, y)$ kernel, a different case must be met. The standard way of dealing with convolution of borders in this project has been the *reflect* case, where outer values are simply 'reflected' across the image matrix border to fill the blank spots in the kernel window. All border handling can become problematic, but for large enough images the borders become increasingly negligible, as long as their case is reasonably handled.

1.3 The Data Set

A significant element of the project of painting categorization is simply selecting which artists and paintings to categorize, and which digitizations of these paintings to consider. While it is a theoretical nicety that categorization might be generalizable to all artists, it is undeniable that certain artists are more or less correlated to certain patterns than other artists, and different artists may tell of their authorship in radically different ways. For this reason, certain painter pairings have shown themselves to be easier to distinguish then others [2]. This is part of the significance of artist selection.

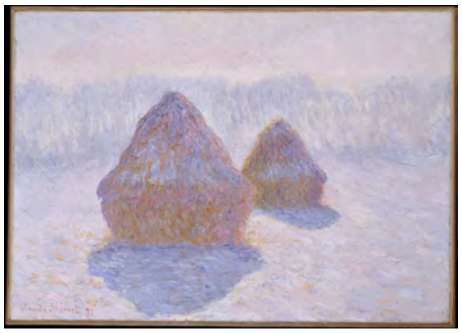
1.3.1 Artists

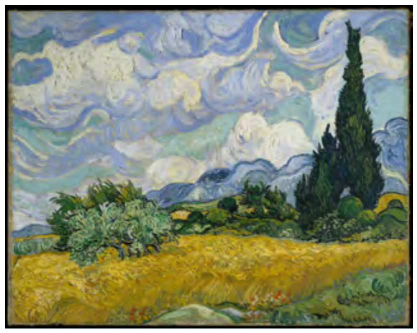
Four artists were used for the categorization project at hand: Cezanne, Monet, Rembrandt, and Van Gogh. Artists were chosen partially for their historical importance, but also bearing in mind variety. Cezanne and Monet were picked, for example, for their purported similarity in palette and mark, while Rembrandt and Van Gogh were chosen for their clear distinction in markmaking and palette. It is expected that the classification will react differently to the comparison of any two of these four artists. An essential aspect of selection is also the size of each artist's available catalogue. In this case, between 15 and 20 paintings were used for each artist. 68 images were used in total.

1.3.2 Sources

It is difficult to gather freely available digitized paintings from reputable sources. Unreputable sources pose a threat to image stability, with the possibility of, among others, cropped images, watermarks, and mislabelled information. All images used were there-

(a) Cezanne


(b) Monet


(c) Van Gogh

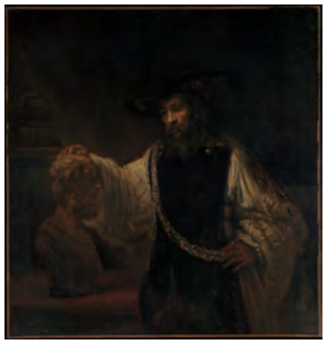

(d) Rembrandt

Figure 1.3.1: A sample work from each artist

fore culled from the Metropolitan Museum of Art's Open Access to Scholarly Content program [1].

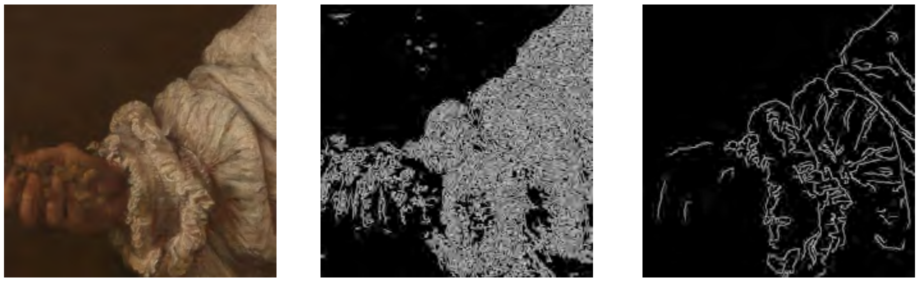
1.3.3 Potential Problems

Drawing all images from the same source should control for any problems in image digitization, as one would assume all images from the same reputable source have the same scanning perimeters. All images should contain the same faults, if there are faults. Nonetheless, various conundrums regarding image digitization still occur.

At high resolutions, even dust or particulate matter can affect a scan, and a certain amount of dithering or, alternately, color banding may cause inaccuracy in the image representation. Though this would not impact and would actually likely aid image categorization, it would make the categorization inauthentic, relying on lossy compression or failed scanning methods rather than artistic style.

1.3.3.1 Scanning Artifacts

Small aspects of a scan or picture can impact painting analysis, especially at high resolutions. This is especially prominent during edge detection.



(a) Hand from Rembrandt's *Flora* (b) Canny Edge Detector without blur (c) Canny Edge Detector with Guassian blur

Figure 1.3.2: Inaccurate edges are detected without blur. See lines in upper left of subfigure (b).

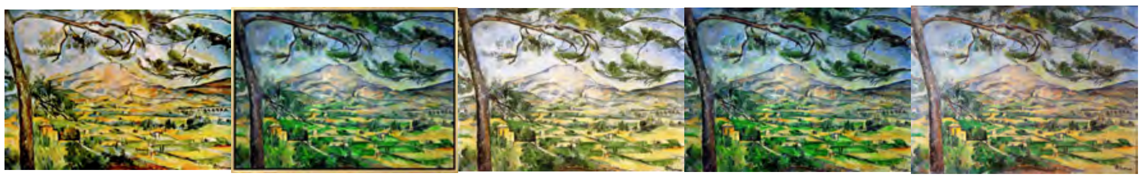
To remove this effect, a strong Gaussian blur (standard deviation of 2) is applied to the image before edge information is gathered. This removes artifacts, but may destroy some less essential line information. As an added benefit, this removes issues with craquelure, or naturally occuring cracks that appear in oil paint over time.

More information on the Canny edge detector can be found in Section 3.2.1.

1.3.3.2 Scanning Variation

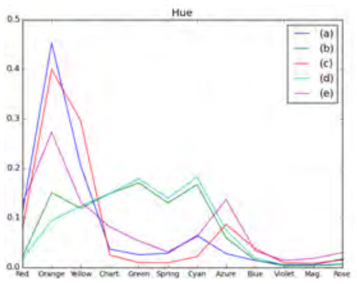
A significant problem with digitized paintings are their variability. Changes in light, time of day, and scanner or camera type all influence the appearance of the image.

It is difficult to account for this problem. In this case, it is assumed that pictures taken from the same sources will have a limited amount of signal interference, and that this interference will be similar across all images.

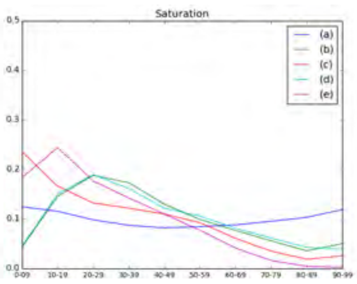


(a) (b) (c) (d) (e)

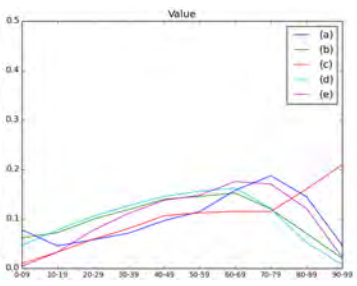
Figure 1.3.3: Five distinct samples of Cezanne's *Mont Sainte-Victoire with Large Pine*, taken from Google Images.



(a) Hue Histogram



(b) Saturation Histogram



(c) Value Histogram

Figure 1.3.4: The variety between all five images is easily visible in the color histograms.

1.4 A Brief Overview

The target goal of this project is to successfully classify paintings by artist. There are two necessary components of this task that appear in virtually all research of it: feature collection and classification [2] [8] [9] [12].

1.4.1 Feature Sets

Other approaches into painting classification will consider different information about each painting. Since the premise of this paper is to compare different types of texture analysis (as a surrogate for brushstroke detection), the other features collected were nominal, but still enough for substantially accurate classification to take place. Color information, edge information, and two distinct forms of texture information were gathered. The resulting information was condensed into a 128-feature vector which was then used for classification. A sample data entry can be found in the appendix.

1.4.2 Classification

The problem of classification means assigning data the correct one of a set of category labels, utilizing a set of training examples that are already correctly assigned a label.

Various types of classification algorithms and data structures have been proposed. Some research papers in image analysis make it their goal to discover which classification algorithm succeeds best for the given data [2].

Since the aim of the paper is to compare methods of texture analysis, only one classification algorithm need be used. Random forests were used, for reasons discussed in section 4.1.

2
Previous Work

Painting analysis is a relative novel field, even if it built off of the much older and theoretical fields of texture analysis, image analysis, and classification. The first literature, focusing mostly on digital restoration, began to appear sporadically in the late nineties [13]. Real attempts at accurate classification didn't see publication until the 2000s. This is certainly not because the technologyisn't available (although classification problems on such a large scale are still seeing significant refinement). To speculate on the recent trend, there was not yet significant economic pressure or means for easy implementation. The presence of computer science as a pursuit of humanities, too, was still lacking.

Reviewing the painting classification literature of the past 15 years gives insight into some of the primary categorization and feature extraction methods, as well as communicating what is lacking or still unconsidered in the field.

2.1 Features Used

Much of the starting literature focuses on achieving classification that is as accurate as possible. The theme generally seems to be to throw as many features as possible at a classifier, and hope it is resilient to any noise that may occur, or that the new information in the data set will at least be distinct enough to compensate for it.

Color features are virtually always considered, and are thought of as the 'basic' features of painting analysis [2].

When accuracy is the primary motivation of the paper, a set of features make a consistent reappearance, most of which do not relate intrinsically to the texture of the image. These are SIFT, Color Histograms, Histogram of Gradients (HoG), and Saliency Maps [12] [2]. There are other suggested features, including texture analysis methods. Gabor filters are occasionally used in a non-texture classification mechanism called 'spatial features', and Fourier analysis was also promoted as a 'good feature' [14].

In general, when desiring accuracy, more distinct features are better than fewer.

2.2 Categorization

In efforts to accurately categorize paintings, different categorization methods were used. Researchers used the Naive Bayes Classifier, the SVM classifier, and random forests, and routinely found the three perform increasingly better, respectively [2] [12]. The distinction in performance was as drastic as almost 10%.

The categorization problem can vary across research as well. More often then artist classification problems, researchers consider problems within a single artist's catalogue, or dating a wide variety of works across time. The work of Hedges created a 'clock' based on mark to date Renaissance copperplates, disregarding specific artist information in favor of roughly estimating dates of any given plate based on other samples [6] [13]. This works best given the assumption of stylistic similarity across artist. The research of Johnson and Jafarpour investigated paintings of Van Gogh, again classifying for dates of creation, but this time aiming for extreme accuracy, and assuming reasonable diversity within just one artist's catalogue [9] [8].

2.3 Painting Analysis With Gabor Filters

The work of Johnson et al. is probably the most investigative study of texture analysis in reference to paintings (or less formally, brushstroke analysis) that has been published thus far [9] [8]. In some ways, their mission is outside the scope of this paper, as they examine the artist Van Gogh specifically, and attempt to date paintings and identify forgeries. But it is this tight focus that inspires the depth of their work into texture recognition, using the fundamental thesis that the stroke of each artist is the most telling attribute of his work. Most significantly, the "Maastricht" group's method of investigating brushstroke focused on Gabor wavelets. Information on wavelets can be found in section 3.3.3.

The work of the Maastricht group establishes the premise of scale and direction importance. Notably, Johnson references the kernel bank as "multi-scale oriented Gabor wavelet filters." The Maastricht group used six orientations at four scales.

The Maastricht group created a summary statistic referenced as 'energy,' which exists for each pixel. The energy value is quite simple the sum of squared value obtained after convolving the real and imaginary part of the Gabor wavelet, collectively known as a Gabor filter.

Energy proves to be a uniquely idiosyncratic classifier in spite of its simplicity, varying highly from painting to painting, and unwaveringly high when given a forgery. The value of the system is likely due to the large kernel bank they used.

The Maastricht group here proposes more information can be constructed from this simple summary statistic. Formerly, energy was averaged across all partitions of a painting, but if one constructs a multidimensional histogram of the energy for each partition, greater accuracy is possible. Though promising, it is not a near perfect system; Johnson notes the research both failed to identify two of six Van Gogh forgeries, and incorrectly identified two real Van Goghs as forgeries.

3

Feature Extraction

A variety of features were collected to be used in classification. The nature and type of each feature collection method is described here.

3.1 Color Spaces

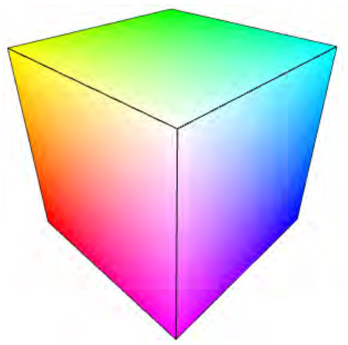
As discussed in section 1.2.1, pixels have specific method of representation. Traditionally, if an image is grayscale, each pixel is represented by a byte of value 0-255. In most paintings, however, color is of essential interest, and images will not be represented as such unless specifically converted.

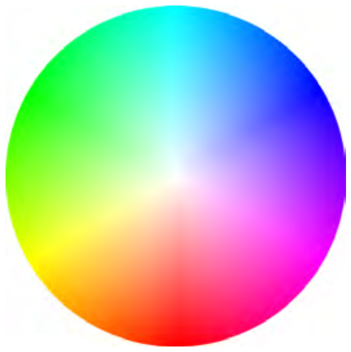
When collecting color data, it is essential to consider color space and collection method.

3.1.1 RGB & HSV

There is a large variety of color spaces for an image to be represented in, each of which influences methods of representation and color conversion. The most traditional representation of color space is the RGB (red-green-blue) color space.

Every color in RGB-space is represented by 3-tuple (for red, green, and blue, respectively), with each integer value representing a byte (an integer between 0 and 255), as in grayscale representation, but instead representing the degree of their respective color in that image. As an example, a point in RGB-space at (x,y) valued at (0, 0, 0) would be absolute black. A point valued at (255, 0, 0) would be absolute red. A point valued at (255, 255, 255) would be absolute white. Since each value in the tuple is of the same value, all colors available can be accurately contained within a cube of each side of length 255.


(a) RGB Cube exterior


(b) 2D HSV wheel for hue and saturation

A less common color space is HSV (hue-saturation-value). As in RGB-space, a point in this color space is represented by a 3-tuple, and both color spaces can represent virtually all the same colors. HSV-space is bounded by a cylinder, rather than a cube. The different method of representation allows different types of image processing and different information about the image data to be presented.

Hue is defined in polar coordinates, and is therefore presented as an integer between 0 and 360. Hue represents the *pure color type* of a pixel. A dark or light red pixel has an equivalent hue to a vibrantly red pixel. This is particularly useful when considering images affected by shadow or lighting.

Saturation is represented by an integer value from 1 to 100, and is associated with the *purity*, or *whiteness*, of the color. Saturation is defined as the distance away from the center, or the length of the vector whose angle is determined by the hue. At the furthest distance is the 'absolute' color, a 'pure' red, yellow, etc. At the center (no distance) is pure white. This can be effectively seen in Figure 3.1.1b

Value is defined along the z-axis of the space. At the bottom is black, and at the top is, once again, the 'pure' color value. An HSV-space pixel with a saturation and value of 100 will be an absolute color. If the saturation is valued at 0 and the value remains 100, the color will be absolute white. If value is 0, the color will be absolute black, regardless of the saturation or hue.

This means that a proposed pixel value of, say, (0,0,0) and (180, 99, 0) are equivalently black. To compensate for the fact that multiple inputs can map to the same color, a simple change is made to the conversion process to convert the proposed saturation value to *chroma*. Chroma is value-dependant, and its range of possible values decreases relative to the value. The color space is now represented in a cone, which can fit inside of the cylinder representing HSV-space. This method of representation was used in the wavelet based research of Jafarpour et al [9].

While notable, chroma is not being used in this instance. HSV is sufficient, as it still represents pixel values in Cartesian space, which is necessary for edge detection and image processing, and provides a more intuitive and perceptual representation of color useful for classification and understanding.

3.1.2 Conversion

The conversion from RGB-space to HSV-space is simple. Below is pseudocode that represents the method of transition from RGB-space to HSV-space.

```
method rgb2hsv(r, g, b)

        //first calculations
        ma = max(r, g, b)
        mi = min(r, g, b)
        c = ma - mi //c here can represent chroma
```

```
//hue
if c == 0
         h = undefined
else if ma == r
         h = ( (g - b) / c ) % 6
else if ma == g
         h = ( (b - r) / c) + 2
else if ma == b
         h = ( (r - g) / c) + 4
hue = h * 60

//saturation
if ma == 0
        sat = 0
else sat = c / ma

//value
val = ma

return (hue, sat, val)
```

3.1.3 Binning

Color, when available, usually serves an important role in image classification, and the instance of painting classification is no different. Gathering color information can be a complex task. The wide variety of color spaces is particularly enough to complicate the issue. There is a wide variety of information traditionally used in the task, including the 'percentage' of dark colors (here represented in 'value'), the range of color 'peaks', and the 'skew' of color, or deviation of the grey levels away from a standard Gaussian distribution [7].

HSV-space is often the most effective color space for categorization, and with the image having been converted, some sort of information about the image must now be gathered. The most common means of doing so is called *binning*.

Binning is, in this context, the means of grouping data together into meaningful pre-defined sets. It is a form of quantization. Binning is a common technique with myriad applications. They are used for hash tables, image resizing, and most research with large data sets.

The types of bins in the case of HSV-space painting images is two-fold. For hue, values are grouped in increments of 30, while saturation and value are grouped by increments of 10. This yields 32 bins overall. The quantity of pixels belonging to each bin is then normalized to one, giving the percentage likelihood of any one pixel belonging to that bin.

It is most like human-designated classification of color to group hue in some number of groups $n = 3 * 2^x$ where the size of each bin is $360/n$. This is because of the natural definition of hue. The entire color wheel, seen in Figure 3.1.1b, can be divided first into 3 bins, yielding the primary colors, and then 3 more, yielding secondary colors. 6 more gives tertiary colors, 12 the quandary, and the process can continue until there is more than one bin for each possible hue (when $x = 7$), after which continued binning would be functionally useless.

Dividing into over 360 bins risks capturing for lighting variation and scanning artifacts once more. The high dimensionality of the data would also risk confusing a classifier. 12 bins is picked as a happy medium, capturing color information while not including too much information.

For saturation and value, 10 bins are chosen as an accompaniment to hue, being the nearest whole to the number of hue bins that divides evenly into 100, the range of both types of pixel color information. Were hue bin values to scale up or down, value and saturation should accordingly.

Color can act as a strong classifier. Though most of the artists seem to follow the same rough trend (which is intriguing in and of itself), significant enough distinctions between artists become clear when their color choices are quantified in bins.

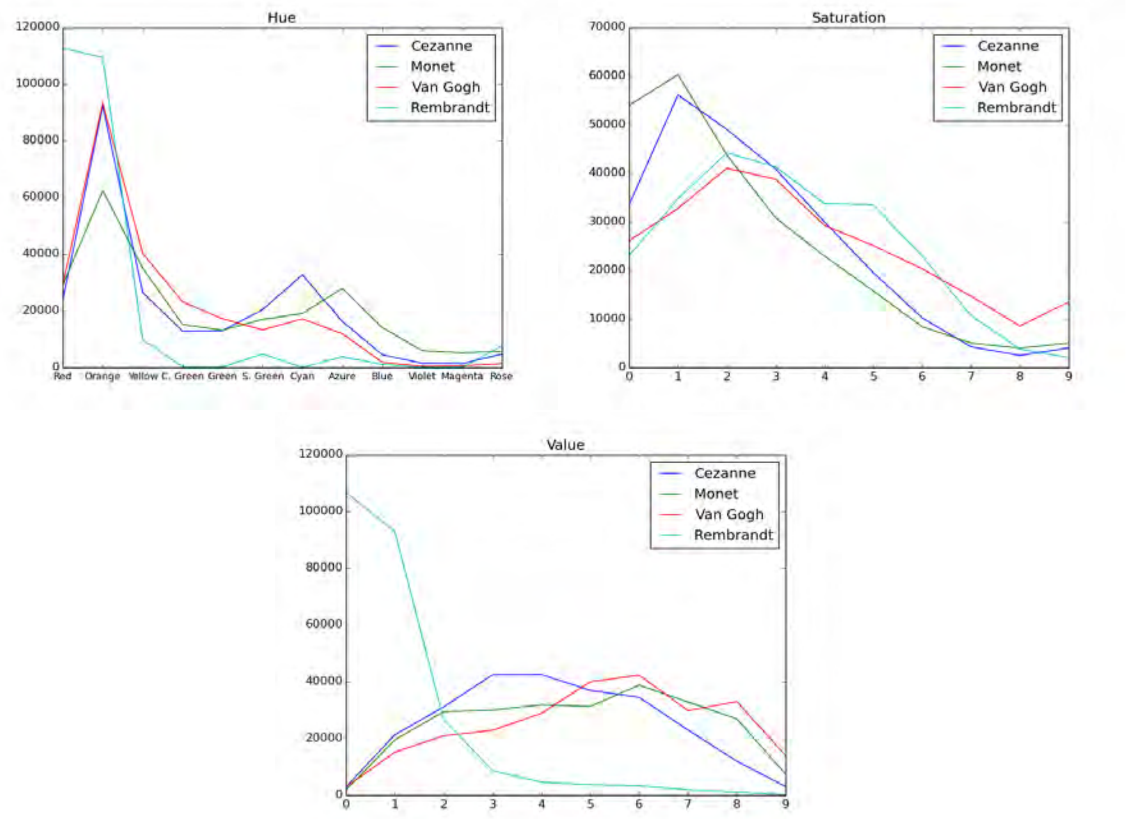


Figure 3.1.2: Average color graphs for each artist normalized to 512x512 patch

3.2 Edge Detection

In addition to color, edges often prove to be a very strong basic indicator of the artist's hand [12] [9]. Like color, there are both a variety of means for detection and a variety of types of information that can be garnered from edges. Edge detection, in this case, is not detection for stroke, but rather for figure and division of space.

There are a variety of well-established and relatively simple methods for edge detection, such as the Difference of Gaussians (DoG), the Laplacian of Gaussian (LoG), and the Sobel filter, each of which depend, at least in part, on the act of convolution.

Extensive research has been performed on the value and significance of edge detectors relative to one another [11] [10]. Though no choice could be reasonably deemed a failure, certain edge detectors are preferred. The edge detection algorithm used in this case is the Canny edge detector. It was chosen for several reasons. Firstly, it routinely outperforms its simpler and previous named counterparts, primarily DoG and LoG. Secondly, it provides edge parameters as a binary matrix, providing clarity and strict definition of the edge. Most importantly, its performance has a heavy degree of malleability. Various components, including a Gaussian blur and hysteresis thresholding, are changeable to given parameters, so experimentation for optimal performance is preferred. As seen in Figure 1.3.2, this adaptability can be extremely beneficial.

3.2.1 The Canny Edge Detector

Firstly, a Gaussian blur is usually convolved about the image to remove noise. In this case, the Gaussian blur was selected to be uncommonly strong, with a standard deviation of 2, for reasons already discussed.

Let g equal an approximation of this blur, as used by the Canny filter. Let f equal the a grayscale version of image in question.

$$g = \begin{bmatrix} .035 & .038 & .040 & .038 & .035 \\ .038 & .042 & .045 & .042 & .038 \\ .040 & .044 & .047 & .044 & .040 \\ .038 & .042 & .045 & .042 & .038 \\ .035 & .038 & .040 & .038 & .035 \end{bmatrix} \tag{3.2.1}$$

$$f * g = f' \tag{3.2.2}$$

After this Gaussian blur is convolved with the image, the edge detector needs to identify potential edge magnitudes and and directions. This is done with an *edge detection operator*, traditionally with two Sobel filters, one for the x and y directions.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 2 \end{bmatrix} * f \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * f \tag{3.2.3}$$

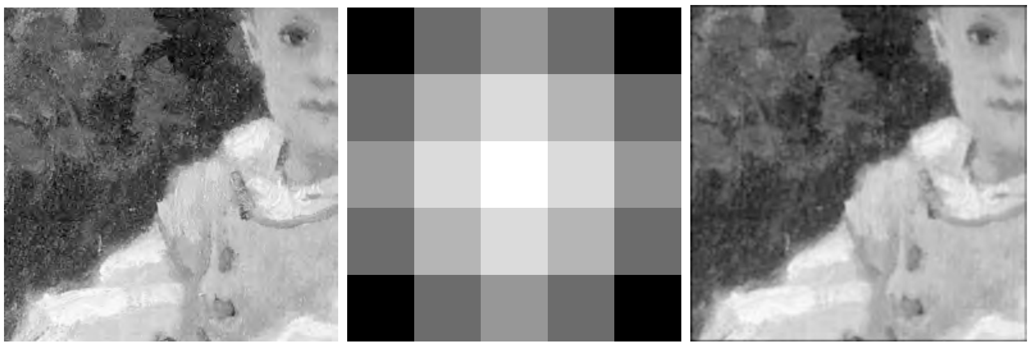


Figure 3.2.1: Visual representation of a Guassian blur

Where G is the gradient strength, and θ is the edge direction.

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{3.2.4}$$

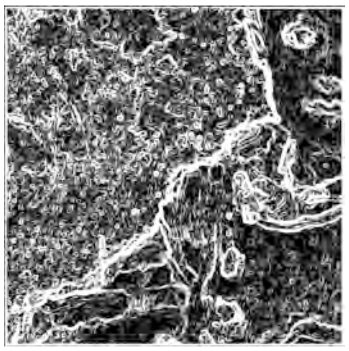
$$\theta = arctan2(G_y, G_x) \tag{3.2.5}$$

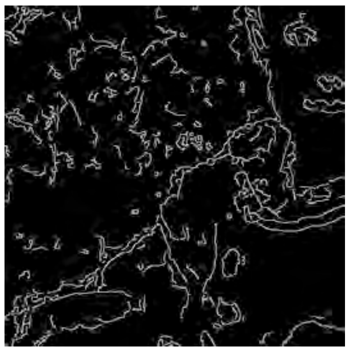
Theta is rounded off to the nearest multiple of 45, as edge direction can only be diagonal, horizontal, or vertical.

The Sobel filter now yields the appearance of edges, but they are weighted rather than binary. The definitive edge is not yet determined.

To accomplish this the process of edge thinning occurs. This uses theta to account for any potential edge pixel's two directional neighbours. If the point has a greater magnitude (G), than it is considered a potential edge.

Lastly, there is the process of hysteresis thresholding. Whether or not a line is part of an edge is in this case a neighbor-dependant problem. Simply put, a potential edge pixel is more likely to be an edge if it is near other accompanying potential edge pixels. The necessary magnitude of these potential edge neighbors is given by a previously determined set of thresholds, high and low. If the potential edge pixel is above the high threshold, it is definitively an edge. If in between these the high and low parameters, and next to an edge pixel, it is an edge. If below the low parameter, this pixel is not an edge.


(a) The Sobel filter


(b) The Canny edge detector

This is considerable in regards to painting, where edge detection can pick up changes in stroke, lighting, and color changes in equal measure. If the threshold is set too low, even with a strong Gaussian blur arbitrary edges will be picked up. If the distinction between high and low thresholds is too large, lines will split off, accounting for parts of brushstroke and accidental shifts in light. In this case, the low threshold is set to 15% of the maximum intensity in the image. The high threshold is set to 20%.

After hysteresis thresholding, the image is now a binary matrix, indicating the edges of the image relative to the blur and hysteresis threshold values given.

3.2.2 Line Features

After edge detection has been performed over the image, information on the edges still needs to be gathered. Two primary statistics were used: edge length and edge count.

Edges were found by iterating across the image. On finding an edge point, the algorithm would create an array representing the edge, add this pixel to the array, and perform this operation again on any surrounding edge points. This means edge length is defined not by the literal length of the edge, but rather the number of pixels that make up its construction. Edge count is the number of edges found in a single image.

The edge gathering mechanism is very similar to the problem of connected component labelling. The image can be interpreted as a graph, where each 'True' labeled pixel in the edge detected image represents a vertex, and an edge exists between any of the eight adjoining pixels. The labelling method used here is a simple single-pass algorithm.

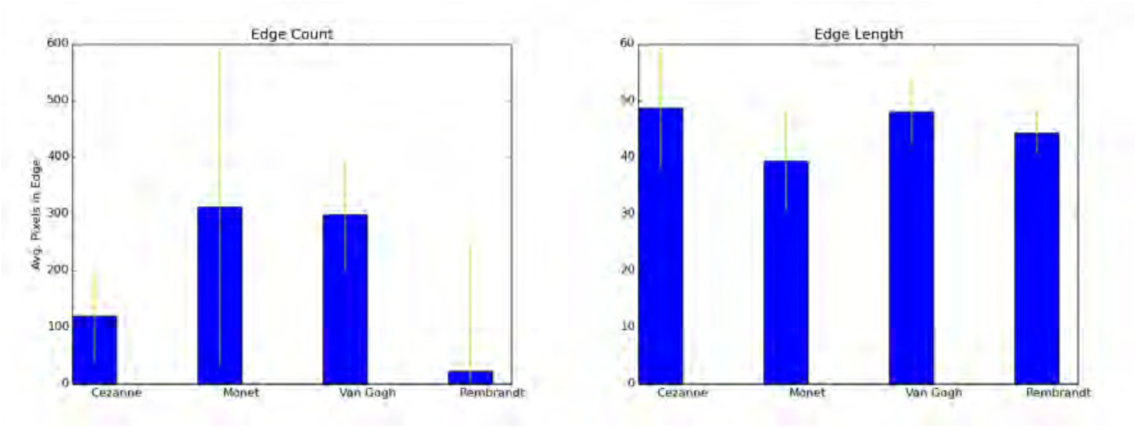


Figure 3.2.3: Edge count and edge length for each artist

The high degree of variance in certain instances is generally unexplained, beyond that of natural variation within an artist's catalogue. Outliers as large as edge lengths over 75 and edge counts over 700 appear for specific paintings.

While these outliers may prove troublesome for classification, they tell us something interesting about the image to which they belong, and the artist in general. A painting like *The Parc Monceau* by Monet has the most edges per patch of any painting of all four artists. For the extent to which edge count might stand in for complexity or information density, this statistic might suggest *The Parc Monceau* is the most complex of Monet's entire catalogue.

3.3 Texture

Edge information may feel influence from stroke and texture, but it does not account for either. For one, edge information does not account for direction. For another, edge

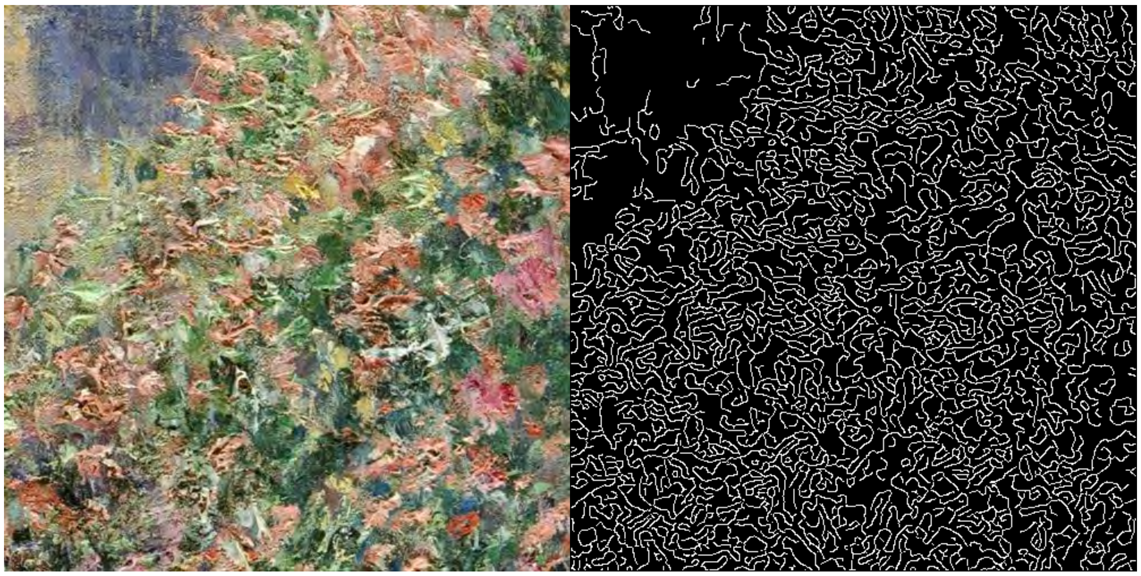


Figure 3.2.4: This patch from *The Parc Monceau* has over 1000 edges. It might be considered the most complex patch from Monet's work.

information does not account for types of transition, merely the fact that it occurs. For instance, a black and white image would read the same edge information even if color inverted, while the texture would be wildly different.

It is difficult to define texture, despite its ubiquity in visual experience. The concept of texture is that, even with a stochastic process applied to it (e.g. noise, variation, shading) there is an underlying deep structure and pattern to a material surface which aids in its recognition.

3.3.1 Texture and Brushstroke

It is a fundamental assumption of this project that artists have an idiosyncratic texture to their work. This texture is manifested not in the color, but in the *brushwork* of an artist. This proposition has been repeatedly confirmed, as attempts to classify artists by brushstroke have proven relatively successful, even allowing artworks within an individual artist's catalogue to be dated, and for potential forgery detection to take place [8].

However, brushstroke is not precisely texture in painting, but rather an essential aspect of it. To consider texture a more representative measure of brushstroke, factors like sizing and lighting must be accounted for.

Even texture itself can only be purported as measured by various techniques. In reality, the relationship any proposed computational measurement has to a more human abstraction is never quite set in stone. It is only through experimentation that one can really access the effectiveness of a texture measurement in actually representing texture. R.M. Haralick makes note of this in the last appendix of his well-known texture-analysis method, discussed below [5].

3.3.2 Haralick Features

Haralick features are a rather well known form of texture analysis, originally designed for distinguishing bird's-eye view images of crops and land. Their use in painting is fairly limited, likely because of the remoteness of their original application from painting, but not unheard of [12]. As a method of general texture analysis, they are often very successful, but their capacity for a task as specific as painting is not well-known.

3.3.2.1 The Co-Occurrence Matrix

Haralick features are just summary statistics for a basic underlying data structure: the co-occurrence matrix. It serves to measure the probability that any given pixel gray level will appear adjacent to any other pixel gray level. The co-occurrence matrix is a square matrix of length N_g, where N_g is the number of gray levels in the image (Haralick features and texture analysis in general do not traditionally operate over a colored image). In this instance, $N_g = 255$, the standard for gray-level images.

Haralick features require iterating over every pixel in the image. In the co-occurrence matrix, every row represents the gray level of the pixel being examined. Every column is the gray-level of one of four possible neighboring pixels. In fact, Haralick features are built for four different co-occurrence matrices, one for each type of neighboring pixel: one each θ at 0^o, 45^o, 90^o, and 135^o. This θ for neighboring pixels is vaguely similar to that used for edge thinning in Canny edge detection, see section 3.2.1.

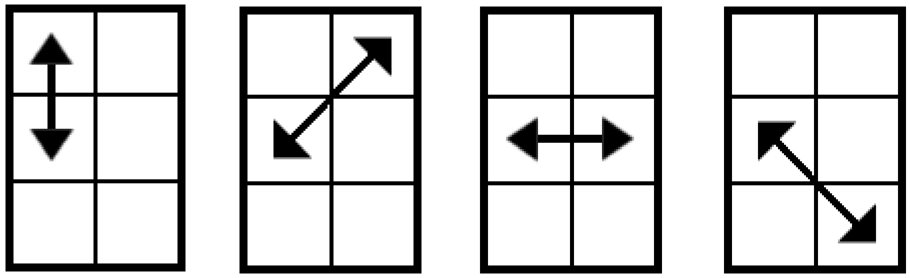


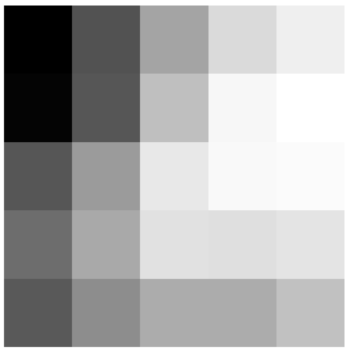
Figure 3.3.1: Potential pixel comparisons for a co-occurrence matrix

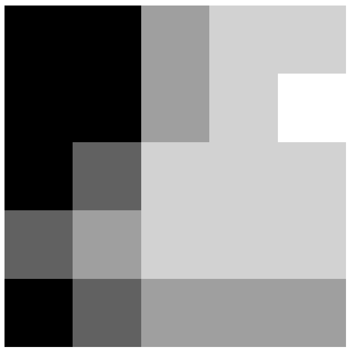
Of the eight possible neighbors of a pixel, the other four or not considered, as after iterating over all pixels in the matrix, they would mirror the first four, with x and y values flipped. No relationships could be garnered from them that could not be garnered from the other four. Even the distinct co-occurrence matrices used, Haralick notes, "are four closely-related measures."

The border case is, as with convolution, a problem. In this case, values are simply not tabulated where there is no pixel-neighbor relationship. The sum value of all cells in the co-occurrence matrix, then, is the image length times image height, minus the image height and/or width, depending on the pixel relationship used.

As an example, consider convolved image g in Figure 1.2.2. When normalized to 5 gray levels, the image has four easily viewable co-occurrence matrices.

While the co-occurrence matrix has a wide variety of applications, it most regarded for its classifying capabilities, using the the 14 summary statistics Haralick makes note of in his first appendix.


(a) image g


(b) g normalized to $N_g = 5$

$$\begin{bmatrix} 0 & 0 & 2 & 3 & 3 \\ 0 & 0 & 2 & 3 & 4 \\ 0 & 1 & 3 & 3 & 3 \\ 1 & 2 & 3 & 3 & 3 \\ 0 & 1 & 2 & 2 & 2 \end{bmatrix}$$

(c) numeric representation of g

$$\begin{bmatrix} 2 & 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 5 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a) 0^o

$$\begin{bmatrix} 1 & 1 & 2 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) 45^o

$$\begin{bmatrix} 3 & 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 3 & 5 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(c) 90^o

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 4 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(d) 135^o

Figure 3.3.3: co-occurrence matrices for each neighbor relationship in g where $N_g = 5$

3.3.2.2 Summary Statistics

14 summary statistics are commonly gathered for each co-occurrence matrix. The features are merely suggestions as to how to summarize a co-occurrence matrix, but they are now used routinely, and the classifying power of each depends on the problem in question. They are as follows:

$p(i, j)$ is the location of an entry in the co-occurrence matrix in row i, column j

$p_x(i)$ is the probability of i in the x column

$p_{x+y}(k)$ is the probability of any value where $i + j$ equals k

N_g is the number of gray levels in the image

Angular Second Moment:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} {p_{i,j}}^2$$

Angular Second Moment increases as any given entry in the co-occurrence matrix does. This means it is largest when the image is most uniform. It is theoretical similar to the 'energy' summary statistic used by the Maastricht group for Gabor wavelets [9].

Contrast:

$$\sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p_{i,j} \right\}, |i-j|$$

Contrast measures the degree of distinction between neighboring pixels. Pixels that move from pure black to pure white very quickly are prone to have higher values in this feature.

Correlation:

$$\frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (ij) P_{i,j} - \mu_x \mu_y}{\sigma_x \sigma_y}$$

where μ_x, μ_y, σ_x, and σ_y are the means and standard deviations of p_x and p_y

Correlation shows the linear dependency of a gray level. If certain gray levels appear only when routinely next to other gray levels, correlation will be high.

Sum of Squares- Variance:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i-\mu)^2 p_{i,j}$$

Variance measures the dispersion of values across the image. For example, a black and white image will have a relatively low variance (assuming N_g is 255), because there are few to no occurrences of other values.

Inverse Difference Moment:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{1}{1+(i-j)^2} p_{i,j}$$

Inverse Different Moment is a measure of homogeneity. If the co-occurrence matrix is has probabilities close to the diagonal, it will be high. Pixels must occur near pixels of the same value.

Sum Average:

$$\sum_{i=2}^{2N_g} i p_{x+y}(i)$$

where x and y are the coordinates of an entry in the occurrence matrix and $p_{x+y}(i)$ is the probability co-occurrence matrix coordinates sum to $x + y$.

In general, 'Sum' and 'Difference' characteristics (Sum Entropy, Difference Variance, etc.) measure mentioned characteristics *across gray levels*, that to say *rows and columns* of the co-occurrence matrix, rather than in regards to specific pixel adjacencies. This gives these statistics a greater generalizing power, and they may serve better as a summary statistic in co-occurrence matrices with high N_g levels.

Sum Variance:

$$\sum_{i=2}^{2N_g} (i - f_6)^2 p_{x+y}(i)$$

N.B. In Haralick's original paper, variance uses the eighth feature rather than the sixth. This is assuredly a mistake, and has been corrected here.

Sum Entropy:

$$-\sum_{i=2}^{2N_g} p_{x+y}(i) log\{p_{x+y}(i)\} = f_s$$

Entropy is difficult to define, but it effectively measures the information density of an image. It is similar to variance, but distinct. To reuse the black and white example, if an image is binary white noise, the entropy will be high, because although not all values are being used, values are distributed in a complex way.

Entropy:

$$-\sum_{i=0}^{N_g-1} \sum_{j=0}^{N-1} p_{i,j} log\{p_{i,j}\}$$

Difference Variance:

$$\sum_{i=0}^{N_g-1} i^2 p_{x-y}(i)$$

Difference Entropy:

$$-\sum_{i=0}^{N_g-1} p_{x-y}(i) log\{p_{x-y}(i)\}$$

Measure of Correlation 1:

$$\frac{HXY - HXY1}{max(HX, HY)}$$

H here represents entropy, so HXY is literally the above entropy feature, HX and HX the entropy of p_x and p_y, respectively, and $HXY1 = -\sum_{i=0}^{N_g-1}\sum_{j=0}^{N-1} p_{i,j} log\{p_x(i)p_y(j)\}$

Measure of Correlation 2:

$$(1 - exp[-2(HXY2 - HXY)])^{1/2}$$

where $HXY2 = -\sum_{i=0}^{N_g-1}\sum_{j=0}^{N-1} p_x(i)p_y(j) log\{p_x(i)p_y(j)\}$

The last characteristic, known as the **Maximal Correlation Coefficient**, depends on drawing eigenvalues from the matrix, which is often computationally inefficient to compute. It has been left out for this reason.

Accuracy rates in classification for all Haralick features considered collectively can be found in table 4.3.1.

3.3.3 Gabor Filters

Gabor filters are the second method of texture analysis to be considered. They operate via the process of convolution, as discussed the section 1.2.2. Gabor filters are particularly useful for texture analysis because of the method of kernel generation, and the use of multiple kernels.

3.3.3.1 The Gabor Wavelet

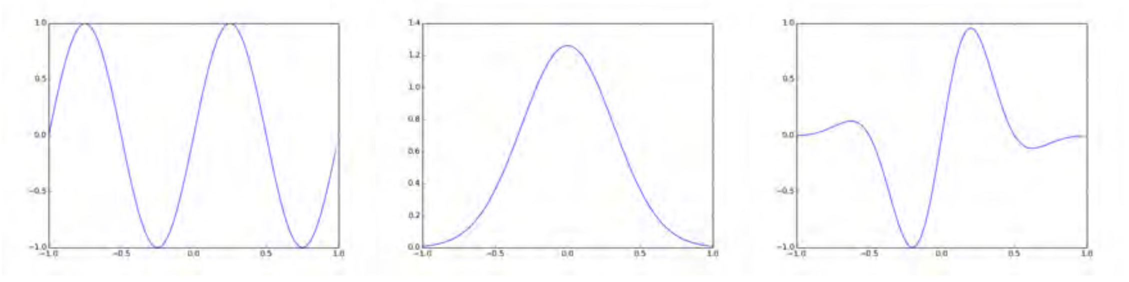
A wavelet is a signal that has a defined beginning and ending of zero. A sine wave, for example, is not a wavelet, but a partition of a sine wave where the signal both begins and ends at zero (for instance, [0,π]) is.

As discussed in section 1.2.1, a wavelet must be discrete in the instance of computational representation. In this regard, a wavelet can be represented as an array of dimension n. A

Gabor wavelet is simply a given signal multiplied by a Gaussian window, which is a given range over any normal distribution.

$$f(x) = e^{-(x-x_0)^2/a^2} e^{-ik_0(x-x_0)} \tag{3.3.1}$$

Formula for a Gabor wavelet



(a) sine wave (b) Gaussian window (c) Gabor wavelet

Figure 3.3.4: Visualization of a Gabor wavelet and its component parts

Gabor filters are two-dimensional kernels, which are seperable into two of these one-dimensional Gabor wavelets. This underlying theory is key in generating a kernel for the Gabor filter, but Gabor filters can be more practically thought of as edge detectors which specifically react to edges of their given orientation and frequency.

3.3.4 Kernel Composition

The two-dimensional Gabor kernel is modeled by a theta, a frequency, and a sigma for the x and y directions. Varying each changes the kernel type drastically. Each Gabor kernel is not unlike a brushstroke, having direction and size.

Theta will vary the kernel orientation. Frequency controls for how often the signal occurs. A high frequency yields a signal being repeated several times in a single kernel. Sigma is the standard deviation of the sigma. A high sigma will yield a large Gabor kernel.

3.3.4.1 Kernel Bank

A key aspect of Gabor filters is that, commonly, more than one are used. A series of Gabor kernels are assembled in what is collectively referred to as the *kernel bank*.

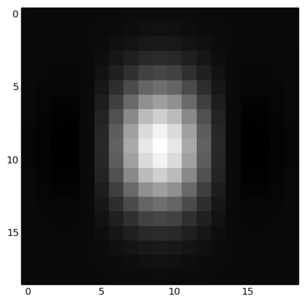


Figure 3.3.5: A simple Gabor kernel, where $\theta = 0$, $\sigma = 1$, and frequency is 3

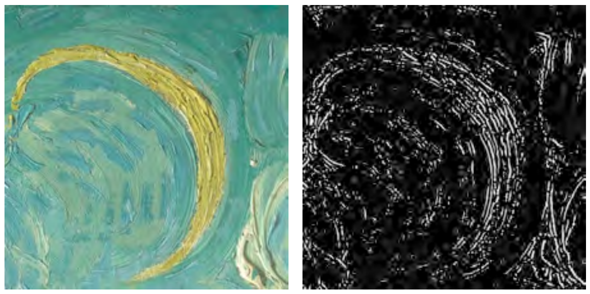
The reasoning behind this is that the various qualities of a Gabor kernel react with an image in different ways. A single kernel may hardly be descriptive of an entire image.

Figures 3.3.6 and 3.3.7 demonstrate Gabor filters of different orientation operating over a patch taken from Van Gogh's *Cypresses*. The patch effectively demonstrates the wide variety of strokes that can be found in one painting, and the inability of any one Gabor filter to adequately summarize any part of a painting.

The number of Gabor kernels necessary to reasonably define an image is never precisely clear. The kernel bank containing the kernels used in this paper contains sixteen kernels, with thetas varying in four directions, and two variable frequencies and sigmas. The kernel bank is illustrated in figure 3.3.8.

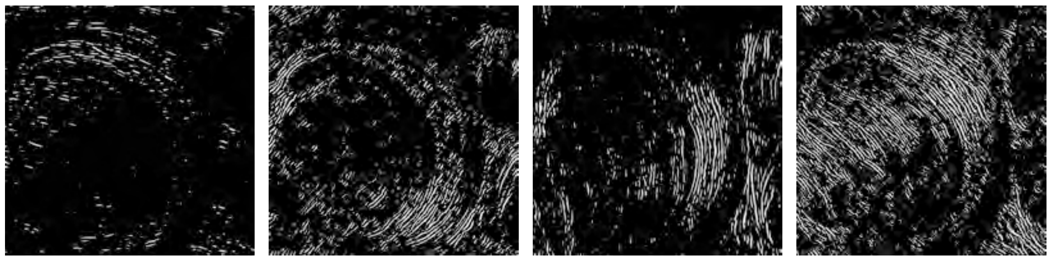
3.3.4.2 Summary Statistics

As with co-occurence matrices, the image generated by Gabor filters cannot be used for classification in and of themselves. Luckily, the summary statistics for an image filtered by a Gabor kernel are much simpler. While a variety of valuable statistics can be taken over the image, it is usually effective enough just to consider the mean and variance of the values in the given image. Notably, these statistics are gathered over the image as filtered by each of the sixteen kernels, for thirty-two features overall (as compared to the 52 features given by Haralick).



(a) Original image (b) Combined θ

Figure 3.3.6: Original and edge detected image



(a) $\theta = 0^0$ (b) $\theta = 45^0$ (c) $\theta = 90^0$ (d) $\theta = 135^0$

Figure 3.3.7: Gabor filters operating over four common orientations

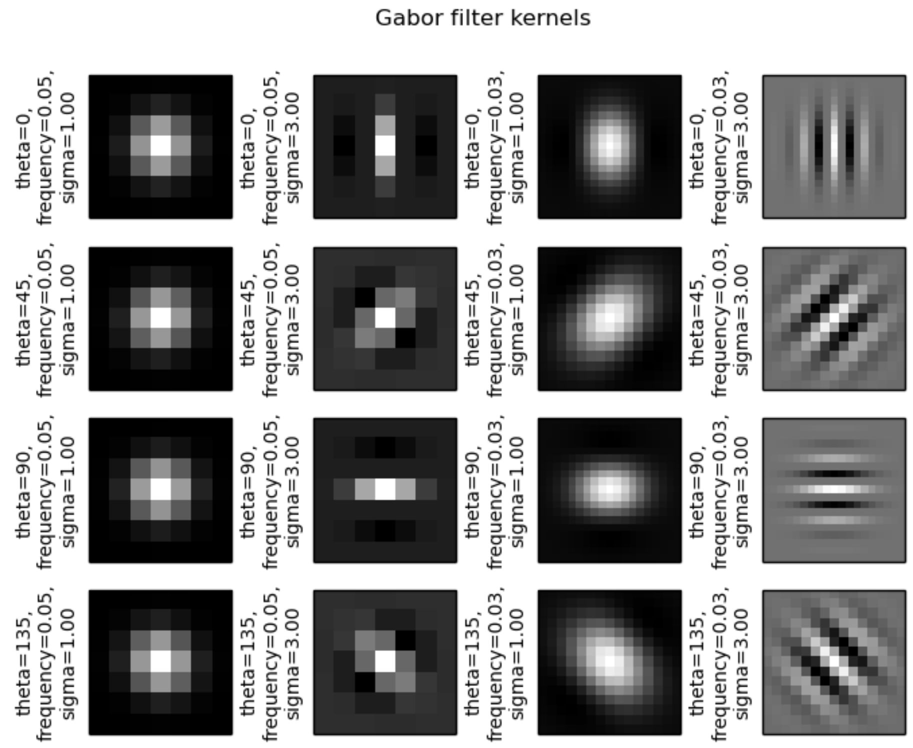

Gabor filter kernels
theta=0, frequency=0.05, sigma=1.00
theta=0, frequency=0.05, sigma=3.00
theta=0, frequency=0.03, sigma=1.00
theta=0, frequency=0.03, sigma=3.00
theta=45, frequency=0.05, sigma=1.00
theta=45, frequency=0.05, sigma=3.00
theta=45, frequency=0.03, sigma=1.00
theta=45, frequency=0.03, sigma=3.00
theta=90, frequency=0.05, sigma=1.00
theta=90, frequency=0.05, sigma=3.00
theta=90, frequency=0.03, sigma=1.00
theta=90, frequency=0.03, sigma=3.00
theta=135, frequency=0.05, sigma=1.00
theta=135, frequency=0.05, sigma=3.00
theta=135, frequency=0.03, sigma=1.00
theta=135, frequency=0.03, sigma=3.00

Figure 3.3.8: Gabor filter kernel bank

4
Classification

After the feature set has been gathered, each feature is represented as a series of numeric inputs. This means each color bin, Haralick or Gabor summary statistic, and piece of edge information is considered separately in the classification process.

(a) Sample from Cezanne's *The Artist's Uncle*

[41086 193006 7723 2050 2431 2351 533 307 185 55 95 178]

(b) Array of binned hue values from sample

41086	193006	7723	2050	2431	2351	533	307	185	55	95	178
Red	*Ora.*	*Yel.*	*Char.*	*Green*	*Spr.*	*Cyan*	*Azu.*	*Blue*	*Vio.*	*Mag.*	*Rose*

(c) In classification, each value is considered individually with the attached label

Figure 4.0.1: Example of individual features for classification

The actual implementation of random forests is relatively simple, but the theoretical underpinnings are not.

4.1 Random Forests

Random forests are remarkably resilient and successful in classification or regression analysis. They operate by generating a series of decision trees, and the decision trees act collectively to pick a likely candidate class for the feature vector.

4.1.1 Decision Tree

The decision tree is the fundamental component of a random forest. Decisions about the data are made at each edge, and each leaf has a label associated with it that concludes to classification.

It should be known that decision trees can also generate numeric outcome rather than enumerative. This is known as a regression tree. However, grouping paintings by artist depends on class specification. This is called a classification tree.

4.1.2 Bootstrapping

Bootstrapping is a relatively simple but pivotal process by which random forests select a sampling of the data to use in constructing a given decision tree within that forest.

Effectively, bootstrapping is sampling with replacement. From data set $\mathcal{L}$ containing N samples in the form of $\{(x_n, y_n), n = 1...N\}$, where x and y are feature vectors and class labels, respectively, draw $\mathcal{L}^{(B)}$ samples. $\mathcal{L}^{(B)}$ is sampled uniformly and with replacement.

This model can help repeatedly identify strong classifier variables from x, and can help strongly improve unstable classifiers, like when the number of variables in x is greater than N, which occurs often for painting analysis where painters have minimal or dubiously authentic output. Generally, it helps to reduce variance and prevent overfitting in the data as well.

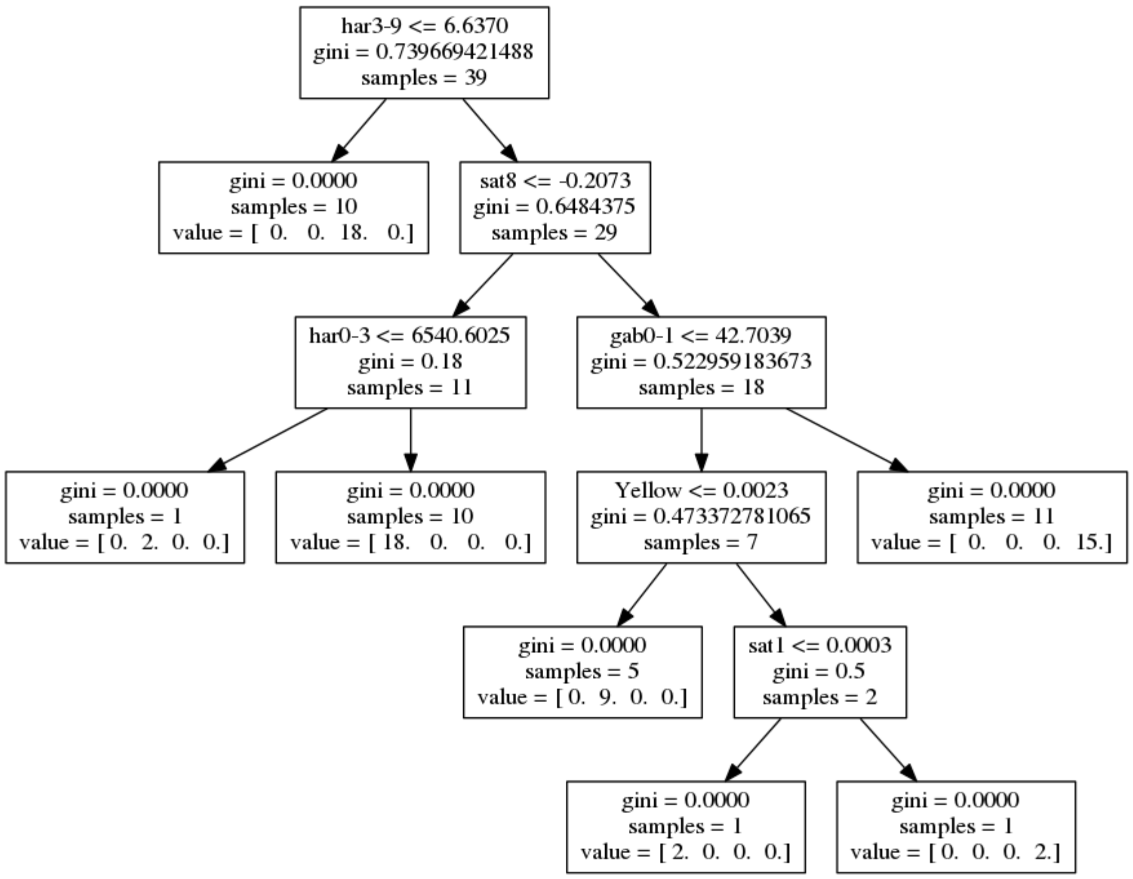


Figure 4.1.1: Sample Decision Tree

Accurate measures of desirable size for $\mathcal{L}^{(B)}$ and number of samples to draw can be obtained experimentally [3]. Traditionally, $\mathcal{L}^{(B)}$ is most dependant on the size of the data set and the number of decision trees generated for the forest. Decision trees in a random forest will not share the same $\mathcal{L}^{(B)}$, though both values will often be close.

4.1.3 Arboreal Voting

Each decision tree 'votes' for its outcome on the given input x_n, y_n. The most voted outcome is that selected by the random forest.

In the event decision trees voted on a numeric output, a simple averaging would occur. It is not necessary to consider this case in artist classification.

4.2 A Collection Schema

In addition to the feature collection set, there are a variety of methods for collecting these features. Different methods measure for the same qualities is distinct ways, yielding different results. Since the classifier acts on these results, classification strength can vary dramatically, although each schema is expected to perform with reasonable success . Three methods are here considered: a generic model, where features are gathered across an entire painting, a feature-dense model, where features are calculated across four parts of the image and concatenated into a significantly larger feature vector, and a feature-specific model, where features are calculated for small subsets of a given image, and then averaged together in the final feature vector. 200 random forests were designed to fit the data, each considering all available features at any given step in the decision tree creation. Their collective performance is considered. Cross-validation was done to each to access accuracy of categorization, using 5 folds.

4.2.1 Generic Model (The Entire Painting)

For the generic model, all features were found over an entire painting. The implication of this is pretty clear. While color values should remain relatively constant (albeit scaled up to a full painting) relative to the feature-specific model, edge lengths will be larger and edge counts smaller. The impact of texture features will be seen. In the four-way classification, the random forest routinely performs with 70% accuracy. Performance in all two-way problems can be found on Figure 4.2.1 on page 46.

When moving onto the two-class problem, the sample size may get small enough that overfitting becomes a serious issue, especially when there is a clear indicator variable or variables, as in most of the Rembrandt-based classification. This consideration remains across models.

4.2.2 Feature-Dense Model (Quadrants)

In the feature-dense model, every painting was divided into four equivalent parts. This means that partitions analyzed may be of a different size across paintings. All color values are normalized, but there is no clear way to account for how the texture features may differ in classifying ability because of this, or if they will differ at all. It is proposed that feature density may compensate for this, if there is enough correspondence between the added features.

The feature-dense model uses a 472-feature vector, four times larger than the 118-feature the other two models use.

The feature-dense model performs better, averaging 73% accuracy. The in-depth Figure 4.2.2 can be found on page 46.

4.2.3 Feature-Specific Model (Patches)

The feature-specific model deconstructs every painting into a small number of 512x512 patches and extracts features; features are later averaged together over all partitions of the image. The resulting feature vector is of the same size of the generic model, but represents an 'average' 512 x 512 pixel block of the picture. Images are usually high-resolution enough to offer anywhere from 25 to 35 blocks.

This form of feature collection is a much simplified version of that proposed by the Penn State Group [9]. The premise is that at lower resolution pieces of an image allow for more accurate measure of dissimilarity, improving the capacities of texture analysis and any distance based measure. Color values are unaffected, and edge values should be impacted in a roughly consistent way across all paintings, as certain edges once read as continuous will be cut off by the cropped 512 x 512 patch.

The feature-specific model performs best in the four-way problem, and is the model used, when not otherwise indicated, for other classification statistics in this paper. Its general performance can be found on Figure 4.2.3 on page 46.

The feature-specific model notably performs worse in certain classifications, but provides a more definitive overall model, performing well in all circumstances, while being less likely to overfit to the given data.

4.3 Feature Comparison

Having tested various classification types, it is of utility to see more information on specific classifiers. Namely, the value of each specific classifier should be clarified to ensure none provide a useless signal or act as noise over the data set. It should also be considered which specific features of classifying groups are most important.

4.3.1 Individual Features As Classifiers

It is important that as many features as possible help in classification, with the exception of texture analysis, where it is anticipated both feature account for the same phenomenon, with one eclipsing the other in terms of useful information.

It is necessary that individual features act in and of themselves as a better than random classifier (about 25% in the 4-way problem), otherwise they are likely only adding noise to the data set. Figure 4.3.1 demonstrates the classification prowess of each data category in isolation (along the diagonal), and in union with any of the other data categories.

Much is confirmed from the data. All forms of feature collection prove themselves to be better-than-random classifiers. As expected, one method of texture analysis is largely eclipsed by the other. Gabor filters proves to be a worse indicator than Haralick features, while not capturing for anything new when the features are combined. Surprisingly, sat-

Class Problem	**Accuracy**	**Std. Dev.**
4-way	70%	10%
Cezanne-Monet	73%	10%
Cezanne-Rembrandt	94%	4%
Cezanne-Van Gogh	78%	10%
Monet-Rembrandt	97%	4%
Monet-Van Gogh	55%	17%
Rembrandt-Van Gogh	100%	00%

Figure 4.2.1: Generic Model

Class Problem	**Accuracy**	**Std. Dev.**
4-way	74%	4%
Cezanne-Monet	78%	10%
Cezanne-Rembrandt	100%	0%
Cezanne-Van Gogh	82%	7%
Monet-Rembrandt	93%	10%
Monet-Van Gogh	63%	11%
Rembrandt-Van Gogh	100%	00%

Figure 4.2.2: Feature-Dense Model

Class Problem	**Accuracy**	**Std. Dev.**
4-way	79%	7%
Cezanne-Monet	72%	9%
Cezanne-Rembrandt	94%	7%
Cezanne-Van Gogh	89%	11%
Monet-Rembrandt	99%	2%
Monet-Van Gogh	78%	13%
Rembrandt-Van Gogh	100%	00%

Figure 4.2.3: Feature-Specific Model

4-Way	**Hue**	**Sat.**	**Value**	**E. Count**	**E. Length**	**Haralick**	**Gabor**
Hue	46%	-	-	-	-	-	-
Sat.	43%	34%	-	-	-	-	-
Value	64%	61%	57%	-	-	-	-
E. Count	57%	49%	58%	52%	-	-	-
E. Length	57%	49%	68%	63%	41%	-	-
Haralick	65%	72%	65%	64%	78%	66%	-
Gabor	63%	63%	60%	60%	75%	65%	58%

Figure 4.3.1: Accuracy of all pairings of features used in classification

uration values are largely without use, and edge length and Haralick features capture a large portion of classifying power.

The results imply saturation values and Gabor values to be noise, and unsurprisingly, when removed classification increases to just over 80%. The significantly smaller 76-feature vector in this instance will be used when measuring feature importance.

4.3.2 Feature Importance

Random forests are also useful because they can give you measures of feature importance. Feature importance can be measured in several ways. After a forest is fit to the data, all samples in the training set are designated a value for *out-of-bag error*, or the likelihood that they will be incorrectly classified by a tree in the forest that did not consider this sample when generating.

For any given feature, the out-of-bag error is again measured with the variable in question "noised up," or given artificially increased importance in the decision trees of the random forest [4]. The difference between these two out-of-bag differences is the importance of the variable in question (with all importances being normalized). It is referred to as "permutation importance."

A more common and related measure of feature importance is simply the raw importance, or the expected fraction of the sample any feature contributes to across all decision trees. This is the importance estimate used here.

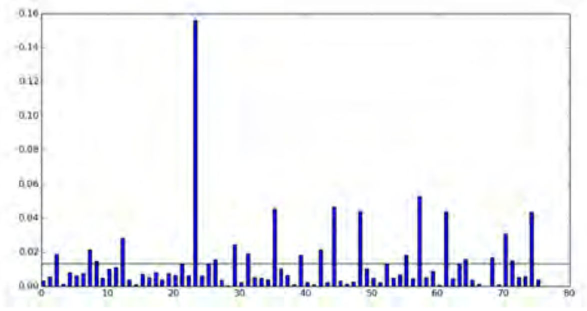


Figure 4.3.2: Graph of all 76 features' importance measurements for one random forest

The running line represents the average value of all features. Along the x-axis, ranges 0-11, 12-21 represent hue and value bins, as explained in section 3.1.3. Values 22 and 23 represent edge count and length. All the following values represent Haralick features.

Feature importances become more clear if we graph and label the top ten features in the importance array.

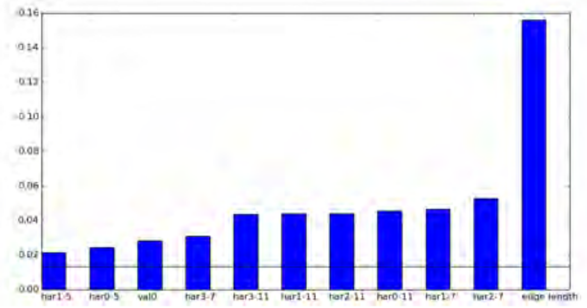


Figure 4.3.3: Features with the highest importance measurement

Edge length is the dominant important variable. Even though in 4.3.1 it serves as one of the worst individual classifiers, it contains the most unique information.

Two of Haralick's features specifically routinely serve as excellent classifiers as well, across all four co-occurrence matrices that are constructed. These are Sum Entropy and the first Measure of Correlation, respectively.

4.3.3 The Strong Classifiers

Considering only these particularly salient classifying variables allows an interesting study to take place, with easy visualization that couldn't be obtained from the normal feature set, given its high dimensionality.

As suggested by figures 3.2.3 and 4.3.1, edge length is, of its own accord, a relatively poor classifier. But when visualized with either of the two well-classifying Haralick features, the value it offers becomes much more clear.

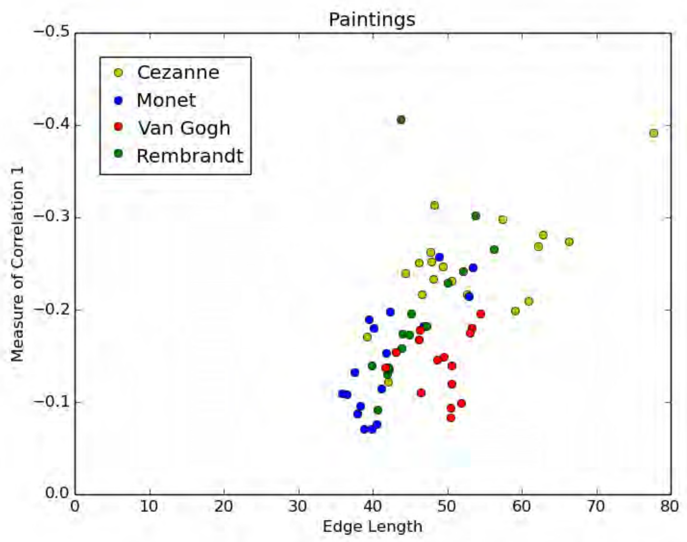


Figure 4.3.4: Measure of Correlation as a function of Edge Length

Edge length helps to clearly delineate the normally quite difficult Van Gogh-Monet classification problem, while the information measure of correlation helps to distinguish Cezanne. Rembrandt is, notably, spread across all the other data. The Rembrandt classification problem seems largely accounted for by Sum Entropy.

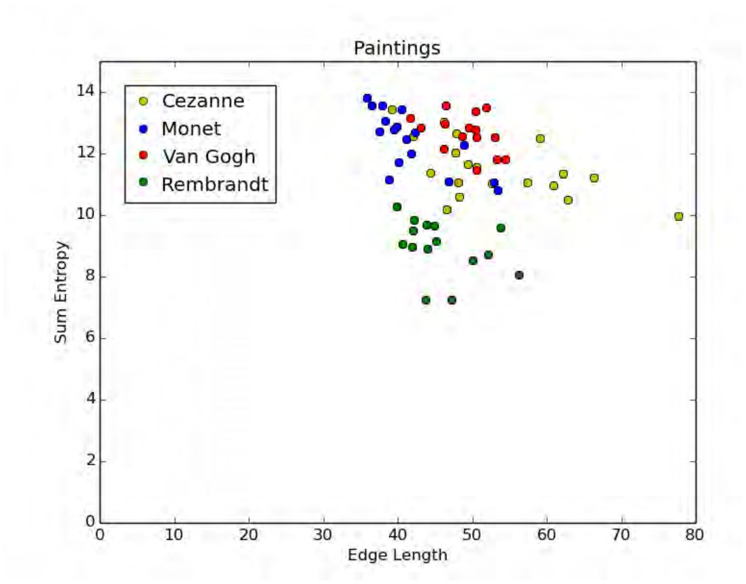


Figure 4.3.5: Sum Entropy as a function of Edge Length

While the information measure of correlation and sum entropy do well to delineate Cezanne and Rembrandt, the Van Gogh-Monet problem becomes confused.

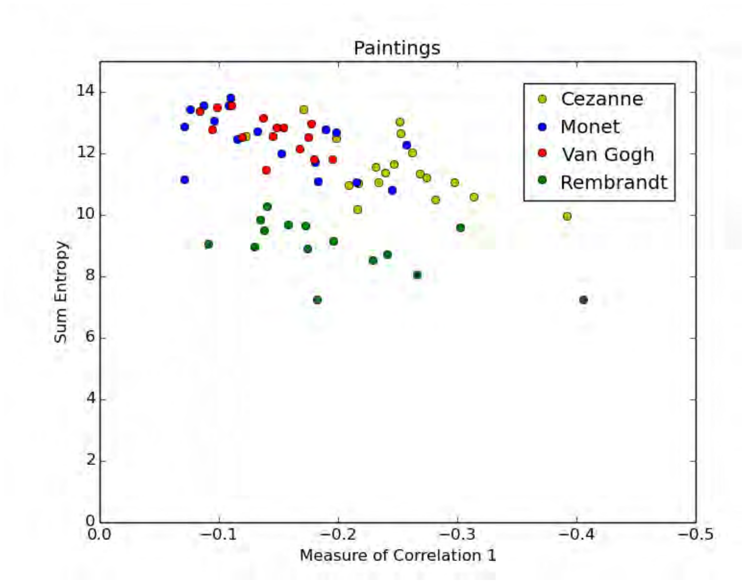


Figure 4.3.6: Sum Entropy as a function of Measure of Correlation

All three figures have the added benefit of visually explaining where confusions and outliers in the data lie. Most artist groupings have at least one painting which, while not necessarily being incorrectly classified, appear distinct from the other data of its set. Discovering what these paintings are provides a painting from the artist's catalogue which is apparently distinct from the others. Bare in mind that these paintings are not distinct in the more normal human registers, primarily that of color, but are rather distinct in the texture and shape of the the figures in the image.

To find these images, the measures across all three graphs were normalized to one, and the Euclidean distance between each painting and the mean value of all paintings in that artist set were calculated. The painting with the largest distance is considered most exceptional.

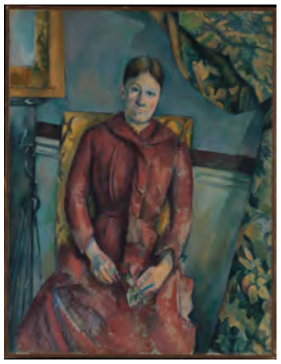
In all graphs 4.3.4, 4.3.5, and 4.3.6, there is a clear outlier in the Cezanne group. This painting is Cezanne's *Madame Cezanne in a Red Dress.*

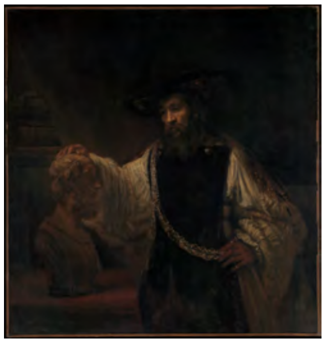
Rembrandt also has a notable outlier in graphs 4.3.4 and 4.3.6, along the Measure of Correlation statistic. This painting is Rembrandt's *Aristotle with a Bust of Homer.*

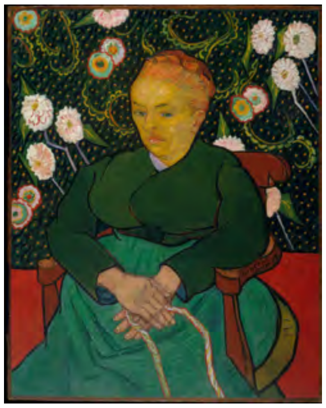
Information on Monet is rather well dispersed, and thus difficult to classify. In graph 4.3.4, there is a suggestion of three mild outliers. The greatest among these is *Camille Monet on a Garden Bench.*

Van Gogh is very well clustered across all graphs, suggesting a style both well-defined and idiosyncratic. Nonetheless, the closest painting to an outlier in his data set is *La Berceuse.*

Abnormal paintings could be determined for any of the characteristics used by the classifier, but since they have been determined given the three routinely strongest classifying attribute, it is these paintings that are exceptional in regards to the most characteristic measures of each artist.


(a) *...in a Red Dress*


(b) *Aristotle*


(c) *La Berceuse*

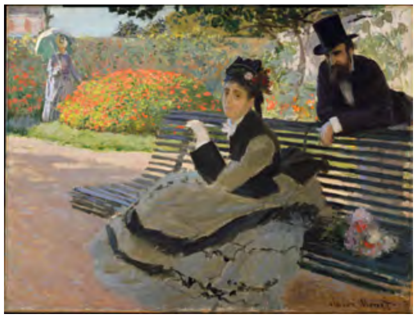

(d) *...on a Garden Bench*

Figure 4.3.7: Relatively abnormal paintings from each of the four artists

5
Conclusions

It is clear in both this experiment and other literature on the topic that significantly accurate painting classification is possible. Access to large and relevant feature sets alongside advanced classification algorithms are making the task progressively easier, and so researchers continue to move steadily towards forgery detection and empirically determined catalogue raisonnés.

A difficulty in this task is that human performance can never reasonably be 100% accurate. A success rate of above 80% is beyond expectations, and shows the value of even basic feature collection, as well as texture analysis.

The task can also serve to empirically illuminate distinctions between artists, and within an artist's catalogue. Specific works can be distinguished for their abnormality, and suggest a change in style or painting application.

5.1 Texture for Painting Classification

Texture is of clear value when considering a painting's authorship. Research even begins to suggest texture analysis to be even more essential to the question of authorship than more readily apparent characteristics, like shape and color, as both Haralick features and Gabor filters performed better in isolation than any other feature collected (cf. 4.3.1).

Haralick features prove themselves as better classifiers than Gabor filters in this specific four way problem, but it is important to note potential reasons for this, especially after Gabor filters have proven to perform quite well in other instances [8] [9].

The size and type of Gabor kernels found in the Gabor kernel bank is one clear justification for this. Larger and more varied kernels may yield more useful classifying data. The summary statistics used may be another answer. As noted in section 2.3, the Maastricht group found significant success with the 'energy' statistic, but only in distinguishing work with an artist's catalogue, not between artists. Other summary statistics should be considered as well.

Haralick features also raise the question of adaptability. While the two features of Measure of Correlation 1 and Sum Entropy serve well to classify the given four artists, does this remain true for new works, and new artists? The nature of Haralick features suggests they are adaptable to a wide variety of instances, but in higher and different class problems, they may show themselves to fail. Nonetheless, Haralick features should likely be considered in basic classification problems before Gabor filters.

5.2 Future Work

Given that even human completion of the painting classification task will never be perfect, there is always room for continued research in the field.

Forgery detection and genre classification has already been suggested and tackled, with a degree of success. A particularly valuable insight would be in the form of optimization of feature classification strength, and feature collection models. Section 4.2 suggests the 'patchwork' approach is best for analysis, but at what scale do telling features emerge? Would it be helpful, for example, to consider 128 x 128 pixel patches rather than the given 512 x 512? In terms of feature collection, what, empirically, is the optimal sigma for blurring during edge detection? What is the optimal size and variation to find in the Gabor kernel bank?

For brushstroke analysis, it is certainly of most value to consider the highest resolution images. But for qualities of color, edge, and a more general texture for broader classification, it may be of more value to consider the responsiveness of lower resolution representations of the image, or lower representation patches. The Laplacian or Gaussian image pyramid is suggested, whereby each image is blurred with that pyramid's respective filter, before the image resolution is halved, until no information of value can reasonably be obtained.

More than inventing new methods of classification, this research would tellingly explain how to best collect feature information using known techniques.

6 Appendix

6.1 Data Format

Gathering features over high-resolution paintings is a time-consuming task. The process itself can be hastened by parallelizing the job, but it is still a waste of time to retabulate statistics for a small change or to create a graphic. For this reason, data was written to a text file, which was then read in by a seperate program for graphing and classification purposes.

Every comma(,) represents a new array, while every space represents a new value in the array. The sequence of the data for each analyzed element was as such: For labelling, the artist, the painting, and the partition of the painting (as determined by the directory the painting file was located at) would begin the data. The numeric data would follow: hue bins, saturation bins, value bins, edge information, an array each for the thirteen Haralick features from each co-occurrence matrix, and an array each for each of the two summary statistics from the sixteen Gabor filters.

In the instance of the feature dense (quadrant) model, the numeric data would appear 3 more times, each representing a different quadrant of the given picture.

6.2 Data Sample

The following is a data sample from the feature specific (patches) model of classification, as it was the most successful method. Also shown is the breakdown of the data: the specific value, the significance of that data, and how it is labelled for feature importance graphing. Feature importance labels are often not entirely self-descriptive for the sake of brevity, as these labels were used to label the x-axis in Figure 4.3.3

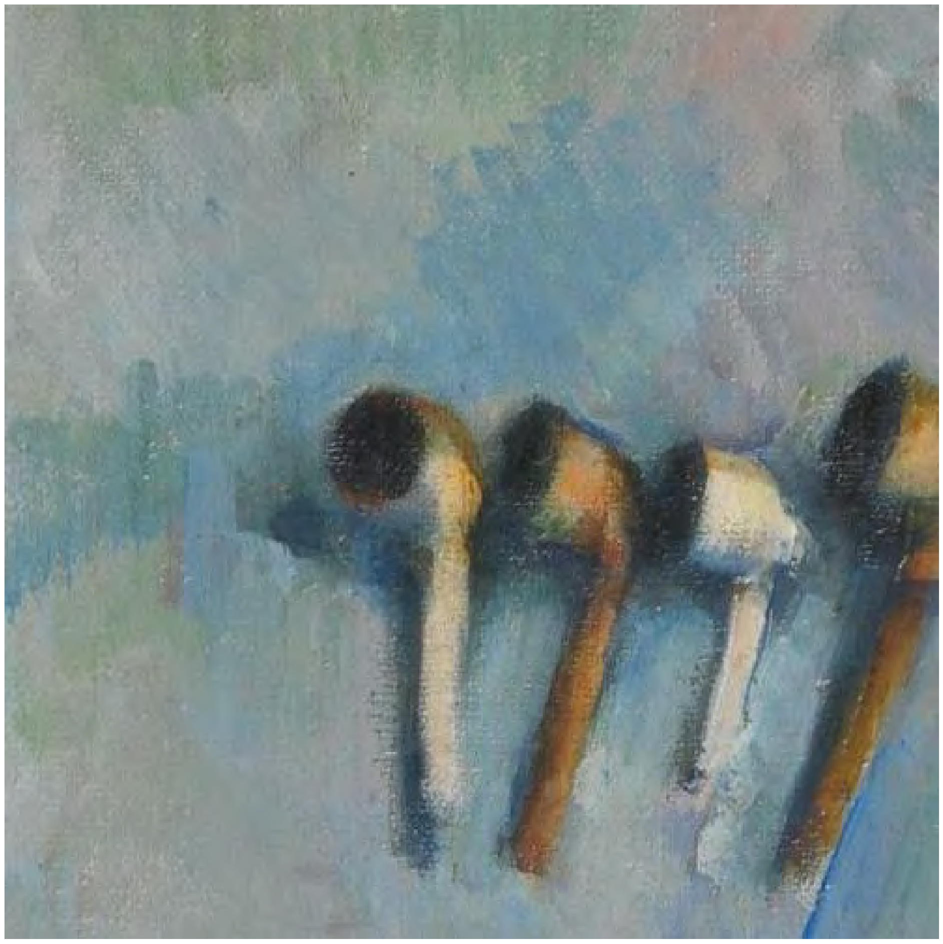


Figure 6.2.1: Sample Image from Cezanne *The Card Players*

Cezanne, TheCardPlayers, TheCardPlayers-5.jpg, 9965.0 15517.0 5521.0 14050.0 28877.0 51401.0 114389.0 8170.0 1256.0 220.0 107.0 527.0, 106365.0 71656.0 45412.0 17099.0 4884.0 3227.0 1270.0 87.0 0.0 0.0, 0.0 3161.0 10297.0 13784.0 16730.0 114909.0 85817.0 4709.0 593.0 0.0, 56, 67.3035714286, 0.00129129158517 43.7771422846 0.9726282171 799.676485157 0.253600529958 266.546857715 3154.92879834 7.31863387617 10.8121925641 0.000470838516473 3.66964213557 -0.296717583836 0.98837221376, 0.00105617486853 67.2912397942 0.957988562297 800.868090621 0.209530286534 266.501443769 3136.18112269 7.30549348478 11.0982683699 0.000384127938241 3.95594347081 -0.252149452314 0.979451134551, 0.00157654353999 30.503511022 0.980940589289 800.221777193 0.298030637423 266.542533066 3170.38359775 7.32899668582 10.5418480673 0.00057197799843 3.40851201006 -0.339831749943 0.993299440861, 0.00107190739164 58.4635162108 0.963499880732 800.867468145 0.213872411241 266.501427705 3145.00635637 7.30895674208 11.0420581836 0.000397026105889 3.88833877836 -0.261001015334 0.981656320455, 0.499142295394 0.0106388381711, 0.33670532125 0.00468260706386, 0.152590494966 0.00105987352289, 0.000502239456194 8.54221417217e-06, 0.49914672401 0.0106368002851, 0.339621190183 0.00470670196224, 0.152804272225 0.00104881742826, 8.08921440868e-05 5.8152160312e-06, 0.499142295394 0.0106320597617, 0.33670532125 0.00456344304993, 0.152590494966 0.00103242625921, 0.000502239456194 7.72652861089e-06, 0.49914672401 0.0106344078758, 0.339621190183 0.00467721292324, 0.152804272225 0.00103757783919, 8.08921440869e-05 3.91802350991e-06

Figure 6.2.2: Sample Image Data

Artist: Cezanne
Painting: TheCardPlayers
Partition: TheCardPlayers-5
Hue:

9965	15517	5521	14050	28877	51401	114389	8170	1256	220	107	527
Red	*Ora.*	*Yel.*	*Char.*	*Green*	*Spr.*	*Cyan*	*Azu.*	*Blue*	*Vio.*	*Mag.*	*Rose*

The labels for feature importance are the same as the meanings above

Saturation:

106365	71656	45412	17099	4884	3227	1270	87	0	0
$0-09$	$10-19$	$20-29$	$30-39$	$40-49$	$50-59$	$60-69$	$70-79$	$80-89$	$90-99$
$sat0$	$sat1$	$sat2$	$sat3$	$sat4$	$sat5$	$sat6$	$sat7$	$sat8$	$sat9$

Value:

0	3161	10297	13784	16730	114909	85817	4709	593	0
$0-09$	$10-19$	$20-29$	$30-39$	$40-49$	$50-59$	$60-69$	$70-79$	$80-89$	$90-99$
$val0$	$val1$	$val2$	$val3$	$val4$	$val5$	$val6$	$val7$	$val8$	$val9$

Edge Count:
56
E. Count
Edge Length:
67.3035714286
E. Length
Haralick Features:
Co-Occurence Matrix 0 (Left-Diagonal):

0.00129129158517	43.7771422846	0.9726282171	799.676485157	0.253600529958
ASM	*Contrast*	*Correlation*	*Variance*	*IDM*
$har0-0$	$har0-1$	$har0-2$	$har0-3$	$har0-4$
266.546857715	3154.92879834	7.31863387617	10.8121925641	0.000470838516473
Sum Average	*Sum Variance*	*Sum Entropy*	*Entropy*	*Difference Variance*
$har0-5$	$har0-6$	$har0-7$	$har0-8$	$har0-9$

3.66964213557	−0.296717583836	0.98837221376
Difference Entropy	*Measure of Correlation* 1	*Measure of Correlation* 2
$har0-10$	$har0-11$	$har0-12$

Co-Occurence Matrix 1 (Left-Right):

0.00105617486853	67.2912397942	0.957988562297	800.868090621	0.209530286534
ASM	*Contrast*	*Correlation*	*Variance*	*IDM*
$har1-0$	$har1-1$	$har1-2$	$har1-3$	$har1-4$
266.501443769	3136.18112269	7.30549348478	11.0982683699	0.000384127938241
Sum Average	*Sum Variance*	*Sum Entropy*	*Entropy*	*Difference Variance*
$har1-5$	$har1-6$	$har1-7$	$har1-8$	$har1-9$

3.95594347081	−0.252149452314	0.979451134551
Difference Entropy	*Measure of Correlation* 1	*Measure of Correlation* 2
$har1-10$	$har1-11$	$har1-12$

Co-Occurence Matrix 2 (Up-Down):

0.00157654353999	30.503511022	0.980940589289	800.221777193	0.298030637423
ASM	*Contrast*	*Correlation*	*Variance*	*IDM*
$har2-0$	$har2-1$	$har2-2$	$har2-3$	$har2-4$
266.542533066	3170.38359775	7.32899668582	10.5418480673	0.00057197799843
Sum Average	*Sum Variance*	*Sum Entropy*	*Entropy*	*Difference Variance*
$har2-5$	$har2-6$	$har2-7$	$har2-8$	$har2-9$

3.40851201006	−0.339831749943	0.993299440861
Difference Entropy	*Measure of Correlation* 1	*Measure of Correlation* 2
$har2-10$	$har2-11$	$har2-12$

Co-Occurence Matrix 3 (Right-Diagonal):

0.00107190739164	58.4635162108	0.963499880732	800.867468145	0.213872411241
ASM	*Contrast*	*Correlation*	*Variance*	*IDM*
$har3-0$	$har3-1$	$har3-2$	$har3-3$	$har3-4$
266.501427705	3145.00635637	7.30895674208	11.0420581836	0.000397026105889
Sum Average	*Sum Variance*	*Sum Entropy*	*Entropy*	*Difference Variance*
$har3-5$	$har3-6$	$har3-7$	$har3-8$	$har3-9$

3.88833877836	−0.261001015334	0.981656320455
Difference Entropy	*Measure of Correlation* 1	*Measure of Correlation* 2
$har3-10$	$har3-11$	$har3-12$

Gabor Kernel:

$\theta = 0$, freq. = .05, $\sigma = 1$:

0.499142295394	0.0106388381711
Mean	*Variance*
$gab0-0$	$gab0-1$

$\theta = 0$, freq. = .05, $\sigma = 3$:

0.33670532125	0.00468260706386
Mean	*Variance*
$gab1-0$	$gab1-1$

$\theta = 0$, freq. = .03, $\sigma = 1$:

0.152590494966	0.00105987352289
Mean	*Variance*
$gab2-0$	$gab2-1$

$\theta = 0$, freq. = .03, $\sigma = 3$:

0.000502239456194	$8.54221417217e-06$
Mean	*Variance*
$gab3-0$	$gab3-1$

$\theta = 45$, freq. $= .05$, $\sigma = 1$:

0.49914672401	0.0106368002851
$Mean$	$Variance$
$gab4 - 0$	$gab4 - 1$

$\theta = 45$, freq. $= .05$, $\sigma = 3$:

0.339621190183	0.00470670196224
$Mean$	$Variance$
$gab5 - 0$	$gab5 - 1$

$\theta = 45$, freq. $= .03$, $\sigma = 1$:

0.152804272225	0.00104881742826
$Mean$	$Variance$
$gab6 - 0$	$gab6 - 1$

$\theta = 45$, freq. $= .03$, $\sigma = 3$:

$8.08921440868e - 05$	$5.8152160312e - 06$
$Mean$	$Variance$
$gab7 - 0$	$gab7 - 1$

$\theta = 90$, freq. $= .05$, $\sigma = 1$:

0.49914672401	0.0106368002851
$Mean$	$Variance$
$gab8 - 0$	$gab8 - 1$

$\theta = 90$, freq. $= .05$, $\sigma = 3$:

0.33670532125	0.00456344304993
$Mean$	$Variance$
$gab9 - 0$	$gab9 - 1$

$\theta = 90$, freq. $= .03$, $\sigma = 1$:

0.152590494966	0.00103242625921
$Mean$	$Variance$
$gab10 - 0$	$gab10 - 1$

$\theta = 90$, freq. $= .03$, $\sigma = 3$:

0.000502239456194	$7.72652861089e - 06$
$Mean$	$Variance$
$gab11 - 0$	$gab11 - 1$

$\theta = 135$, freq. $= .05$, $\sigma = 1$:

0.49914672401	0.0106368002851
$Mean$	$Variance$
$gab12 - 0$	$gab12 - 1$

$\theta = 135$, freq. $= .05$, $\sigma = 3$:

0.339621190183	0.00467721292324
$Mean$	$Variance$
$gab13 - 0$	$gab13 - 1$

$\theta = 135$, freq. $= .03$, $\sigma = 1$:

0.152804272225	0.00103757783919
$Mean$	$Variance$
$gab14 - 0$	$gab14 - 1$

$\theta = 135$, freq. $= .03$, $\sigma = 3$:

$8.08921440869e - 05$	$3.91802350991e - 06$
$Mean$	$Variance$
$gab15 - 0$	$gab15 - 1$

Bibliography

[1] *Open access to scholarly content*, Metropolitan Museum of Art. `http://www.metmuseum.org/research/image-resources/frequently-asked-questions`.

[2] Alexander Blessing and Kai Wen, *Using machine learning for identification of art paintings*, Technical report, Stanford University, 2010.

[3] Leo Breiman, *Bagging predictors*, Machine learning **24** (1996), no. 2, 123–140.

[4] ———, *Random forests*, Machine learning **45** (2001), no. 1, 5–32.

[5] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein, *Textural features for image classification*, Systems, Man and Cybernetics, IEEE Transactions on **6** (1973), 610–621.

[6] S Blair Hedges, *Image analysis of renaissance copperplate prints*, Electronic imaging 2008, 2008, pp. 681009–681009.

[7] Krassimira Ivanova, Peter L Stanchev, and Boyan Dimitrov, *Analysis of the distributions of color characteristics in art painting images*, Serdica Journal of Computing **2** (2008), no. 2, 111–136.

[8] Sina Jafarpour, Güngör Polatkan, Eugene Brevdo, Shannon Hughes, Andrei Brasoveanu, and Ingrid Daubechies, *Stylistic analysis of paintings using wavelets and machine learning*, 2009.

[9] C Richard Johnson, Ella Hendriks, Igor J Berezhnoy, Eugene Brevdo, Shannon M Hughes, Ingrid Daubechies, Jia Li, Eric Postma, and James Z Wang, *Image processing for artist identification*, Signal Processing Magazine, IEEE **25** (2008), no. 4, 37–48.

[10] Mamta Juneja and Parvinder Singh Sandhu, *Performance evaluation of edge detection techniques for images in spatial domain*, methodology **1** (2009), no. 5, 614–621.

[11] GT Shrivakshan and C Chandrasekar, *A comparison of various edge detection techniques used in image processing*, IJCSI International Journal of Computer Science Issues **9** (2012), no. 5, 269–276.

[12] Arvind Sowmyan and Jeffrey Geevarghese, *Painting classification based on art styles.*

[13] David G Stork, *Computer vision and computer graphics analysis of paintings and drawings: An introduction to the literature*, Computer analysis of images and patterns, 2009, pp. 9–24.

[14] H Jaap van den Herik and Eric O Postma, *Discovering the visual signature of painters*, Future directions for intelligent systems and information sciences, 2000, pp. 129–147.