[WORKING TITLE] Improved Perception-Driven Semi-Structured Image Vectorization via a Machine-Learning-Based Approach

A Special Problem by

Sean Francis N. Ballais

2015-04562

BS CS

Presented to the Faculty of the

Division of Natural Sciences and Mathematics

In Partial Fulfillment of the Requirements

For the Degree of

Bachelor of Science in Computer Science

University of the Philippines Visayas

TACLOBAN COLLEGE

Tacloban City

Month Year

This special problem, entitled "[**WORKING TITLE**] **IMPROVED PERCEPTION-DRIVEN SEMI-STRUCTURED IMAGE VECTORIZATION VIA A MACHINE-LEARNING-BASED APPROACH**", prepared and submitted by **SEAN FRANCIS N. BALLAIS**, in partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE** is hereby accepted.

<div align="center">

_____

PROF. VICTOR M. ROMERO II
Special Problem Adviser

</div>

Accepted as partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE**.

<div align="center">

_____

DR. EULITO V. CASAS JR.
Chair, DNSM

</div>

Acknowledgements

First of all I would like to thank the Lord for his guidance during the course of my research and for making this thesis possible. I would like to thank my family who served as my inspiration, and never failed to support me all throughout my studies. Thanks to ...

Abstract

Although the abstract is the first thing that appears in the thesis, it is best written last after you have written your conclusion. It should contain spell out your thesis problem and describe your solution clearly.

Make sure your abstract fits in one page !

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In computer graphics, most images are typically stored as a sequence of dots in a rectangular grid (see Fig. 1.1). Each dot is called a pixel, a small part of an image that holds one specific colour, Photographs, also called natural images [12], are one of, if not the most, common images that are stored in this manner. Many digital forms of art or any graphics work, such as paintings, posters, and icons, are also stored the same. Digital images stored in this manner are called *raster images*. These images are stored in various image formats. The most commonly used formats are JPEG, GIF, BMP, TIFF, and PNG. Each have their pros and cons, from quality of the resulting image to the file size. Nevertheless, they all still accomplish the task of holding raster image data. Everything you see in the displays of devices such as laptops and mobile devices is a raster image. Computer displays are collections of pixels, in the common definition of a dot on the screen, which computers map images to to be able display them. This is the reason why **all** *displayed* images are raster images.

A positive aspect of raster images is their simplicity. As mentioned earlier, raster images consists of a grid of pixels (also called a pixel matrix in other literature [22]). This pixel grid can simply be assigned a combination of colour values to create an

Figure 1.1: When zoomed in enough, each individual pixel of a raster image is visible. Meme image obtained from `http://thesismemes.tumblr.com/post/73483120281`.

image. As such, working with raster images can be analogous to painting in the real world [5]. Given the right combinations of colours, we can produce natural images, i.e. photographs [12]. Intuitively, this means that we can store fine details in a raster image [18]. This is in contrast to *vector images*, which use a series of points and mathematical calculations to form lines and shapes. Vector images are unable to display lush colour depth and keep granularity, as found in raster images, as they use solid colours or gradients [5][6]. There are studies that have been conducted in improving and utilizing *gradient meshes*, a vector graphics primitive that allows for intricate colour gradients in regular quadrilateral meshes first introduced by Adobe Illustrator, to produce photorealistic vector images. However, as noted in the paper by Jian, S, Liang, L., Wen, F., and Shum, H., simple gradient meshes are insufficient to keep the fine details of images [7][18]. It is also important to mention that vector graphics, despite represented as mathematical calculations, are still converted to

raster format in a process called *rasterization* for it to be displayed on-screen, since many modern screens are raster displays [19].

1.1 The Problems of Raster Graphics

With all the pros raster graphics have, it does not mean raster graphics are not without their caveats. Raster graphics have their own disadvantages which could affect the image quality and their use.

Image at 64px by 64px.

Image scaled to 350x350 pixels with nearest neighbour interpolation.

Image scaled to 350x350 pixels with bicubic interpolation.
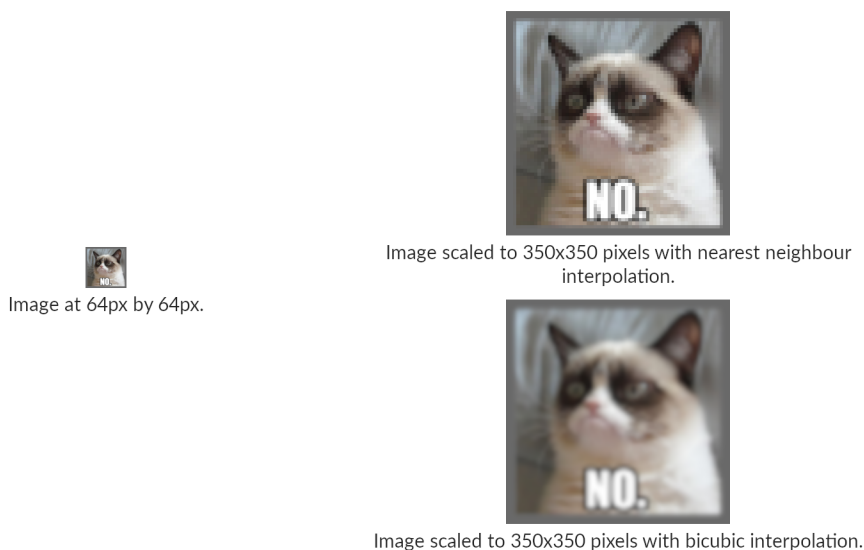
Figure 1.2: Different interpolation algorithms will produce different results. As seen in the image, the quality will also differ, from an image looking blocky to an image looking blurry. Cat meme image obtained from `https://www.hercampus.com/school/uwindsor/school-thoughts-told-grumpy-cat-memes`.

Raster graphics are **resolution-dependent**. This simply means that raster images are in their highest quality in the resolution they are initially created in and attempting to scale it up will gradually degrade the quality of the image as the image size or resolution grows larger. Rasters only have a finite number of pixels. Increasing

the size of an image (also called upsampling [12] or single-image super resolution [24]) would entail moving the individual pixels into different locations depending on the scaling factor, the distance the individual pixels will be moved to horizontally and vertically. Upsampling will create empty pixels in between the shifted pixels when there is nothing done to substitute the empty pixels. This will create an unusable image [10]. We can utilize interpolation to fill these empty pixels with colour. *Classical* image upsampling approaches approximate colour and intensity values of these empty pixels are calculated based on the values of surrounding pixels, typically the shifted pixels. However, the specifics are dependent on the interpolation algorithm used in upscaling the images. Commonly used classical interpolation methods for resizing images include Nearest-Neighbour, Linear Interpolation, and Cubic Interpolation, which most, if not all, are readily available in popular raster image editing applications, such as Adobe Photoshop [2] and GIMP [1]. The results produced by these interpolation algorithms typically suffer from blurring of sharp edges and ringing artifacts due to the fact that the algorithms do not assume anything about the data [14][16]. See Fig. 1.2 for an example of blurring caused by scaling. There are adaptive image scaling techniques that consider image features such as edge information and texture to scale images with better quality than the classical methods. Examples of adaptive techniques are content-aware image resizing, seam curving, and warping-based methods. These techniques have their downsides as they take more computational time than their non-adaptive counterparts and may produce unexpected or even unsatisfactory results [16]. One of the latest advancements in upscaling images involves the use of artifical intelligence, specifically *neural networks*, as found in the works of AI Gigapixel and by Yang, C. Ma, C. and Yang, M.. They produce high quality upscaled images

and is a significant improvement over previous non-AI based upscaling techniques. However, they require high-end expensive hardware to produce results in the shortest amount of time possible. In the case of AI Gigapixel, a laptop with an integrated graphics card takes 20 minutes to produce a final high resolution image. For the work by Yang. C, Ma.C., and Yang. M., they utilized an Nvidia Titan Xp, a high-end GPU that costs $1,200 as of November 18, 2018 [4], to upscale a 520x520px image 2x, 4x, and 8x its size, and took 0.8s, 2.1s, and 4.4s, respectively, to complete [23][20].

1.2   Vectorization To The Rescue

Vectorization is the process of converting raster images into vector images [25]. Vector graphics uses collections of geometric primitives, such as points, curves, and points, and mathematical calculations to form an image [5]. Unlike raster graphics which uses a large pixel matrix (which will require large spaces without using proper image compression), vector graphics are able to smoothly scale to different resolutions, large or small, without any degradation in image quality [22][7]. This makes them **resolution-independent**. Vector graphics innately have this property due to their reliance on mathematics, instead of context-free pixel grids. Each primitive have their own mathematic formulas which, obviously, stay the same no matter what the size of an image is. As such, the primitives can simply be re-rendered whenever the image is scaled [5][6]. Vector graphics also allow for easier editing [12][18], as you only need to modify individual polygons, lines, and curves, instead of dealing with individual pixels like you normally would when using raster images editors.

Vectorization would often be done manually. In a study by Hoshyari, et. al. [12], each of their raster images, which includes icons and small graphic illustrations, take
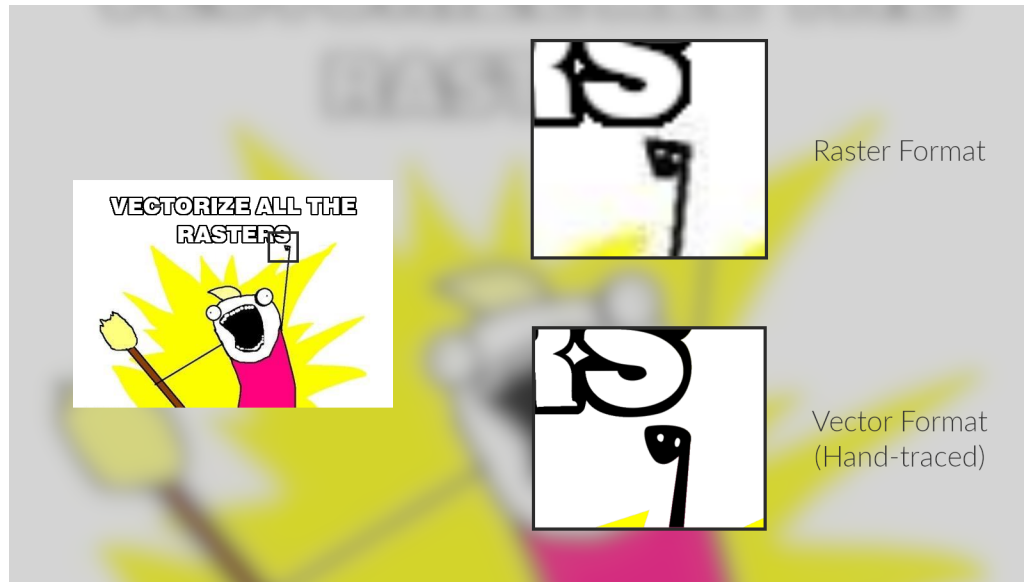
Figure 1.3: Zooming in closely at the same image, but one being in raster format (top right) and the other in vector (though hand-drawn; bottom left), quickly reveals the quality differences of both image formats. Raster images will show you individual pixels when close enough, but vectors will remain smooth.

.

30-45 minutes to be vectorized by an artist. More than 7 million man hours are being spent on vectorizing raster graphics in the United States every year, according to a survey in the PhD dissertation of J.R. Diebel entitled, "Bayesian image vectorization: The probabilistic inversion of vector image rasterization" [25]. Demand is, therefore, there for a robust vectorization algorithm.

Vectorization have been applied in many cases including, but not limited to, 2D maps and natural images. There are multiple methods that can be utilized in the vectorization of raster images. Their results differ from one method to another and even from one input to another, as many vectorization methods are fine tuned to specific inputs, such as those by Hoshyari, S., et. al. (semi-structured images) [12], Kopf, J. and Lischinski, D. (pixel art) [14], and Bessmeltsev, M. and Solomon, J. (line

drawings) [8]. Additionally, the ability to utilize GPUs for computational tasks has allowed parallelization of vectorization such as in the paper where a GPU was used to vectorize a video stream in real time [22].

Many vectorization methods target natural images. As noted by Hoshyari, S., et. al. many of these natural image vectorization methods utilize *image segmentation* to identify the portions of the image that will be converted into geometric primitives. These primitives are then filled with solid colors (e.g. such as what was done by Birdal, T. and Bala, E. [9]), gradient meshes (which are used in Adobe Illustrator and Corel CorelDraw [18][7]), and/or diffusion curves (such was the case in the paper by Xie, G. Sun, X., Tong, X., and Nowrouzezahrat, D. [21]). These vectorization methods produce differing results whose image qualities vary. Some results are as close as possible to the original raster image. Typically, gradient meshes and diffusion curves were utilized to achieve these results [21][18][7]. Others produce results with obvious colour segmentations, as seen in results that purely utilize solid colours (see [9] for an example).

Natural images are not the only raster images that are being vectorized. Images called semi-structured images are also candidates for image vectorization. *Semi-structured images* (SSIs) are images that consists of distinctly coloured regions and have well-defined boundaries [12]. Logos, cartoons, clip art, computer icons, and even simple graphical illustrations (such as flat 2D art) can considered to be semi-structured images. 60% of the 10 million images to be vectorized are semi-structured images [25]. This makes the demand for vectorizing these types of images evident. Various methods for vectorizing semi-structured images have been proposed by numerous papers. The common methodology of these proposals is that they attempt

to fit curves or Bezier splines on the boundaries of each region in SSIs and fill in the appropriate colour. However, each of these methods naturally have their own unique ways of fitting curves. In the work of Kopf, J., and Lischinski, D. [14], they use similarity graphs and additional intermediate steps in identifying the regions of an image, though their work is targeted at pixel art. Another paper by Yang, M., et. al. would directly optimize the shapes of individual Bezier segments connecting each boundary transition vertices to produce high fidelity vectorized images [12][25]. The work by Hoshyari, S., et. al. can be seen as a complement to the aforementioned paper. Their work utilizes human perceptual cues, primarily guided by Gestalt psychology, to produce vectorizations of semi-structured images that align much more closely to what viewers expect from a raster image [12].

An alternative method we can perform for image vectorization is to use machine learning via convolutional neural networks, a type of neural network that is well suited for image classification for their ability to learn various image features [11], to create vectorizations of semi-structured images by having the computer learn to produce Bezier curves of image region boundaries that align well with the expected vectorization. This is the method that this paper is proposing. A similar work, in that convolutional neural networks were for vectorization, to this is that of the paper of Simo-Serra, E., Iizuka, S., Sasaki, K., and Ishikawa, H. where they convert and simplify paper-and-pencil sketch drawings to vectorized images [17].

Chapter 2

Review of Related Literature

Image vectorization has been a long and well-researched field. One of the earliest works on the field dates back to 1982 with the paper by Jimenez, J. and Navalon, J. Their works have been focused on vectorizing digital images from natural scenes [13]. The techniques they have used, such as contour following, have been adapted in later works. Later works have made significant progress over their work and made use of hardware advancements such as GPU utilization for computational tasks.

The primary goal of vectorization is to support multiple resolutions without any loss in the original data. Vectorization is suitable for the task. However, various methods have been proposed that do not involve vectorization. In this chapter, we will explore the previous works done for vectorization and making images support larger resolutions.

2.1 Image Upscaling

Raster images are commonly used in representing images. Therefore, these types of images tend to be scaled to different resolutions. The process of scaling these types of images to higher resolutions is called *image upscaling* [12], and is the most common

forms of having raster images support larger resolutions. Naively scaling images would result in images with empty pixels between the shifted pixels, producing a mostly empty image. Thus, the empty pixels are coloured based on their position from, and colour and intensity values of their neighbouring pixels. This process is called *interpolation* [10]. Traditional interpolation algorithms can be classified into two types: Non-adaptive techniques, and adaptive techniques. Each of these types have their own method of resizing images and would produce different results, which may fit in with different objectives of different users [16]. Most recent works uses artificial intelligence to construct a scaled image with as little image quality loss as possible. These works uses neural networks, specifically convolutional and adversarial neural networks, to produce photorealistic upscaled versions of images with little to no distinguishable loss of image quality [23][20][15].

Non-Adaptive Techniques

Non-adaptive interpolation techniques only scale images horizontally, vertically, or both, by a certain scaling factor, the amount in which the image will be scaled to. They do not assume anything about the underlying image data, except that it is band-limited [14]. This makes them computationally cheap [16], but also suffer from artifacts such as sharp edge blurring and ringing artifacts [14]. Nevertheless, these non-adaptive image interpolation techniques have been widely used in the industry and is standard across different raster image editing programs, such as Adobe Photoshop and GIMP [2][1]. The common non-adaptive interpolation techniques used are:

- Nearest Neighbour

- (Bi)linear Interpolation

- (Bi)cubic Interpolation

Nearest Neighbour

Nearest Neighbour is the simplest interpolation method to understand. At the high-level, nearest neighbour simply enlarges the size of each individual pixel by a certain factor.

At the lowest level, though this **may** differ implementation-wise, the algorithm will refer back to the original unscaled version to obtain the appropriate colours for each empty pixels in between the shifted pixels in the scaled image. Each empty pixel in the scaled version will map itself to a corresponding pixel in the original image. This is done by obtaining the square coordinates of the empty pixel and dividing the coordinates by the scaling factor. Note that the coordinate system in images starts its origin from the top left, instead of the bottom left. We may acquire a decimal as a result of dividing the coordinates, of which we will floor the values. The resulting coordinates can be mapped to a pixel in the original image. We will then use the mapped pixel's colour as the colour of the empty pixel we are calculating the colour for. The interpolated pixels can be seen as a set I, as quantified by the equation below.

$$I = \{i | i = C((\lfloor \frac{e_x}{s} \rfloor, \lfloor \frac{e_y}{s} \rfloor))\}, s > 1, e \in E$$

s is the scaling factor, C is the function that gets the colour of a pixel based on its coordinate, and E is the set of empty pixels that are in between the shifted pixels when naively upscaled. The resulting image will look blocky since the pixels are just

enlarged. However, this will be ideal for pixel art [10].

(Bi)linear and (Bi)cubic interpolation

Both of (bi)linear and (bi)cubic interpolations schemes are fairly similar to one another. The colours between the empty pixels between them are based on the values of the colours of the shifted pixels.

Each empty pixel acquire most of their colour from the nearest shifted pixel and least from the farthest shifted pixel.

Adaptive Techniques

Super Resolution Methods

Super resolution methods is the process of generating high-resolution images from low-resolution image input. This allows

2.2   Vectorization of Images

Various methods have been proposed throughout the years in the pursuit of supporting larger resolution without any reduction in image quality through vectorization. Typically, vectorization methods target specific inputs, such as natural images or artist drawn images, as certain methods are unsuitable for different inputs. Hoshyari, S., et. al. states that vectorizations targeted at natural images frequently produce inconsistent results when applied to artist-drawn imagery such as logos, and simple graphic illustrations [12]. No matter what the methods used are, they produce vector graphics that will vary in quality, photorealism, and artistic look (view [21], [7], and [9] for a comparison of the results of various vectorization methods).

Despite vector graphics providing a compact and alternative form of representing [22], it is important to remember that due to the inherent characteristics of vector primitives where certain fine details of raster images cannot be accurately represented in vectorized form, vector images will only be giving approximations of the details of images [18]. Nevertheless, finding the perfect balance for image level of detail and image vectorization is an endeavour that is left as an exercise for users [9].

(Use for talking about Hoshyari's work) There are multiple possible vectorization outputs for a single raster image, with many outputs being similar to one another and are good enough for use. For this reason, technically, there is no "correct" solution and we can use outputs that we find good enough. However, most people already have an expectation of what the ???

Image vectorization can be dated as far back as the early 1990s, though experiments have started since 1982 at the earliest [13]. Commercial packages, both proprietary and open source, have image vectorization tools whose quality vary. The packages include Adobe Illustrator (Live Trace), Corel CorelDRAW (PowerTRACE), and Inkscape (based on Potrace [3]) [12]. Their wide adoption, though impressive, do not always immediately translate to quality vectorizations. The current available methods still have their own shortfalls and are still in active development. As a result, manual vectorization are still being performed in many industries that heavily require vectorizations [8].

Natural Images Vectorization

Many raster images widely used today are photographs. They contain fine details and lush colour depth that are prominent in the real world. These photographs are also called natural images in digital image processing. Increasing their resolutions would

entail using image upscaling techniques (see 2.1), especially the standard classical approaches. This would give the chance of producing low quality images [14] when super resolution methods are not utilized. Vectorization of these natural images are then an alternative to producing high quality higher resolution images.

Vectorization methods targeted at natural images consist the large body of work that deals with automatic natural imagery vectorization. The core method of these algorithms involves the reliance on edge detection and/or region segmentations to cluster large quantities of pixels together into larger regions [14]. These regions are then filled with either a solid color [9], or gradients via gradient meshes [18][7] or diffusion curves[21].

Vectorization of Semi-Structured Images and Artworks

Artwork, especially its subset, semi-structured images, is undeniably widely used around the globe. They are used to convey information, and express ideas. Both purposes would infer the necessity to maintain the high, or at least good enough quality images of those artworks to properly fulfill their tasks. Vectorization of these images would ensure that the quality of the image is kept at any resolution possible without any degradation. In the pursuit for quality vectorizations, many papers have been proposed that target specific inputs (such as pixel art, or small resolution images) and give out varying vectorization results.

A framework that optimizes bezigons, closed paths composed of Bezier curves, to match their raster counterparts as much as possible was proposed by Yang, M., et. al. as a vectorization process. Their work takes bezigons as input, of which they obtain either using pre-existing vectorization methods or extracting them from the raster image by segmenting the image into a set of regions then fitting piecewise Bezier

curves on the region boundaries. The input is called the *initial bezigons*. These initial bezigons are then optimized to reflect the raster image as close as possible. In optimizing the bezigons, they use a non-linear optimization algorithm (to be referred as *NLOAs* from hereafter) such as NEWUOA and conjugate gradient. Being an optimization-centered process, their process requires a method to evaluate their optimizations. Evaluating the vectorization output would involve getting the rasterization of the output and comparing it with the original raster image. Since there are various rasterization functions, a good enough rasterization function that works well with NLOAs is required. Using discontinuous and piecewise rasterization functions yields poor results during optimization. As such, a continuous rasterization function is required. Yang, M., et. al. chose to use a rasterization approach that utilizes a hierarchical Haar wavelet representation. The key component, which we view as a primary contribution of their work, is the energy E used to evaluate the vectorized outputs. E is the sum two parts: (a) the data energy, and (b) the prior energy. The data energy refers to the distance of the input raster image and the rasterization of the vectorization output. The prior energy refers to the severity of unreasonable bezigons. These bezigons would typically fall under one of the following categories, as per the work's authors intensive experimentation: (a) self-intersection, (b) false corners with small angle variations, (c) short handles, and (d) twisted sections. A larger E would indicate a more inaccurate vectorized output. As such, optimization of bezigons, as it is their work, would be to minimize E. Basing from their experiments, they produce high quality vectorizations that are superior to the results of Vector Magic and Adobe LiveTrace. However, the frameworks yields poor results when vectorizing noisy or low-resolution inputs. Assuming that the experimental results be any sign

for the actual theoretical speed, the execution time of this ranges from 10 seconds to 10 minutes, depending on the complexity of the shapes being vectorized. It is important to note that their implementation is not optimal and is written in Python, which provides a significant overhead. An implementation in a static, compiled language is expected to yield faster optimization speeds [25]. Basing from personal experience in hand vectorization, this framework resembles manual vectorization in that a bezigon is initially created that does not immediately match the raster image. The bezigon is later adjusted to fit the raster image boundaries. Being that they have proposed a framework, their work can be viewed as an optional and/or complimentary post-processing step to tweak pre-vectorized accuracy-ambiguous images, rather than a complete alternative or replacement of pre-existing and well-established vectorization methods, such as Adobe LiveTrace, and Potrace. At least one work, specifically that of Hoshyari, S., et. al., has already stated that their work is complimentary to this [12]. The only disadvantage to this framework is that it requires multiple iterations before settling on an optimal vector solution, which, consequentially, require some time to complete. It is also unclear whether the framework becomes more computationally expensive as the raster image size becomes larger as their paper does not indicate whether it is so or not.

Another form of digital art that is a target of vectorization is pixel art. Pixel art is a form of art where the level of detail is limited to the pixel level. This type of art tends to be blocky, and is reminisce of the art style of retro-era games. The work of Kopf. J, and Lischinski, D. specifically targets pixel art. As with other vectorization methods, their work detects edges of which they fit curves to create a final vector

result. Due to the inherent property of pixel art, their work is met with a few non-trivial challenges. The authors have identified four: (1) Every pixel matters, since a pixel can possibly represent a different feature given its colour is different enough from its neighbours, Additionally, each pixel can be viewed as an approximation of certain details in the overall image. (2) Pixels in a pixel art look connected at original scale, but disconnected when magnified. (3) The reduced level of detail in pixel art gives way to locally ambiguous configurations, in which it is unclear which pixels are part of which features. (4) Lastly, jaggies, collection of pixels that we perceive as forming jagged lines, in pixel art make differentiate features from pixelization artifacts hard. The proposed vectorization method of Kopf, J., and Lischinski, D. would first involve creating a similarity graph that would connect similar pixels with one another via edges. In this case, similar pixels would refer to pixels with similar colours. It should be noted that their work has some basis on Gestalt psychology in order to closely reflect what a human will imagine the final vector output to be. In integrating Gestalt psychology, they utilized heuristics to help identify which nodes in the similarity graph are to be connected or not. Once a similarity graph has been constructed, a Voronoi diagram, which consists of Voronoi cells, is made based on the similarity graph. The diagram now represents a rough shape of the object. The connected visible edges of the diagram, simplifyingly, are converted into quadratic B-spline curves. Related to the work done by Yang, M., et. al., optimization of the resulting quadratic B-spline curves are performed. Optimization involves the "minimization of a sum of per-energy nodes". A node, in this context, refers to a single endpoint of a curve. The energy E of each node i is computed to be the sum of both smoothness and positional terms, $E_s^{(i)}$, and $E_p^{(i)}$, respectively.

$$E^{(i)} = E_s^{(i)} + E_p^{(i)}$$

The smoothness term, $E_s^{(i)}$, measures the absence of curvature in the curve region influenced by node i. On the other hand, the positional term, $E_p^{(i)}$, measures the distance a control point node has moved away from their initial positions. The latter term is a prerequisite to prevent objects from changing too often. The control points are allowed to move freely within a relatively small region around their origin location, but are penalized when they deviate too far away. Not all nodes are optimized as a random walk is performed per iteration do identify which nodes will be optimized. The resulting vector image, after optimization, may have nodes that were not constrained. As such, these unconstrained nodes are computed a new location using harmonic maps. This tweaking results in distortion of cells in the Voronoi diagram. This work has yielded excellent results that closely resembles that of the original pixel art image. Comparing this with the results of previous works for vectorization, especially those with an inclination towards pixel art, this work has produced results that are comparable or superior (for example, when comparing against Adobe LiveTrace). The published execution time of the work opens it up for the possibility of use in vectorizating videos of retro games, especially when done in real time. The work, however, is limited hand drawn pixel art. Downsampled pixel art tend to look anti-aliased, which makes these inputs closer to natural images — inputs this work was not designed for. The results produced by this work will not always agree with human perception [14]. When viewed at a naive resolution independency angle, it would seem that this work is a step in the wrong direction as a simple nearest neighbour scaling algorithm can be applied to pixel art to support higher resolutions, as the algorithm is

compatible with this type of input. However, when looked from a far enough distance, pixel art is viewed to be smooth and continuous. Simply upscaling such images would have it retain its blocky nature. This work by Kopf, J. and Lischinski, D. offers a way to vectorize pixel art to **not only** support higher resolutions, but as well as providing a *smoother*, not blocky, output that closely matches the perception of the figure.

Chapter 3

Research Question or Problem Statement

Engineering theses tend to refer to a "problem" to be solved where other disciplines talk in terms of a "question" to be answered. In either case, this section has three main parts:

1. a concise statement of the question that your thesis tackles

2. justification, by direct reference to Chapter 1, that your question is previously unanswered

3. discussion of why it is worthwhile to answer this question.

Item 2 above is where you analyze the information which you presented in Chapter 3. For example, maybe your problem is to "develop a Zylon algorithm capable of handling very large scale problems in reasonable time" (you would further describe what you mean by "large scale" and "reasonable time" in the problem statement). Now in your analysis of the state of the art you would show how each class of current approaches fails (i.e. can handle only small problems, or takes too much time). In

the last part of this section you would explain why having a large-scale fast Zylon algorithm is useful; e.g., by describing applications where it can be used.

Since this is one of the sections that the readers are definitely looking for, highlight it by using the word "problem" or "question" in the title: e.g. "Research Question" or "Problem Statement", or maybe something more specific such as "The Large-Scale Zylon Algorithm Problem."

Chapter 4

Objectives

This part of the manuscript aims identify the specific objectives that your thesis aims to meet. These objectives must be in alignment with the problem identified in chapter 3.

Chapter 5

Describing How You Solved the Problem or Answered the Question

This part of the thesis is much more free-form. It may have one or several sections and subsections. But it all has only one purpose: to convince the examiners that you answered the question or solved the problem that you set for yourself in Chapter 3. So show what you did that is relevant to answering the question or solving the problem: if there were blind alleys and dead ends, do not include these, unless specifically relevant to the demonstration that you answered the thesis question.

Chapter 6

Describing How You Validated Your Approach.

Chapter 7

Stating Your Results and Drawing Insights From Them.

Chapter 8

Summarizing Your Thesis and Drawing Your Conclusions.

Appendix A

What should be in the Appendix

What goes in the appendices? Any material which impedes the smooth development of your presentation, but which is important to justify the results of a thesis. Generally it is material that is of too nitty-gritty a level of detail for inclusion in the main body of the thesis, but which should be available for perusal by the examiners to convince them sufficiently. Examples include program listings, immense tables of data, lengthy mathematical proofs or derivations, etc.

Bibliography

[1] 6.22.scale image. `https://docs.gimp.org/en/gimp-image-scale.html`.

[2] Photoshop image size and resolution. `https://helpx.adobe.com/photoshop/using/image-size-resolution.html{\\#}resampling`.

[3] Potrace - inkscape wiki. `http://wiki.inkscape.org/wiki/index.php/Potrace`.

[4] Titan xp graphics card with pascal architecture — nvidia geforce. `https://www.nvidia.com/en-us/titan/titan-xp/`.

[5] Raster vs vector. `https://vector-conversions.com/vectorizing/raster_vs_vector.html`, n.d.

[6] Raster vs. vector graphics: What's the difference? `https://pixellogo.com/blogs/pixellogo-blog/raster-vs-vector-graphics`, n.d.

[7] P. J. Barendrecht, M. Luinstra, J. Hogervorst, and J. Kosinka. Locally refinable gradient meshes supporting branching and sharp colour transitions. *The Visual Computer*, pages 949–960, 2018.

[8] M. Bessmeltsev and J. Solomon. Vectorization of line drawings via polyvector fields. *CoRR*, abs/1801.01922, 2018.

[9] T. Birdal and E. Bala. A novel method for vectorization. *CoRR*, abs/1403.0728, 2014.

[10] Computerphile. Resizing images. `https://www.youtube.com/watch?v=AqscP7rc8{\\_}M`, 2016.

[11] Juan Carlos Gonzales. Use of convolutional neural networks for image classification. `https://www.apsl.net/blog/2017/11/20/use-convolutional-neural-network-image-classification/`, 2017.

[12] S. Hoshyari, E. A. Dominici, A. Sheffer, N. Carr, Z. Wang, D. Ceylan, and I. Shen. Perception-driven semi-structured boundary vectorization. *ACM Transactions on Graphics*, 37(4):1–14, 2018.

[13] J. Jimenez and J. Navalon. Some experiments in image vectorization. *IBM Journal of Research & Development, volume = 26 no = 6 month =*.

[14] J. Kopf and D. Lischinski. Depixelizing pixel art. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4):99:1 – 99:8, 2011.

[15] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, abs/1609.04802, 2016.

[16] P. Parsania and P. Virparia. *International Journal of Innovative Research in Computer and Communication Engineering*, (12):7409.

[17] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa. Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM Trans. Graph.*, 35(4):121:1–121:11, July 2016.

[18] J. Sun, L. Liang, F. Wen, and H. Shum. Image vectorization using optimized gradient meshes. Association for Computing Machinery, Inc., July 2007.

[19] Techquickie. How do vector graphics work? `https://www.youtube.com/watch?v=W2xknX3k6FY`, 2016.

[20] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers. A fully progressive approach to single-image super-resolution. June 2018.

[21] G. Xie, X. Sun, X. Tong, and D. Nowrouzezahrai. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Trans. Graph.*, 33(6):230:1–230:11, 2014.

[22] X. Xiong, J. Feng, and B. Zhou. Real-time image vectorization on gpu. *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016)*, 1:143–150, 2016.

[23] A. Yang. A.i. gigapixel an inside story topaz labs. `https://topazlabs.com/a-i-gigapixel-story/`, 2018.

[24] C. Yang, C. Ma, and M. Yang. Single-image super-resolution: A benchmark. pages 372–386. Springer International Publishing, 2014.

[25] M. Yang, H. Chao, C. Zhang, J. Guo, L. Yuan, and J. Sun. Effective clipart image vectorization through direct optimization of bezigons. *IEEE Transactions on Visualization and Computer Graphic*, Feb. 2016.