Fractals and Art

By: Andrew Fernandes

October 7th, 2021

An Interactive Qualifying Project submitted for
partial fulfillment of the requirements for the
Degree of Bachelor of Science

Submitted to Worcester Polytechnic Institute

Advisor Professor Mayer Humi

ABSTRACT

Fractals, due to their intricate nature, have served as foundations for art, natural phenomena, and realistic digital images. The goal of this project was to treat the details of great works of art as fractals and analyze them to find patterns corresponding to concepts such as beauty. For this, we found fractal dimensions for binary versions of some works of art in pursuit of such patterns. However, we did not sample enough images to be able to support a conclusion.

CONTENTS

1 EXECUTIVE SUMMARY

The objective of this project was to analyze paintings and similar mediums of art with methods used to measure and describe fractals. Aspects of the art could then be compared to this data to see if any patterns worth further research arose. The main purpose of this research was to find patterns relating to beauty, but details such as style, artist, culture, etc. were relatively simple to consider in addition to it.

The first sections of the paper describe important properties of fractals to contextualize them. Well known fractals are emphasized as examples to help provide an introductory understanding of fractals and convey their visual appeal. The explanations primarily focus on the dimension of fractals, specifically in their relationship to space and scaling. This concept is basis of a technique for calculating fractal dimension, as well as the box-counting method, a general means of estimating fractal dimension used for the project. Both processes are described thoroughly to explain how the code used for the project works on a conceptual level.

The latter half of the project is about the programs that applied the box-counting method, as well as the images used with it and their results. The FracDim function initially used returns fractal dimensions for input binary images, paired with some lines of code that ran it on non-binary images indirectly. From here, the paper covers the iterative improvement of the auxiliary code used over the course of the project, including basic tests used to aid our comprehension of FracDim. Aside from updates and error-fixing, the most prominent addition included is code that measures the dimension for individual RGB colors in the whole image. The results of the function on a sample of images of art is provided, though the nature of the sample prevented us from using it to meaningfully draw conclusions about the images. The project finishes with an explanation of its failings and how it failed to find a reasonable source of data.

2 INTRODUCTION

Of all the projects I found, the fractal IQP appealed to me for two reasons; I could theoretically do it from home in Tiverton, RI (which could be said of plenty of other projects), and it genuinely interested me (which I can't say of the other projects). The section about the use of computers to generate and view fractals strongly relates to my Computer Science major. Similarly, the strong potential of interactive media such as video games to show and make use of fractals, especially relative to other mediums, leaves a place for my experience in my IMGD major. Similarly, the subject would be an interesting, if situational, tool and inspiration for my desired career of designing and making video games, and I would distinctly like to grasp this potential. The exploration and use of fractals can lead to further use as a tool in these fields, comparable to popularizing a new artistic medium (ex. watercolors, paint, acrylics, etc.) or new kind of paintbrush, though I don't expect to have much of an impact in this regard. While the artistic appeal of fractals is by no means a need, the surprising trend of them being considered beautiful is the foundation of the IQP, so better understanding their connection to the human desire for beauty makes it a valid IQP subject from my perspective. Lastly, while I don't have a strong preference for any specific medium to spread my results in (mostly due to lacking experience and knowledge of said mediums), a simple software or game could be useful for showcasing and describing specific fractals within it.

3 FRACTALS AND THEIR DIMENSION

"Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lightning travel in a straight line."

-Benoit Mandelbrot

In many ways, fractals are difficult to completely describe. These patterns contain infinite amounts of detail without being infinitely large, so no matter how closely they are examined, there is always more to find within them. This intricacy is part of their appeal; people have designed, adjusted, explored, and photographed fractals just to capture their patterns as a unique form of art. Sometimes they are found through calculations and computing, but they are far more common to see in nature itself. Crystals, leaves, shells, and flowers are but a few of the ways that the natural world has manifested fractal patterns, and it's not hard to find more if one looks closely.

Figure 1: A lightning strike, one of many natural phenomena with fractal properties

Humanity's first research into processing these details happened on a country-wide scale with the measurement of coastlines. In the mid-20th century, their lengths were estimated by straightening their curves into uniform lengths of line. These lines were drawn end-to-end, with each vertex placed further on the original coastline, keeping the approximation near the original shape. The length of a simplified coastline was the length of a single segment times the number

of segments; an easy calculation compared to the geometry even smooth curve would require. It wasn't an exact method, but the coastlines themselves were too large to directly measure and too detailed to completely record in maps or other forms. Different countries approximated their borders with different lengths of line, and thus different degrees of accuracy. Specifically, smaller lines made more accurate representations of coasts, which led to more accurate measurements. However, as smaller and smaller lines were used to represent them, they became longer and longer without limit. All coastlines, if measured with a high degree of accuracy, would be immensely long, if not infinite in length. This came to be known as the Coastline Paradox, and it led to a curious conclusion; the true measure of such a complex line, if it even existed, was useless compared to a guess.

3.1 The Research of Beauty

Many things have been mathematically proven about fractals, but the idea that fractals are beautiful can never be one of them. It's not necessarily false, but an abstract property without an objective definition has nothing solid to prove about it. More difficult, however, is how the claim implicitly refers to humanity itself, that fractals are beautiful to humans. This means that the nature of beauty also occupies the fields of biology, psychology, and, arguably, anthropology; the sciences of living organisms, the workings of the brain, and the collective nature of society and culture itself. Research would be difficult to perform on such a large, interdisciplinary scale.

One way to manage this is to take an indirect approach. Instead of asking what beauty is, we can ask what is beautiful, or in other words, what has beauty? This is much easier to grasp, but it is still rather subjective, so it's usually measured with the percent of people in a group that think it's beautiful. Once we find examples of beauty and of its absence, we can find qualities shared by one of these sets and see how much of a connection those qualities might have to being beautiful.

3.2 Fractal Properties and Generation

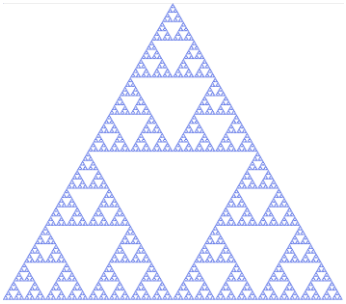


Figure 2: A Sierpiński Triangle

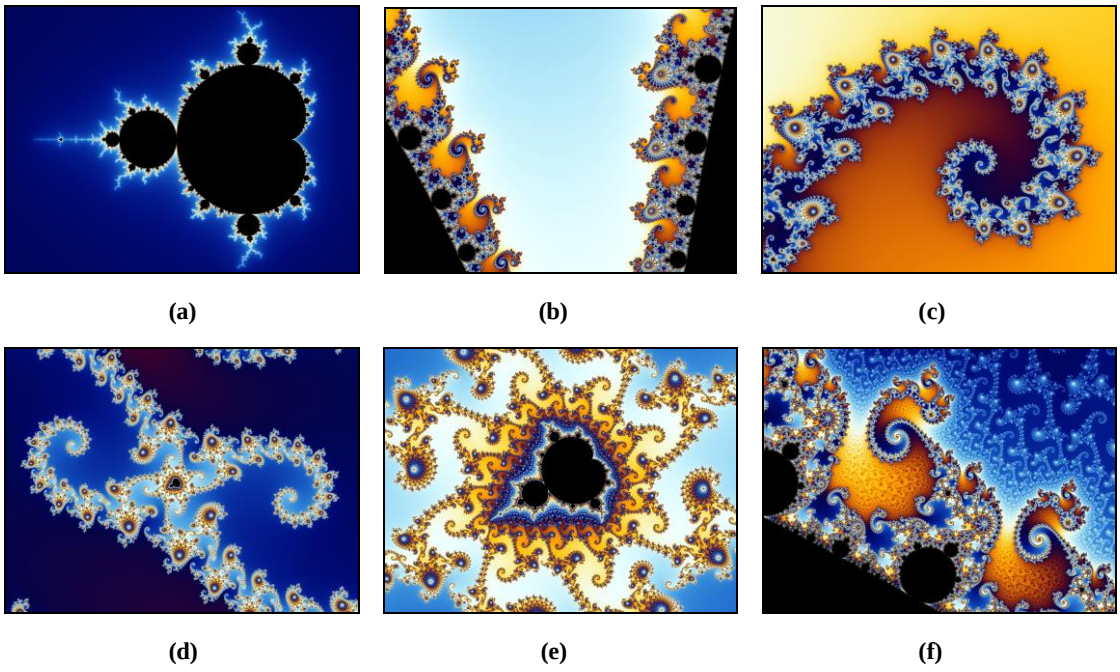
Besides the boundless amount of detail inherent in all fractals, there are a few central properties they generally share with one another. The most commonly known of them is that they are recursive. Specifically, their definition or nature is dependent on itself. Take, for example, the Sierpiński Triangle, a relatively simple fractal that can be made from an equilateral triangle with an infinitely-long 3-step process. To start, you take the midpoints of each of its three sides and draw lines between them, separating the triangle into four smaller equilateral triangles. Next, remove the middle triangle formed by all three midpoints, leaving the other three triangles connected by their vertices. The last step is turning those three remaining triangles into Sierpiński Triangles. In order to finish making a Sierpiński Triangle, you must produce 3 smaller versions of it, which, in turn, require 9 versions of it that are even smaller, and so on. Many fractals can be produced similarly, with a set of instructions that repeat themselves on sections of themselves. Any such fractal is known as an Iterated Function System (IFS) fractal. Other fractals still rely on recursion, though they aren't all made from themselves.

The most well-known of these, and of fractals in general, is the Mandelbrot Set. It is dependent on complex numbers, which, in turn, are dependent on imaginary numbers. To put the subject simply, the value i is defined such that $i^2 = -1$, something normally impossible, and it can be multiplied by any number not involving i to produce an imaginary number. A complex number is a real (non-imaginary) number, an imaginary number, or the sum of a real and imaginary number. A complex number is part of the Mandelbrot Set if the process of squaring it and adding its original value to itself repeatedly leaves it bound within a certain distance of the origin. Specifically, the equation that complex numbers iterate over is

$$Z_k = Z_{k-1}^2 + C$$

Where $Z_0 = 0$. In other words, the values produced by the process approach infinity if the constant is not part of the set.

While a set of numbers isn't as easy to visualize as a triangle with specific sections missing, it's relatively simple for a computer to render. The real and imaginary components of a complex number can be treated as coordinates on a 2D plane, defining what is known as the complex plane. With this, not only can the Mandelbrot Set be graphed, but the complex numbers not in the set can be separated and graphed differently in plenty of ways. This is usually done along the lines of how many iterations they take to get a pre-specified distance away from the origin, at which point they are guaranteed to approach infinity. Once the specifics of the rendering process are chosen, computers can produce shapes and patterns that defy expectations.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 3: Several renders of the Mandlebrot Set. Each image is a section contained in the previous image at a lower resolution

note: Created by Wolfgang Beyer with the program Ultra Fractal 3., CC BY-SA 3.0 <http://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia Commons

At first glance, the edges of the Mandelbrot Set seem a little repetitive at best. There are a lot of circular sections around the main body of the Mandelbrot Set, with smaller circles at their apexes. There are also spiraling patterns that branch away from the set, roads that one can zoom into in pursuit of their end. Yet, there are countless places where we can find a familiar shape, one shockingly similar to the original, "whole" Mandelbrot Set. It might be wrapped in spirals, circled with bands of color, or approached by crystalline branches in every direction, but it is

unmistakable. This is an example of self-similarity, another of the common features of fractals. While not all fractals exist within themselves like this, there is always some variation of them, an extension of their fundamental workings. Delving yet further, we can find more Mandelbrot Sets, but this undersells the other sights that can be discovered. Patterns appear again and again, never quite the same, meshing with their surroundings in new ways. This behavior is an appeal that IFS fractals cannot replicate, a chaotic collage that entails an extensive fractal nature.

3.3 Definition of Fractal Dimension

The last fractal property to mention is deceptively simple; their dimension is not an integer. Straight lines exist in one dimension, squares exist in two, and cubes in three, but fractals can lie between them. For shapes like the Sierpiński Triangle, it's hard to believe, but a central property of dimensionality makes this apparent. When the size of an N-dimensional object is changed by a factor of F, the amount of N-dimensional space occupied by it changes similarly by a factor of F^N. More formally,

$$S = F^D$$

Where S is the amount of space occupied. As an example, consider a pair of cubes with edges 1 and 2 units long, respectively. The smaller cube's sides would each have an area of a single square unit, while its volume would be a cubic unit. The larger cube, twice as large as the unit cube, would have sides covering 4 square units each, or 2^2u^2, while the volume would be $8u^3 = 2^3u^3$. For a cube with edges 3 units long, the areas and volume are 9 square units and 27 cubic units, respectively. With a fractal dimension, not only could space scale at any rate between these, but it wouldn't be measured in squares, cubes, or such of any unit.

The Sierpiński Triangle serves as a good demonstration of what having a fractal dimension means. As mentioned earlier, it is made of 3 smaller copies of themselves, specifically half the original size. This is not hard to prove, considering their outside edges form an equilateral triangle (a triangle with equal-length sides) and that, with one vertex on the whole triangle's vertex and another at the midpoint of their side, one of their sides is half the length of the whole triangle. The result of scaling a Sierpiński Triangle up by a factor of 2 can be replicated by arranging 3 copies of it, which take up 3 times as much space as the original. 1-

dimensional objects would take up twice as much space, and 2-dimensional objects would take up quadruple. Thus, the dimension must be more than 1, but less than 2. In order to find the exact dimension, in this and other situations, the relationship between space and size can be modified by taking the log F of each side to produce an equation for dimension.

$$log_F (S) = log_F (F^D) = D$$

For this fractal, it would be;

$$D = log_2 (3) \approx 1.585$$

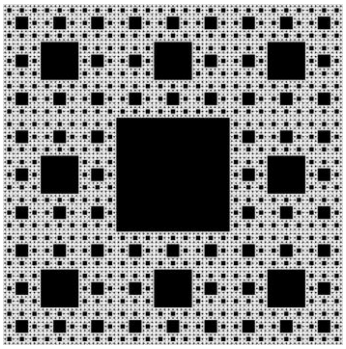


Figure 4: A Sierpinski Carpet

Another of Sierpiński's fractals, known as the Sierpiński Carpet, works similarly to his triangle, and can be measured in the same way. It is based off of a square divided into a 3 by 3 grid of identical, smaller squares. The middle square is removed, and the remaining 8 sections become more Sierpiński Carpets. The 9 equal 2-dimensional squares would have to have equal area, so, working backwards through the equation, they would each be one third of the original size. Scaling size by a factor of 3 changes space occupied by a factor of 8, which means its dimension is;

$$log_3 (8) \approx 1.893$$

This behavior can still be proven in fractals with vastly different dimensions. A Menger Sponge is a cube whose faces are all Sierpiński Carpets. To produce, a cube would be divided into a 3x3x3 array of 27 cubes, 7 of which are removed. These are the center cubes of each of the 6 faces, as well as the one in the absolute center of the whole cube. This leaves 20 sections to become more sponges. Inputting this with the scaling factor, we get;

$$log_3 (20) \approx 2.727$$

In the other direction, there is the Cantor set, the set of points on a given line segment that do not occupy the middle third of it, the middle thirds of the two remaining thirds, and so on. It can be divided into cantor sets of its first and last third, so its space doubles when its size triples, which leads to a dimension of;

$$log_3 (2) \approx 0.631$$

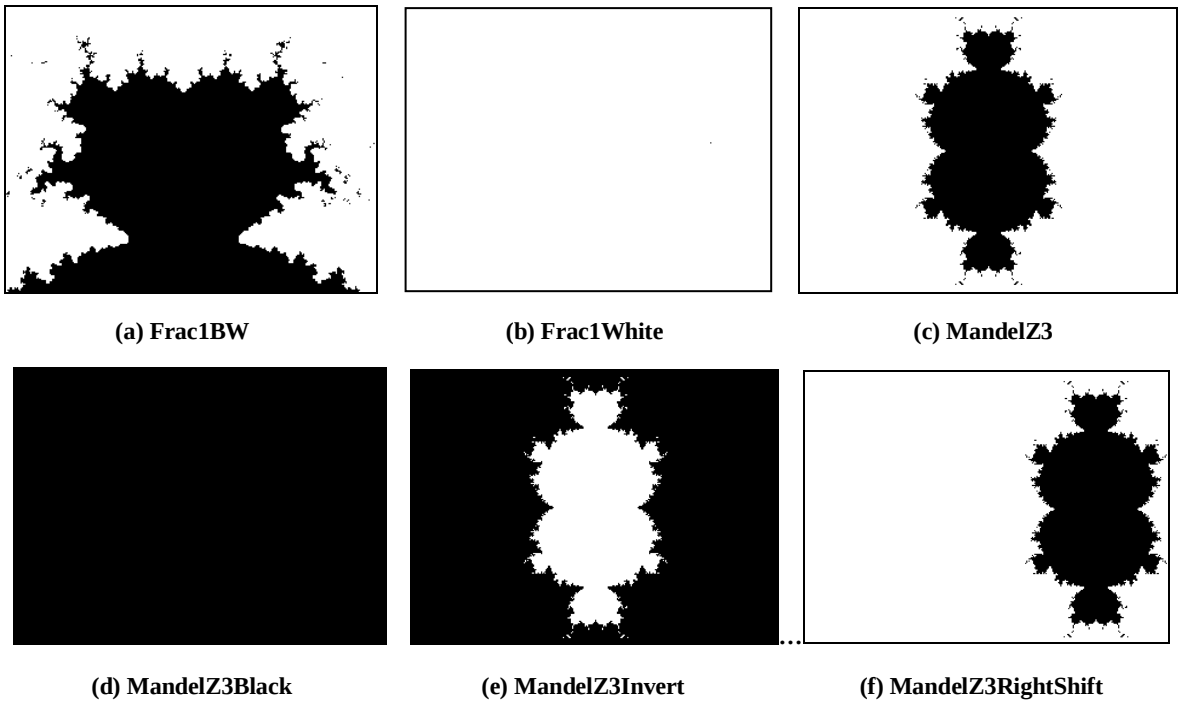
There are many other ways to reach fractal dimensions, though most of them are beyond the scope of this method.

3.4 Estimation of Fractal Dimension

Determining the dimension of a fractal object is particularly difficult when it isn't entirely recursive. Coastlines, as mentioned in the introduction, weren't created by an understood, replicable method. The chaotic nature of their environment was what shaped them, without a clear identifiable beginning or perfect information of their state at any point in time to base research upon. The Sierpiński Triangle can be conveyed entirely with a few statements, but the countless factors that influenced the shape of the coast are impossible to know sufficiently. As such, when researching the fractal nature of similarly complicated objects, calculations can only be done on approximate, finite representations of them. By approximating, it becomes possible that the original object that was approximated was one of a range of objects with varying details, one of which being their fractal dimension, so determining the exact dimension this way is impossible. Thus, the dimension can only be estimated.

There are a variety of methods for estimating fractal dimension for different contexts, though the research for this project only used and relied on the box-counting method. This method can be used on any object that occupies N dimensional space, where N is an integer. For the sake of explanation, this only matters for 2D space, which all the images of the project existed in. First, the representation of the object has every position on it marked as either containing the object, or not containing it. Then, several grids of squares with varying lengths are applied over the representation. For each grid, the total amount of squares that contain some amount of the object is counted. Relatively, each grid is considered to be the same size, with the object being what changes sizes. For example, grids of unit length 2 are instead considered to be measuring the space occupied by a half-sized version of the object. These measurements serve as estimates for how much space the actual object would occupy. The relationship between size scaling and area scaling in this data is returned as the fractal dimension.

4 RESEARCH



(a) Frac1BW **(b) Frac1White** **(c) MandelZ3**

(d) MandelZ3Black **(e) MandelZ3Invert** **(f) MandelZ3RightShift**

Figure 5: Basic fractal images used to understand and practice with FracDim

Binary Image	Fractal Dimension
Frac1BW	0
Frac1White	1.6676
MandelZ3	1.5737
MandelZ3Black	1.5282
MandelZ3Invert	1.7863
MandelZ3RightShift	1.8294

Table 1: Box-Counting Dimensions of Simple Fractal Images

When the subject of fractal dimensions was brought up in a meeting, I was eager to pay attention, as I had heard of the subject while skimming through the subject of fractals in the past. It was after this discussion that I learned about the relationship between the scaling of space and dimension (see section 2.3). Following that, the instructor provided me code for FracDim, a MATLAB function that finds fractal dimensions for images. The images did need to be binary, but I was also pointed to the im2bw function, which produced binary versions of images to use with FracDim. There are a few other limitations to FracDim, however. The function only takes 2D binary images as inputs, so objects that don't exist between 1 and 2 dimensions cannot be adequately represented within the function. This limit applies well to most visual art outside of sculptures and exotic mediums. The other limit is the ability to only estimate the fractal dimension, as it uses the **box-counting algorithm** explained in section 2.4. To summarize, grids of different sizes are used to simulate an object being scaled, and the dimension of the object is the ratio at which the amount of squares occupied changes relative to the scale. Specifically, the binary input is the image in terms of object and non-object positions, so only grid squares with "black" parts of the image in them are considered occupied.

I didn't completely understand the workings of the function at the time, nor at the time of writing, so I experimented with it a bit, producing and processing some fractal images as shown in Figure 5 and Table 1. A white image depicts no objects, so it has nothing to find a dimension for. Also worth noting is how MandelZ3RightShift has a slightly lower dimension than MandelZ3, when the only difference was the location of the object within the image. Though the object itself didn't fundamentally change, the grids used to measure its space seem to be applied differently depending on its location. The last meaningful aspect of the data was a potential bias toward dense images; MandelZ3Black, which was entirely black, had the highest dimension, and MandelZ3Invert, a color- inverted version of MandelZ3, had a significantly larger dimension despite depicting the same fractal edges. This was not a concern, as it was based off a few data points from a small, casually produced pool of data, and estimating algorithms do not need to free of bias to be sound for research, so it was not investigated further. I also spent time researching the functions I used alongside FracDim, in hopes of better understanding MATLAB since I had very little experience with it. Quickly, two flaws in the overall algorithm turned up. Firstly, the documentation for im2bw describes it as "not recommended", and suggests using the imbinarize function instead, likely due to the former being outdated and replaced by the latter,

though still included to not break code that still used it. Second, both the former and the latter function are supposed to take greyscale images as inputs, and treat any inputted image as if it is greyscale, leading to incorrect binarization. The documentation here pointed me to rgb2gray, which makes a greyscale from a color image. After those adjustments were made, I found a way to add an additional, useful feature to the overall code: being able to immediately look at produced images in MATLAB itself, rather than finding and opening them from their folder. This is done through the imshow family of functions, which could even show several images side by side. I had spent a bit of time earlier trying to figure this out, but examples of code in MATLAB documentation that use images usually use such a function to make the functions they are trying to explain easier to understand.

(a) Bliss **(b) The Mona Lisa** **(c) The Last Supper**

Figure 6: Initial art used to test FracDim

After another meeting, I was directed toward trying to find a way to find the fractal dimensions for the separate colors in an image. It took until another meeting until I was able to find a function to split an image this way in the first place, which, while pretty much the only thing I needed to find, was rather elusive. The imsplit function can be used similarly to rgb2gray, making a greyscale image based on each color channel instead of a single greyscale image based on the luminosity of each pixel. To keep the code clean, I duplicated the M file with the previous code and altered it to use imsplit, rather than making the prior code make 4 different binaries

alongside each other. To simplify the actual use of the code, I converted what I had into functions, rather than a few lines commented with what to change for what purpose. The original code became the quickly-named colorBinarize function, while the channel-splitting code's function was called colorSplitBinarize. In the transition, the imshow functionality was lost for a while, which I later realized was due to the function terminating while the image was trying to display, and remedied with a brief wait. Other than the input for the name of the image file and the choice of color (for colorSplit), displaying the final binary is now optional, and the specific threshold value used by imbinarize can be manually selected, rather than automatically determined. To clarify the later, imbinarize puts all grey values below the threshold value to 0 (white), and all values above it to 1 (black). The addition of the threshold feature, while not immediately needed or desired, was motivated by my dissatisfaction with how well the binary images were able to capture the shapes of the original pictures. The expectations weren't particularly high in the first place, considering how the binarization process is supposed to leave only the details relevant to the box-counting method. However, the purpose of the test in the first place is to find the dimension of the art itself, so it is important that the process results in another form of the art, rather than a collage of details derived from that art. Research was wrapped up without time to address this or other concerns.

Image	**Greyscale**	**Red Channel**	**Green Channel**	**Blue Channel**
Bliss.png	1.6449	1.6016	1.6749	1.6893
LastSupper.png	1.6694	1.6697	1.6666	1.6626
MonaLisa.png	1.6470	1.6479	1.6421	1.5791
Starry_Night.png	1.7851	1.7474	1.7840	1.8078
Poet.jpg	1.8024	1.8048	1.8039	1.8037
Abdustion.jpg	1.7233	1.7284	1.7231	1.6825
Artist_In_His_Studio.jpg	1.6317	1.6321	1.6322	1.6407
Nightwatch.jpg	1.6247	1.6605	1.6135	1.5065
Rembrandt_Portrait.jpg	1.2268	1.2239	1.2244	1.2253

Table 2: Box-Counting Dimensions for Multiple Binaries of Digital Images of Art

(a) Starry Night **(b) Poet on a Mountaintop**

(c) The Night Watch **(d) Portrait with Beret and Turned-Up Collar** **(e) The Artist In His Studio**

Figure 7: Additional Paintings Selected for Analysis

The art analyzed by the function, as well as the results, are contained in Figures 6 and 7, as well as Table 2. Other images were selected to try and determine a baseline fractal dimension for images in general, though the attempt was rather flawed, and as such not considered trustable. Images were found and downloaded from the internet in multiple ways, resulting in differing file formats and diverse sources. The first 3 images in the table were obtained early in the project and were intended to precede a set of data large enough to provide a sufficient sample size for confident conclusions. However, no person involved in the project knew how to access such a data set, so the idea was left aside, underprioritized, and never realized. The initial images all had

dimensions of around 1.66, though this appears to either be an anomaly or a mistake in the data-gathering process, as none of the other images returned similar dimensions, as well as the other image sets not having similarly shared dimensions. Overall, binaries of specific color channels usually returned similar fractal dimensions to binaries of the whole image, and most of the difference in this regard existed in the blue channel. There is not much else to say of the data, and due to the small size of the data pool, it cannot make strong claims about art or images in general.

5 CONCLUSION

Abstract concepts such as beauty are not aspects of objects that can be measured, defined, or extracted. They are emergent patterns of the rules that define our world, and each interpretation of them is a piece of a greater puzzle. Some scholars, like Aristotle, write of it being “order, symmetry, and definiteness”. Others, more akin to Socrates, cast doubt on such absolute definitions, not out of contempt, but in pursuing a greater understanding. The pursuit of beauty has no destination, only a web of philosophies to use in understanding the world. That is not to say there is no such thing as concrete beauty. The fractals Mandelbrot revealed to the world had clear, mathematical foundations that gave life to stunning patterns. Despite their complexity, they were almost minimalistic in how they held beauty through raw, unfettered detail. It is through this that we believed in art’s potential fractal natures.

In this project, we sought connections between beautiful images and fractal details that lay within them. Using an application of the box-counting method, we estimated fractal dimensions that could explain the details of famous art we obtained digital versions of. However, we could not find evidence to support or even suggest significant relationships in our data. Images obtained for research purposes were obtained a few at a time, as potential subfields of research were suggested: The initial few paintings were from classical western culture, so the art valued by eastern cultures could also be considered and even compared; limiting the scope of a section to a single artist might have interesting results; perhaps female artists have expressed differences through the paths of their brushes; by focusing on enough eras of a culture, there just might be a description for its change over time. Only the first two of these suggestions were seriously considered, but there wasn’t a way to evaluate enough data for any type of conclusion, not just ones focused on specific groups of art. Art had only been gathered in small amounts since we did not yet know how to sample them with low or minimal bias. The process was inquired to, but never figured out and, sadly, not prioritized in time. As research depends on having enough unbiased samples of data to confidently represent a greater population, we were only able to prepare for the research of data, not the collection of it. In this regard, despite not focusing on the use of FracDim, the code written around it should prove useful for future attempts at this research, as its programmer aided us in the same way.

The only wisdom we found on the nature of beauty is that a bouquet of flowers cannot stand like a meadow; the art that research is focused upon needs to be varied and plentiful in order to support conclusions, and we failed in this regard.

6 APPENDIX-

MATLAB PROGRAMS

6.1 Fracdim.m

```
% FRACDIM Returns the fractal dimension of the input binary image.
% FD = FracDim(I) calculate the fractal dimension of I by using the box
% counting method and assigns it to FD.
%
% Author: Weizhe Shen wshen@wpi.edu
% June, 09, 2016

        largerLength = max(size(I));
        power = ceil(log2(largerLength));
        lengthNum = 2^power;

        % get the amount of padding to add
        pad_afterRow = lengthNum - size(I,1);
        pad_afterCol = lengthNum - size(I,2);

        % pad I with 0's after its last row and column
        I = padarray(I, [pad_afterRow, pad_afterCol], 'post');

        boxCount_store = zeros(1, power);
        scale_store = zeros(1, power);

        boxNum = 1;
        index = 0;

        % use the for loop to shrink the box size
        for i = 1:power
                boxCount = 0;
                for box_row = 1:2^(i-1)
                        for box_col = 1:2^(i-1)
                                % the four terms below are the index range of the current
```

```
                % box we are checking
                minRow = (lengthNum/boxNum)*(box_row - 1) + 1;
                minCol = (lengthNum/boxNum)*(box_col - 1) + 1;
                maxRow = (lengthNum/boxNum) * box_row;
                maxCol = (lengthNum/boxNum) * box_col;

                contain = 0;
                for row = minRow:maxRow
                        for col = minCol:maxCol
                                if I(row,col)
                                        % if ture, then the current box contains the
object
                                        boxCount = boxCount + 1;
                                        contain = 1;
                                        break; % break from the "col" loop
                                end
                        end

                        if contain
                                break; % break from the "row" loop
                        end

                end
        end
end

index = index + 1;
scale = 1/(lengthNum/boxNum);

boxCount_store(index) = boxCount;
scale_store(index) = scale;

boxNum = boxNum * 2; % double the number of boxes per dimension
```

```
        end

        % fit a line for the log-log plot in the least square sense
        FD = polyfit(log(scale_store), log(boxCount_store), 1);
        % return the slope
        FD = FD(1);
end
```

6.2 ColorBinarizeProto.m

```
% Turns color image into black + white image based on total luminosity

I = imread(`MonaLisa.png');% Change name to desired color image.

Must be in current folder
Ia = rgb2gray(I);

%Ic = ~im2bw(Ia); % "Old" function for MATLAB

Ib = imbinarize(Ia); % "Current" function for MATLAB, reccomended

% Note that there are many options available for this function,

imshowpair(Ia,Ib,`montage') % Displays matrixes

FracDim(Ib); %returns dimension
```

6.3 colorSplitBinarizeProto.m

```
% Turns color image into black + white image based on specific color

I = imread(`MonaLisa.png');% Change name to desired color image.
Must be in curr ent folder
```

```
[r,g,b] = imsplit(I);

IR = imbinarize(r);

IG = imbinarize(g);

IB = imbinarize(b);

I2 = IR; % Replace IR with whichever you want to look at.

imshow(I2);

FracDim(I2)
```

6.4 colorBinarize.m

```
%  Given the name of a color image in the current folder, colorBinarize
% produces a greyscale, then binary version of the image, then calculates
% and returns the fractal dimension of the binary.
%   2 options exist:
%       -show (WIP) causes the binary to be displayed.
%       -thresh allows one to input a threshold value of their choice

function dim = colorBinarize(name, option, T)
        if ~exist('option')
                option = "default";
        end
        img = imread(name);
        if ~exist('T')% T should exist if Thresh is option, but this prevents it from
        % crashing for now
                T = graythresh(img);
        elseif T < 0 | T > 1
                T = graythresh(img);
        end
```

```
        imgGrey = rgb2gray(img);
        if option == "thresh" | option == "show&thresh"
                imgBin = imbinarize(imgGrey,T);
        else
                imgBin = imbinarize(imgGrey);
        end
        dim = FracDim(imgBin);
        if option == "show" | option == "show&thresh"
                imshow(imgBin); %fixed
                pause(0.05);
        end
end
```

6.5 colorSplitBinarize.m

```
% produces 3 greyscale versions of an image (one for each color channel),
% then calucates and returns the fractal dimension
% the fractal dimension of the binary.
% 2 options exist:
% -show(R/G/B) (WIP) causes that binary to be displayed.
% -thresh allows one to input threshold values of their choice for
% each color channel

function[dimR,dimG,dimB] = colorSplitBinarize(name, option, TR, TG, TB)
        if ~exist('option')
                option = "default";
        end
        img = imread(name);
        [r,g,b] = imsplit(img);
        if option == "thresh"
                if ~exist('TR')
                        TR = graythresh(r);
                end
                if ~exist('TG')
```

```
                TG = graythresh(g);
            end
            if ~exist('TB')
                TB = graythresh(b);
            end
            imgR = imbinarize(r, TR);
            imgG = imbinarize(g, TG);
            imgB = imbinarize(b, TB);
        else
            imgR = imbinarize(r);
            imgG = imbinarize(g);
            imgB = imbinarize(b);
        end
        dimR = FracDim(imgR);
        dimG = FracDim(imgG);
        dimB = FracDim(imgB);

        if option == "showR"
            imshow(imgR);
            pause(0.05);
        elseif option == "showG"
            imshow(imgG);
            pause(0.05);
        elseif option == "showB"
            imshow(imgB);
            pause(0.05);
        end
end
%Note: to get all outputs, run function with [a,b,c] = function(input)
```

REFERENCES

1. H. Trochet, A History of Fractal Geometry, MacTutor History of Mathematics, 2009.
2. C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (1948) 379–423. 623–656.
3. A. Rényi, Probability Theory, Elsevier, 1970.
4. A. Garrido, Classifying entropy measures, Symmetry 3 (2011) 487–502. http://dx.doi.org/10.3390/sym3030487.
5. M. Higashi, G.J. Klir, Measures of uncertainty and information based on possibility distributions, Int. J. General Syst. 9 (1982) 43–58.
6. T.S. Han, K. Kobayashi, Mathematics of Information and Coding, American Mathematical Society, ISBN: 978-0-8218-4256-0, 2002, pp. 19–20.
7. P. Grassberger, I. Procaccia, Characterization of strange attractors, Physical Review Letters 50 (5) (1983) 346–349. http://dx.doi.org/10.1103/PhysRevLett.50.346.
8. P. Grassberger, I. Procaccia, Estimation of the Kolmogorov entropy from a chaotic signal, Phys. Rev. A 28 (4) (1983) 2591–2593. http://dx.doi.org/10.1103/PhysRevA.28.2591.
9. http://www.fractalarts.com/
10. https://sourceforge.net/projects/detool/
11. http://www.psy.cmu.edu/~davia/mbc/10start.html
12. Jürgens, H., Peitgen, H.-O., & Saupe, D. (2004). *Chaos and Fractals: New frontiers of science* (2nd ed.). Springer-Verlag.
13. Mandelbrot, B. B. (1967). How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Science*, *156*(3775), 636-638. https://doi.org/10.1126/science.156.3775.636
14. By Beojan Stanislaus, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=8862246
15. By Johannes Rössel - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid
16. Sinai, Y. (2011, October 21). *Kolmogorov-Sinai entropy*. Scholarpedia. Retrieved October 5, 2021, from http://www.scholarpedia.org/article/Kolmogorov-Sinai_entropy.