Creating Rough Pencil Sketches from Photographs
Abstract:

Most available pencil sketch image filters fail to emulate the defining properties of a real pencil sketch; the goal of this project was to rectify this by creating an image modification process which took into account the defining characteristics of a pencil sketch and produced a result that faithfully recreated them. While the achieved result was able to satisfy the set requirements, the algorithm is not without faults, and this report will explore both the strengths and weaknesses of this algorithm. This report also covers the motivation for this project and contains a detailed explanation of the process, as well as a description of problems that were encountered in the creation of the algorithm and how they were solved.
Acknowledgments:

I would like to thank the friends, family, and colleagues who provided feedback and insight to aid in the completion of this project.

I would also like to thank Dr. David Mould for acting as my supervisor for this project and providing me with helpful feedback and criticism.
# Table of Contents:

Abstract ................................................................................................................................. 2  
Acknowledgements ................................................................................................................ 3  
Table of Contents .................................................................................................................. 4  
List of Figures ...................................................................................................................... 5  
Introduction .......................................................................................................................... 7  
Motivation ............................................................................................................................ 8  
  Issues With Current Methods ............................................................................................ 8  
  Objectives ......................................................................................................................... 11  
Methodology ........................................................................................................................ 14  
  Planning Stages ................................................................................................................. 14  
  Creating the Pencil Stroke Line Drawing ............................................................................ 15  
  Creating the Tonal Pencil Shading ...................................................................................... 20  
Results ................................................................................................................................. 26  
  General Results ................................................................................................................ 26  
  Problems and Solutions ..................................................................................................... 30  
  Strengths and Weaknesses ............................................................................................... 33  
Conclusion ............................................................................................................................ 37  
References ............................................................................................................................ 38
List of Figures:

Figure 1: The output of various popular pencil sketch apps – Page 7

Figure 2: The two primary components of a pencil sketch – Page 10

Figure 3: The differences in tone between a pencil sketch and a grayscale photo – Page 12

Figure 4: Image kernels $d_x$ and $d_y$ – Page 14

Figure 5: The source image of an apple and the resulting gradient image $G$ – Page 14

Intensity of Figure 5 multiplied by 5 for visibility purposes

Figure 6: A graphical representation of the 8 stroke matrices – Page 15

Figure 7: Matrices $G_3$ (45°) and $G_7$ (135°) – Page 16

Figure 8: Matrices $C_3$ (45°) and $C_7$ (135°) – Page 16

Intensity of Figure 8 multiplied by 10 for visibility purposes

Figure 9: Matrices $S_3$ (45°) and $S_7$ (135°) – Page 17

Figure 10: $S'$ – Page 17

Figure 11: The completed pencil stroke drawing with close-ups – Page 18

Figure 12: A sample grayscale photo and its distribution histogram – Page 19

Figure 13: The distribution histogram of the ideal pencil sketch – Page 21

Figure 14: The cumulative histogram of sample photo and of ideal pencil sketch – Page 21

Figure 15: Our source image before and after preforming the histogram matching – Page 22

Figure 16: Our pencil texture image and the beta for our apple image – Page 23

Figure 17: The tone map and completed tonal pencil shading for our apple image – Page 25
Figure 18: our completed pencil sketch of the apple – Page 25

Figure 19: General Results – Page 26

Figure 20: General Results – Page 26

Figure 21: General Results – Page 27

Figure 22: General Results – Page 27

Figure 23: General Results – Page 28

Figure 24: General Results – Page 29

Figure 25: General Results – Page 29

Figure 26: General Results – Page 29

Figure 27: $G_3$ when generated from our gradient image vs a canny result – Page 30

Figure 28: Difference in shading between [Lu et al., 2012] and our method – Page 32

Figure 29: The incorrect handling of glare – Page 33

Figure 30: The results of giving daytime tone distribution to a night photo – Page 34

Figure 31: The effects of a large number of edges with no particular focus – Page 35

Figure 32: The subject matter is not drowned out by the background – Page 36

Figure 33: Construction of pencil strokes is robust against noise – Page 36

Figure 34: An image before and after passing it through our process – Page 37
Introduction:

In today’s increasingly digital landscape a large number of personal computers have some form of image editing software installed on them and a large percentage of cellphones have some sort of camera app with various image filters. Within these applications a common filter is the “pencil sketch” filter, which attempts to mimic what the photograph would look like if it were sketched with a pencil. Unfortunately these filters are often not much more than a cheap gimmick and leave a lot to be desired if the user truly wanted an accurate representation of the product they claim to deliver. These applications cannot really be blamed for the lack of complexity in their “pencil sketch” filters, since this filter is not the selling point of the application and said application likely comes bundled with numerous other filters that also required development time. However, the lack of complexity in these filters did raise the question that if a proper effort were put into developing an algorithm that could convert a photograph into a rough pencil sketch that was as true to the real thing as possible, what would it look like and how would it be accomplished?
Motivation:

Issues with Current Methods:

Given that a photograph is much more difficult to manually reproduce graphically than a pencil sketch, due to the wider spectrum of colours and level of detail required, it seems strange that when provided with a photograph no application has been able to reverse engineer a method for accurately building a pencil sketch from that photograph. My reasoning with regard to why this is the case is that these applications are not approaching the task in the correct manner; they are attempting to produce an end result that mimics the real thing, but pay no heed to the process that is used to create the product they are trying to recreate.

Figure 1: The output of various popular pencil sketch apps
It is perhaps useful to demonstrate some common issues found within pre-existing applications that claim to create pencil sketches from photos in order to fully grasp what is meant when I state that these applications are approaching the process incorrectly. Figure 1 shows the results of passing a simple image of an apple into 3 different pencil sketch filters that appeared as the top 3 results for the search “pencil sketch” in the Google Play app store. Within Figure 1; image 1 is the unaltered source image of the apple, image 2 is the result achieved by passing the image into Pencil Sketch [Dumpling Sandwich, 2013], image 3 is the result given by Photo Sketch Maker [Aero Tools, 2017], and image 4 is the result given by Sketch Guru – Photo Editor [Cheetah Mobile, 2017].

Image 2 is little more than a grayscale version of the source image with some modifications made to the tone and contrast of the image; it does not give the impression that it was sketched with a pencil, but instead it looks like a photograph that was modified to have a coloration that mimics that of a pencil sketch. Image 3 is probably the most convincing of the bunch, since it at least initially gives the impression that it was shaded with a pencil, but it has 2 major flaws. Firstly, it appears that the image was created almost exclusively by trying to reproduce the grayscale image by stitching together pencil textures of varying darkness. Secondly, because the image is composed entirely of shading there is no sketch outline; if an artist were to sketch this apple they would first sketch an outline of the edges of their subject before shading it. This would help define the edges of the subject matter and give the artist a frame of reference to work with while shading it in. Image 3 is similar to image 2, except it does feature an outline; unfortunately this outline is no more than inverted canny edge detection. One problem with this outline is that it does not match how an outline in an actual sketch would look; sketch outlines are composed of a bunch of short straight pencil strokes, not one long rounded one. Additionally, the outline is in black, which is a colour that cannot be achieved with a standard graphite pencil, regardless of how hard the artist pushes.
Despite their flaws, the results given by these 3 apps were some of the better ones in the grand scheme of things; other “pencil sketch” filters were observed which were quite literally an inverted edge detection or a grayscale version of the source image blended with a pencil shading texture. To summarize, the common issues found with existing pencil sketch filters are as follows:

- The image is either a tonal pencil shading or a pencil stroke outline, instead of a combination of both as a real pencil sketch would be.
- The tone distribution of the shading in the image does not match that of an actual pencil sketch.
- When the tone distribution of the image does match that of an actual pencil sketch, it does not have a texture that gives the impression that it was shaded with a pencil, making it look more like a modified photograph than a pencil sketch.
- The pencil shading looks unnatural due to how the shading transitions between different intensities.
- The pencil shading texture looks unnatural due to it being poorly combined with the tone map for the shading.
- The pencil stroke edge outline is one continuous line as opposed to a combination of short stroke lines, and therefore looks unnatural as this would be difficult to accomplish by hand.
- The image looks unnatural because it contains colour intensities that cannot be achieved with a pencil.
Objectives:

From these prevalent issues with existing methods which have been outlined in the previous section, we can begin to derive how to avoid making the same mistakes and establish objectives for our process. Firstly and perhaps most importantly, our pencil sketch needs to be composed of both a pencil stroke outline and a tonal shading layer. Figure 2 demonstrates the two primary halves of a pencil sketch; the stroke outline is used to define the general structure of the scene and specify clear edges between objects, while the tonal shading layer is used to emphasize the colour and 3D dimensional shape of objects within the scene, by denoting the location and shape of shadows to represent the effects of lighting.
When sketching the outline of key shapes in a drawing an artist will use short pencil strokes to form this outline rather than one long stroke; this is because a longer stroke is more likely it is to deviate from the shape of what they are sketching, with a bunch of short strokes the artist can adjust the shape on the fly without having to erase and try again. Essentially, using a bunch of short strokes to feel out a shape means the artist does not have to commit to getting it right on their first try, instead they can find the shape they are looking for in all the small strokes they made and then go back and add darker strokes to make that shape stand out [SchaeferArt, 2013]. Drawing the outline of shapes using short pencil strokes is such a fundamental technique in the process of creating a pencil sketch in real life, but none of the observed “pencil sketch” image filters attempted to replicate this technique, which can be attributed as part of the reason they don’t look right.

As for the tonal shading layer, there are multiple areas where existing pencil sketch creation algorithms fall flat, which our method will attempt to improve on. As demonstrated in Figure 3, the distribution of tone in a pencil sketch is much different from that of a grayscale photograph; since shading is a time consuming process for the artist, light coloured surfaces that are in full view of the light source are often portrayed as white. For example; pale skin, pink flowers, or a blue sky would all be white in a pencil sketch, which makes white a much more plentiful colour in pencil sketches than it is in grayscale photographs. On the other hand, dark shading is much less common in pencil sketches, as it is usually reserved for surfaces which are either very dark in colour or areas where a heavy shadow is needed to define the shape of an object. Additionally, transitions between light and dark shading used to define shape and lighting should not be sudden drastic shifts in intensity, but gradual smooth transitions [SchaeferArt, 2014].
Figure 3: The differences in tone between a pencil sketch and a grayscale photo

By attempting to emulate the techniques used by actual artists to create pencil sketches, our process should generate more accurate and realistic looking pencil sketches from a photographic input than any application on the market. A summary for the objectives of this project is as follows:

- Result should consist of both a pencil stroke edge outline and pencil tone shading.
- Pencil stroke outline should be created by combining small pencil strokes as an artist making the sketch would.
- Tonal pencil shading should be true to the tone allocation found in a real pencil sketch.
- Pencil shading should transition smoothly between light and dark when defining illumination.
- Result should have realistic texture and colour with regards to a real pencil sketch.
- The process should be reasonably resistant to elements of variance between images, such as lighting and visual noise.
Methodology:

Planning Stages:

For this project I chose to use the paper *Combining Sketch and Tone for Pencil Drawing Production* [Lu et al., 2012] as a guideline to build the pencil sketch algorithm. Originally published in *NPAR '12 Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, this paper summarizes the methodology for creating pencil sketches from a photographic input. This paper was chosen to serve as a guideline for this project due to its analysis of real life pencil sketches and the process behind creating them as a basis for developing its method, which corresponded with our objectives for this project. Specifically this paper highlights that edges in pencil sketches are not defined by long curved lines, but rather by a series of short pencil strokes. It also specifies that the tone distribution of actual pencil sketches differs significantly from that of a greyscale image, an attempts to discern the specific patterns that determine the tone of an object within a pencil sketch. These two stipulations are two of the core objectives we are attempting to accomplish with this project, so this paper seems like a suitable base of knowledge to work from.

For this project I chose to utilize the open source computer vision and machine learning software library OpenCV. Having used this library in the past for other computer vision projects, OpenCV seemed like a good fit; as it’s functionality for creating and manipulating image matrices would prove useful for the tasks this project would need to accomplish. The language I chose to use for this project was C++; not only was C++ one of the languages that I had the most familiarity with, but in the past I had used C++ in conjunction with OpenCV, making it an obvious choice.
Creating the Pencil Stroke Line Drawing:

The first step of creating the pencil stroke drawing is to obtain a gradient map from the grayscale version of the inputted source image. This gradient image will be determined by the equation \( G = \sqrt{(d_x I)^2 + (d_y I)^2} \), where \( I \) is the grayscale version of the input, and \( d_x \) and \( d_y \) are the image kernels specified in Figure 4. In \( d_x \) and \( d_y \) the value \( n \) is equal to 0.0002; this value was determined through the testing of numerous inputs to provide the best results.

\[
\begin{align*}
\text{dx} & = \begin{bmatrix}
-n & 0 & n \\
-n & 0 & n \\
-n & 0 & n \\
\end{bmatrix} & \text{dy} & = \begin{bmatrix}
n & n & n \\
0 & 0 & 0 \\
-n & -n & -n \\
\end{bmatrix}
\end{align*}
\]

Figure 4: Image kernels \( d_x \) and \( d_y \)

To obtain the gradient image \( G \), we separately convolve the greyscale image by these two kernels, then on a per element basis we do each of the following; square each of the results, add the two results together, and finally square root that sum in order to receive the gradient image, which is seen in Figure 5.

Figure 5: The source image of an apple and the resulting gradient image \( G \)
The next step is to create 8 line segments 22.5 degrees apart which we will use as our pencil strokes. To do this we create 8 matrices with length and width equal to $1/30^{th}$ the size of the source image’s shortest side. We then loop through all y values of the first 3 matrices (0°, 22.5°, and 45°) and using trigonometry we determine what the x value of that stroke should be for the given y value, and set the value at that coordinate equal to 255. We create the 7th line segment (157.5°) by setting the x values we received for the 22.5° line segment in the reverse order. Finally, we create the last 4 line segments (67.5°, 90°, 112.5°, and 135°) by rotating each of the matrices we have so far by 90°.

Figure 6: A graphical representation of the 8 stroke matrices

Once we have our 8 stroke matrices, we create 8 new matrices $G_1$ through $G_8$, which are obtained by convolving the gradient image by the corresponding stroke matrix as a kernel. The result of this process can be seen in Figure 7; areas of the image where the direction of the gradient matches that of the kernel will appear focused with a high magnitude, while areas of the image where the direction of the gradient does not match that of the kernel will appear blurred with a lower magnitude. To the naked eye it should be clear from this step which sections of the image best match a particular stroke direction and are therefore suited to be represented by a pencil stroke in that direction.
Next we create 8 more matrices the size of our image called $C_1$ through $C_8$ and fill them with zeroes. We loop through our image pixel by pixel, and at each pixel we determine by comparison which $G$ matrix has the greatest magnitude at that particular pixel. When we have determined which $G$ matrix has the highest magnitude for that pixel we set the value of that pixel in the corresponding $C$ matrix equal to the value of that pixel in the gradient image. Essentially what is being accomplished by this step is we are determining for each non-zero pixel in our gradient image which stroke direction it is most compatible to be represented by, and then we are storing the value at that pixel in the $C$ matrix corresponding to that stroke direction. Each resulting $C$ matrix will only contain non-zero pixels at locations where that particular pixel would be best represented by the corresponding stroke direction.
The next step is to use the C matrices to create our pencil strokes for each direction, which we will place in matrices named $S_1$ through $S_8$. We accomplish this by convolving each C matrix with its corresponding line segment and placing the result in the corresponding $S$ matrix.

![Figure 9: Matrices $S_3$ (45°) and $S_7$ (135°)](image)

Next we create matrix $S'$ at the same size of our image and fill it with zeroes, this matrix will house all of our pencil strokes combined into one matrix. This is accomplished by individually adding $S_1$ through $S_8$ to $S'$ on a per element basis.

![Figure 10: $S'$](image)
The final step needed to receive our pencil stroke line drawing is to invert the matrix \( S' \) by looping trough each pixel of \( S' \) and setting its value equal to 255 – its value. We add a special condition that if the value of any pixel is less than 76.5 we set that pixel to 76.5; this way no pixel in our pencil stroke image will be darker than 76.5, which we have determined to be the darkest colour achievable by a standard graphite pencil. Much like a real pencil sketch our stroke drawing gives the impression that it was created with numerous small pencil strokes, as multiple intersecting “sketchy lines” can be observed at corners and around curves of edges in the image. This result also abides by the technique that artists like to go over their light guess strokes with darker more definitive strokes once they have found the shape that they like [SchaeferArt, 2013]. Since our method manipulates the gradient image, which is smaller in magnitude farther from edges or in places where edges are less well defined, it gives the impression that pencil strokes are darkest right on the edges and weaker in the area around them.

![Image of pencil stroke drawing]

Figure 11: The completed pencil stroke drawing with close-ups
Creating the Tonal Pencil Texture Shading:

The first step of this part of our process is to modify the tone distribution of our grayscale image so that it matches the tone distribution of a pencil sketch. To achieve this we first need to create a distribution histogram for the grayscale version of our source image; a distribution histogram is a representation of the total number of pixels in the image that have a given intensity from 0 to 255. We accomplish this by creating an array of size 256 where the value at each position in the array is equal to the number of pixels in our source image of that intensity.

![Figure 12: A sample grayscale photo and its distribution histogram](image)

We also need to construct a distribution histogram for a pencil sketch so that we will have a target for modifying the tone distribution of our image. Thankfully the paper we are using outlines an equation that can be utilized to obtain a distribution histogram for the “average” pencil sketch, which was obtained through testing and the analysis of many real life pencil sketches. This equation is composed of three main parts, which each account for one of the three tonal layers in a pencil sketch; the bright white regions that are the colour of the paper where practically no shading is present, the areas with mild gray shading where artists use many strokes with different levels of pressure, and the areas with dark shading strokes that represent dark colours, depth change, and the geometric contours of objects [Lu et al., 2012].
\[ p_1(v) = \begin{cases} 
\frac{1}{\sigma_b} e^{-\frac{255-v}{\sigma_b}} & \text{if } v \leq 255 \\
0 & \text{else}
\end{cases} \]

Equation \( p_1 \) defines the bright white layer, where most pixels in this layer concentrate at its peak of 255. In this equation we use 9 as the value of the scale parameter \( \sigma_b \), as recommended by the paper.

\[ p_2(v) = \begin{cases} 
\frac{1}{u_b - u_a} & \text{if } u_a \leq v \leq u_b \\
0 & \text{else}
\end{cases} \]

Equation \( p_2 \) defines the mild tone shading layer that does not peak at any particular intensity. In this equation we use 105 as the value of \( u_a \) and 225 as the value of \( u_b \), which are the two parameters that control the range of the distribution.

\[ p_3(v) = \frac{1}{\sqrt{2\pi\sigma_d}} e^{-\frac{(v - \mu_d)^2}{2\sigma_d^2}} \]

Equation \( p_3 \) defines the dark shading layer that peaks at its mean value of \( \mu_d \), which we have chosen to be 90. In this equation we use 11 as the value of the scale parameter \( \sigma_d \), as recommended by the paper.

\[ p(v) = \sum_{i=1}^{3} \omega_i p_i(v) \]

From these 3 equations we are able to form equation \( p \), which can be used to create a distribution histogram for the “ideal” pencil sketch. We do this by passing the values 0 through 255 into equation \( p \) as \( v \); using 76, 22, and 2 as the values of \( \omega_1 \), \( \omega_2 \), and \( \omega_3 \) respectively, and storing the results in an array in the same manner as did previously for the other distribution histogram. The resulting histogram can be seen in Figure 13.
Next we need to convert each of these distribution histograms into a cumulative histogram. A cumulative histogram is when the relative frequency of each intensity value (it’s probability) is combined with the probability value of the intensities before it on a scale from 0 to 1 (the value of element 255 will always be equal to 1). This is accomplished by dividing each element of our distribution histograms by the total number of pixels in the image to obtain that intensity’s relative probability on a scale from 0 to 1. Then we loop through each element of our histogram and set the value of the cumulative histogram at that index equal to the probability of the index value + the value of the element before it.

Figure 14: The cumulative histogram of our sample photo and of our ideal pencil sketch
Once we have obtained a cumulative histogram for both our source image and our ideal pencil sketch, we can then adjust the tone distribution of our source image so that it’s histogram matches that of our ideal pencil sketch by preforming histogram matching. If we let $F_a(x_1)$ be the function of the cumulative histogram for our source image and $F_d(x_2)$ be the function of the cumulative histogram for our ideal pencil sketch, we need to determine for each value of $x_1$ from 0 to 255, some value of $x_2$ from 0 to 255 where $F_a(x_1)$ is as close as possible to $F_d(x_2)$ [Gonzalez and Fittes, 1975]. This histogram matching is implemented by looping through each pixel of our grayscale source image, and for that pixel with intensity $i$, we set its intensity to $x$, where $x$ is the value which satisfies:

$$\arg \min_{x} \left| f(x)_{x \in [0,255]} - |F_d(x) - F_a(i)| \right.$$
Once we have our adjusted tone image, we need to give it a texture that makes it appear as though it was shaded with a pencil. For that we will need a pencil shading texture which is the same size as our source image. This is accomplished by repeating our pencil texture image vertically and horizontally until the resulting image equals or exceeds the size of our source image, and then cropping the result so that it is exactly the size of our source image. For this process we utilize alpha blending along the edges of each copy of the pencil texture so that there aren’t any obvious seams or creases in our final texture image.

We will also need a “beta” image, which will be used to apply our pencil texture to our tone map. When our pencil texture image is raised to the power of our beta image on a pixel by pixel basis, it should produce a result that resembles our tone map with the texture of our pencil texture image. This is accomplished via inverting our tone map by simply looping through each pixel of our tone map and setting its value equal to 255 – its value.

![Figure 16: Our pencil texture image and the beta for our apple image](image)

Finally to receive our pencil shading result we raise each pixel of our pencil texture image to the power of the value of our beta image at that pixel.
Figure 17: The tone map and completed tonal pencil shading for our apple image

The final step to create our completed pencil sketch is quite simple; we loop through each pixel of our image and multiply the value of our stroke image at that pixel by the value of our tonal shading image at that pixel.

Figure 18: our completed pencil sketch of the apple
Results:

General Results:

The results generated by our procedure manage to meet the objectives set at the start of the project; however the process is not without its flaws. The following sections will detail some of the issues that were encountered in the creation of the process, as well as the strengths and weaknesses of our process. Below are some general results produced by our program.

Figure 19 and 20
Figure 21 and 22
Figure 24, 25, and 26
Problems and Solutions:

Initially I was using a canny edge detection to produce the gradient image used in the creation of the pencil stroke line drawing; since I was under the impression that this would be a strict improvement to the method specified in the paper. Canny edge detection creates a gradient intensity image in an almost identical manner to the one specified in the paper, but it also utilizes non-maxima suppression and hysteresis thresholding to produce thinner more precise edges which are resilient to false positives caused by noise and colour variation. Unfortunately, the result returned by canny edge detection contains pixels of only two intensities; 255 if the pixel is classified as being part of an edge, and 0 if it is not. This caused issues when I attempted to create the G matrix series as outlined in my methodology, as a lack of varying intensity in the gradient image meant that the response maps generated by this step would not show intensity peaks at any location. This lack of variation in intensity within the G series response maps meant that they could not be used to determine which sections of the image best aligned with a particular stroke direction, since there were no peaks in intensity to use as an indication of strong alignment. Figure 27 demonstrates this issue; Response map $G_3$ (45°) when generated from our gradient image (left) shows obvious peaks in intensity where edges match the direction of a stroke at 45 degrees, where $G_3$ when generated from canny edge detection (right) only contains pixels of intensity 0 or 255. Once this issue was noticed it was corrected by creating our own code for generating the gradient image from scratch.

Figure 27: Response map $G_3$ when generated from our gradient image vs a canny result
Another roadblock that was encountered during the creation of the process was determining how to create the “beta” image. In the paper beta is specified as the result that when used in the equation $T = H^\beta$ will produce the final tonal pencil shading $T$, where $H$ is the pencil texture image. The paper states that $H^\beta$ should be approximately equal to tone map $J$, and provides the following equation to calculate beta:

$$\beta^* = \arg\min_{\beta} \| \ln H - \ln J \|_2^2 + \lambda \| \nabla \beta \|_2^2$$

The paper further states that this equation can be transformed into a standard linear equation and solved via the conjugate gradient method. Unfortunately, this step is rather poorly documented in comparison to the rest of the paper, which is odd since it is seemingly the most mathematically complex step of the entire process. Despite stating that the equation can be transformed into a standard linear equation, neither the transformed equation nor the process for transforming it are provided, which makes an actual implementation very difficult as even the original equation itself is practically unexplained. I attempted to transform this equation into the form $Ax = b$ and solve it using the conjugate gradient method; by implementing the Eigen linear algebra C++ library and using it to create sparse matrices for $\ln H$ and $\ln J$, and solve using it’s conjugate gradient computation tools. Regrettably, I was not able to produce a result that provided a final pencil texture map that looked anything like what was desired. After being stuck on this section of the process for a number of weeks I decided to speak to a colleague who is a graduate math student about transforming this equation; when he was unable to provide any insight into how to solve for beta, I decided that it was time to take a different approach.

While attempting to understand how to solve my problem I realized that the beta in the paper was shown as having intensities between 0 and 2.0, and therefore should look quite similar to an inverted version of our tone map. This is logical since on a colour intensity spectrum of 0 to 1.0, any pixel to the power of 0 (black) will become 1.0 (white), and the higher the value of any given pixel in beta is, the darker the corresponding pixel in the pencil texture image will be displayed.
One fundamental difference between this method and the one outlined in the paper is that in our method the darkness of the pencil texture cannot exceed its default intensity. Say our beta image at a particular pixel has intensity 1.0 (which is the maximum beta value our method can produce), if our pencil texture at that pixel has an intensity of say 0.5, then the intensity of that pixel in the result will be $0.5^1 = 0.5$, which is identical to the intensity that pixel has in the pencil texture image. In the paper a graphical representation of beta shows its values ranging from 0 to 2.0, which is means the value of a pixel in the result can be reasonably darker than its counterpart in the pencil texture image. To use the same example as earlier, a pixel in the pencil texture with a value of 0.5 raised to a beta value of 2.0, will produce a value of 0.25 at that pixel in the result, which is a rather noticeable increase in darkness.

This difference can be seen as strength; as noticeably darker versions of the texture map begin to look unnatural as they are darker in colour than what can feasibly be achieved with a standard pencil. With our method, the darkness of the pencil texture will never exceed its original darkness, so it is impossible for it to look unnatural. The pencil shading does need to look heavy in some locations of the sketch, but we can solve this issue by simply using a darker shading texture for our pencil texture image. By choosing a pencil shading texture with a darker average intensity we can still achieve heavy shading while at the same time guaranteeing that the texture will always look natural. Figure 28 demonstrates the difference in shading intensity between the results produced by the paper [Lu et al., 2012] and our method; in the paper’s method (left) there are areas where the shading has a blackish colour, which looks rather unnatural, while our result (right) looks more feasible to have been done with a pencil.

Figure 28: Difference in shading between [Lu et al., 2012] and our method
Another advantage of our method for finding beta is that it is much faster than the method utilized by the paper. Most of the images used for testing the algorithm had an average of one million pixels, meaning the sparse matrices used in the conjugate gradient solution would be one million by one million pixels, with the solution being approximated over 100 or so iterations. On a one million pixel image, the run time of conjugate gradient solution was more than double the run time of our method, with the gap only increasing for even larger image sizes. At its current speed our process could likely be implemented as a filter on a mobile phone application, while the method utilized by [Lu et al., 2012] would likely be far too slow for the average consumer.

**Strengths and Weaknesses:**

One weakness of our process is how it handles heavy lighting glare on objects that are glossy in texture. Due to the relative simplicity of our process for generating the gradient image, the quick transition between the colour of the surface to white is interpreted as an edge, and thus pencil strokes are generated for it. In a real pencil sketch this glare would only be expressed via shading.

![Figure 29: The incorrect handling of glare](image)
Another weakness of this process is its tone mapping for photographs that were taken at night or in similar low lighting environments. Our distribution histogram demonstrates that over 50% of pixels in an average pencil sketch are white, since the artist only needs to use shading to demonstrate shadows, contours, or dark colours. This histogram was created using multiple pencil sketches as a frame of reference, but a common theme among them was they were all sketched in the daytime or indoors with sufficient lighting; a pencil sketch of a night time scene by contrast would have very few of the pixels be represented by the colour of the page (white). When our process attempts to perform histogram matching on a night time photo, it runs into problems when the dominant background colour which would normally get set to white is darker than the subject matter. This problem generally leaves the entire sketch mostly white with unnatural looking blotches of shading in the darkest regions. The only way to fix this issue would be to generate a second “ideal” histogram for night time sketches, since the tone distribution is so hugely different. A detection system would also need to be constructed to determine which histogram to use, or users would need to be allowed to choose depending on the content of their image. This problem is not a huge concern though, because night time pencil sketches are quite rare, since it is difficult to capture the essence of nighttime lighting with a pencil and most artists would rather sketch in better visibility during the day.

Figure 30: The results of giving daytime tone distribution to a night photo
One final weakness of this process is handling images which have a high level of detail, but no distinct subject matter. Our particular emphasis on making stroke lines appear visibly “sketchy” causes the result to look messy when a clear focus is not present. A large number of edges within a small area cause the intersecting “sketchy lines” to overlap with each other enough that it begins to look like a jumbled mess. It is possible to tone down these pencil strokes by adjusting variables so that they look better in this case, but I chose to leave them as is because I enjoy the distinct “rough sketch” effect they give to other images. Additionally most pencil sketches have a clearly defined subject or capture a landscape from far enough away that these details aren’t an issue. Generally speaking, the types of photos that cause this issue are ones that people are least likely to want to turn into a pencil sketch, since they are not particularly interesting.

Figure 31: The effects of a large number of edges with no particular focus

Outside of the general results which fulfill this project’s objectives, there are a few notable strengths worth highlighting. One of these strengths is the algorithm’s resilience to background noise; much like a real pencil sketch the objects in the background of the image are not drawn with the same complexity or focus as those in the foreground.
Another particular strength is our algorithm’s consistency for determining the correct direction of strokes for creating edges. Because our method does not attempt to determine stroke direction based on a single pixel and the values of those surrounding it, it is rather resilient to noise and consistently draws strokes in the correct direction.
Conclusion:

In conclusion, this project was able to accomplish its objectives by creating a pencil sketch from a photographic input that was authentic to the look and feel of a real life pencil sketch. Through analysis of the procedure used to create actual pencil sketches we were able to create a process that mirrored these techniques to produce a satisfactory result. Although the completed process is not devoid of issues, this process could feasibly be implemented into a cellphone application to be used as a “pencil sketch” image filter and would be superior to those currently on the market.

Figure 34: An image before and after passing it through our process
References:

**Aero Tools** (2017) *Photo Sketch Maker* (Version 1.0)

**Brandon Schaefer** (2013) *Learn to Draw #01 - Sketching Basics + Materials*
https://www.youtube.com/watch?v=ewMksAbgdBI

**Brandon Schaefer** (2014) *Learn to Draw #03 - Shading Techniques*
https://www.youtube.com/watch?v=5Zc1xVS_X7Q

**Cheetah Mobile (Photo Editor)** (2017) *Sketch Guru - Photo Editor* (Version 1.2)

**Dumpling Sandwich** (2013) *Pencil Sketch* (Version 6.2.4)


**Guennebaud, G; Jacob, B** (2017) *Eigen (C++ library)*


**Lu, C; Xu, L; Jia, J** (2012) *Combining Sketch and Tone for Pencil Drawing Production*,
International Symposium on Non-Photorealistic Animation and Rendering, pp. 65-73