MSc Computer Science (Conversion), COMSM2202

University of BRISTOL

Research Review

Faster Conversion of Bitmaps into Scalable Function Graphics using a Pre-trained Neural Network Database & Multi-threaded Fine-tuning

---

Supervisors: Chris Preist & Daniel Schien

Author: Alexander Lorimer

07 May, 2016

Contents

0.0 **Executive Summary** 1

0.1 - Added Value 1

1.0 **Introduction** 2

1.1 - Background & Motivations 2

1.2 - Aims & Objectives 2

1.3 - Research Areas 3
- 1.3.1 Image Interpolation 3
- 1.3.2 Neural Network Regression 3
- 1.3.3 High Performance Computing 3

1.4 - Shape of Argument 4
- 1.4.1 Overview of Industry Standard Image Scaling Tehniques 4
- 1.4.2 Introduction to Scalable Function Graphic Conversion 4
- 1.4.3 Improvements to PhotoFunction Software 4
- 1.4.4 Further Considerations 5

2.0 **Overview of Industry Standard Image Scaling Techniques** 6

2.1 Spine Interpolation 6
- 2.1.1 - Bicubic Spline 6
- 2.1.2 - S-Spline Max 7

2.2 Resolution Independence 7
- 2.2.1 - Vector Graphics 7
- 2.2.2 - Fractal Images 9
- 2.2.3 - Fractals vs. Vectors for Resolution Independent Photographs 10

2.3 Neural Networks & Super-resolution 10
- 2.3.1 - Convolution Neural Networks 10
- 2.3.2 - Other Neural Network Projects 11

2.4 Summary 11

3.0 **Introduction to Scalable Function Graphic Conversion** 12

3.1 Overview 12
- 3.1.1 - The Conversion Process 12
- 3.1.2 - Exta-pixel Generalisation 12
- 3.1.3 – High-quality Sub-pixel Generalisation 13

3.2 Relative Advantages & Limitations 13
- 3.2.1 - SFGs vs. Fractals: File Format Redundancy 13
- 3.2.2 - SFGs vs. Splines: A Pixel as a Domain 14
- 3.2.3 - SFGs vs. Splines: Runge's Phenomenon & the Variable n 14
- 3.2.4 - SFGs vs. Splines: Runge's Phenomenon & Curriculum Learning 15
- 3.2.5 - SFGs vs. CNNs: Training on Demand 16
- 3.2.6 - SFGs vs. CNNs: Approaching the Global Minimum 16
- 3.2.7 - Aesthetic Comparison 16

3.3 Summary 17

4.0 **Improvements to the PhotoFunction Software** 18

4.1 Machine Learning 18
4.1.1 - Tile Size & the Global Minimum 18
4.1.2 - Initialisation & Pre-training 18
4.1.3 - Data Pre-processing & Feature Space Reduction 19
4.1.4 - Neuron Redundancy & Adaptive Growth 20

4.2 High Performance Computing 20
4.2.1 - Concurrency 21
4.2.2 - Memory Caching 21
4.2.3 - Precomputations 21
4.2.4 - Compiler Optimisation 22

4.3 - Image Filters 22

5.0 **Further Considerations** 23

5.1 - Evaluating Results 23

5.2 - The Java Language 24

6.0 **Conclusion** 25

6.1 - Future Research 26

7.0 **Work Plan** 27

7.1 - Expected Time-line 27

7.2 - Risk & Contingency Analysis 29

8.0 **Bibliography** 31

9.0 **Appendices** 34

9.1 - Appendix 1, Artomatix Communication 35

9.2 - Appendix 2, PhotoFunction Code Organisation Overview 36

List of Figures

1 – Bicubic Interpolation 6

2 – Optimised Gradient Mesh 8

3a – Original Chick 9
3b – Vectorised Chick 9
3c – Retouched Chick 9

4 – Fractal IFS 9

5a – Original Sunset 12
5b – Extrapolated Sunset 12

6a – Identified Tile, Original Sunset 13
6b – Nearest Neighbour, Original Sunset 13
6c – Bicubic, Original Sunset 13
6d – SFG, Original Sunset 13
6e – Fractal, Original Sunset 13

7 – Pixel Domain 14

8 – Sub-pixel 14

9 – Runge's Phenomenon 15

10a – SFG 17
10b – Waifu2x 17
10c – S-s Max 17
10d – Fractal 17
10e – Bicubic 17
10f – Nrst N. 17

11a – Lenna, unprocessed 19
11b – Lenna, 4 by 4 preprocessed tiles 19

12 – Unsharp example 22

Executive Summary
0.0

A wide range of algorithms have been developed over time to approach the non-trivial problem of enlarging bitmap images, each providing a different balance between aesthetic quality (the preservation of smooth gradients versus sharp edges) and processing efficiency. Aiming for aesthetics over speed, artificial neural networks have recently been used to precisely map the X and Y coordinates of an image's pixels to corresponding red, green and blue (RGB) colour channels. This time-consuming process converts a bitmap image into a complex mathematical function, or Scalable Function Graphic (SFG). SFG files are significantly larger than the original bitmap, however they act as a resolution independent representation of an image, capable of outputting colour vectors at X and Y coordinates that were not defined in the original (low-resolution) bitmap. By this process of generalisation, the mathematically modeled image can be rendered at any scale, with minimal impact on aesthetic quality.

The aim of this research is to build on recent SFG technology, by exploring how the efficiency of conversion and storage can be improved (using a database of pre-trained neural networks that can be concurrently fine-tuned), while preserving or improving aesthetic quality.

The project will require research and development in the areas of image interpolation, neural network regression and high performance computing, consisting of 60% type I (software development) and 40% type II (investigation). It will involve significant programming and refactoring, as further development on top of an existing project. The development process will be guided by continuous testing and experimentation, gauging factors such as the suitability of learning algorithm parameters and the generality of features on which neural networks are pre-trained.

The objectives (and final deliverables) are:

- A broadly applicable database of pre-trained neural networks that can be retrieved by closest match to image characteristics (providing a head start on SFG conversion).
- An improved SFG learning algorithm, growing or adding neurons during fine-tuning until sufficient complexity has been achieved (reducing neuron redundancy and file size).
- A concurrent implementation that better utilises memory caching and precomputation.
- An implementation with previous code re-factored for readability and efficiency.
- A set of image filters to further enhance perceived aesthetic quality.
- An evaluation of this implementation.

The evaluation will compare performance to the previous SFG conversion software, and to leading commercial algorithms, in terms of speed, storage efficiency, and aesthetics as well as accuracy (using human participants and mean squared error).

0.1 Added Value

The author has an implicit understanding of the limitations of the current implementation and areas that need improvement, having independently developed the Scalable Function Graphic format and conversion software. SFGs are capable of being rendered at significantly larger scales than the original bitmap, with output quality that rivals and even surpasses industry standards (fractal image enlargement and spline fitting). However, the current implementation is limited by poor performance in terms of processing speed and file storage efficiency. This research aims to develop an updated implementation improving on these issues. With complete access to the current software, benchmarks can be run to compare performance between the previous and updated implementation. On addressing these challenges, it is hoped that commercial viability for SFG conversion software will be improved.

Future research and development will likely involve a neural network learning algorithm that can more accurately approximate sub-pixel output averages (without impeding error backpropagation), to further improve scaled image sharpness.

Introduction
1.0

1.1 Background & Motivations

Bitmap (or raster) images are those that are encoded as a sequence of discrete values that directly represent colours at each of a fixed number of pixels. This is a popular format, typical of scanners and digital cameras.[1] It can store detailed photographic images, however it does not naturally support scaling to larger dimensions without a loss of resolution (as pixels become individually perceptible). The result is a reduction in aesthetic quality, or continuity of content, in the displayed image.[2]

On the other hand, PhotoFunction is a prototype piece of software that converts bitmaps into Scalable Function Graphics or SFGs (developed by the author). SFGs represent images as connection weights (signal strengths) between neurons within an artificial neural network, acting as a mathematical model of image content. Neural networks are effectively complex functions that translate inputs into corresponding outputs, with an ability to generalise (make educated guesses) about inputs that they have never seen before.[3] In the case of SFGs, inputs represent pixel coordinates and outputs represent colour values. As a consequence, having converted a bitmap image into an SFG, X Y co-ordinates that were not defined in the original bitmap (such as sub-pixel co-ordinates) can be rendered based on the co-ordinate/colour relationship encoded by the SFG. SFGs therefore naturally support rendering at much higher resolutions than the original bitmap image, with minimal impact on aesthetic quality. The clarity of enlarged SFGs even rivals state-of-the-art, commercial image scaling algorithms such as S-spline Max and fractal images.[4]

However PhotoFunction and the SFG file format suffer from significant limitations in terms of file storage efficiency and conversion speed (taking anywhere from several hours to days depending on bitmap size and complexity).[4] Nevertheless, online responses to the December 2014 launch of the PhotoFunction prototype indicated potential usefulness, along with the need for further development. As mentioned by the budget graphic design company, Titanium Alley Graphics, “there's much work to be done but PhotoFunction, still in prototype stage of development, is producing stunning results.”[5] Referring to SFGs, one user in a Doom9 forum post remarked “I have never seen black magic this powerful”[6] while another mentioned “I've got the original program now, but the problem is it's extremely slow.”

The subsequent premise of this project is that there is a need for a further developed PhotoFunction implementation, to significantly improve SFG conversion speed and file size without negatively impacting (or potentially further improving) aesthetic quality.

1.2 Aims & Objectives

The aim of this research is to implement a series of improvements to the current PhotoFunction implementation, focused on reducing conversion times, reducing SFG file size, and boosting aesthetic quality by applying known image filters. The objectives are to provide the following:

- A broadly applicable, read-only database of pre-trained neural networks that can be retrieved by closest match to image characteristics (providing a head start on the conversion process). To ensure efficient storage, miniature neural networks will be store in the database and fine-tuned on demand to specific images, requiring neuron growth.
- An improved SFG learning algorithm, growing neurons in each hidden layer during fine-tuning until sufficient complexity has been achieved (reducing neuron redundancy and SFG file size, compared to the current one-size-fits-all neural network size).
- An adapted implementation that runs concurrently, and takes advantage of memory caching and precomputed lookup tables. This will ensure that the conversion process runs more efficiently, better utilising available resources.

- An implementation with previous code refactored for readability and efficiency. The current implementation has extremely long methods (with redundant variables and messy indentation). This hinders inline compiler optimisations and maintainability.
- A set of known image filters to further enhance perceived aesthetic quality. Unsharp will be applied to increase edge contrast, and noise will be explored as a means of suggesting photographic detail when scaling by large factors. These filters are already employed by other image scaling software.
- An evaluation of this implementation in terms of speed, storage efficiency, and accuracy as well as aesthetics (using mean squred error, in addition to a human participant sample). Benchmarks will be run against the previous SFG implementation, and against established state-of-the-art image scaling algorithms.

1.3 Research Areas

This project will involve research and development in several identifiable areas; these are image interpolation, neural network regression, and high performance computing.

1.3.1 **Image Interpolation**

Image interpolation technologies utilise extremely varied approaches, but all attempt to solve the problem of increasing the spatial resolution of an image while establishing a trade-off between speed and aesthetic quality. Research in this area will be necessary to avoid re-invention of the wheel, and to provide insight into how SFG conversion might be best improved. Specifically spline interpolation, resolution independent file formats, and super-resolution-based machine learning are significant areas that will require critical examination.

The novelty of PhotoFunction and the SFG format will therefore not be presumed, and will instead be established as part of this literature review. In the process of surveying relevant research and technologies, this document will highlight SFG conversion issues that limit efficiency, and will explain how these will be addressed and improved. Also, due to the fact that a large amount of information about the specific mechanisms of PhotoFunction and the SFG file format remains unpublished, references to external sources will not always be possible. PhotoFunction source code has therefore been uploaded to an online, password protected repository in an attempt to mitigate this problem. Appendix-2 provides the repository URL and temporary password details, along with an overview of the current PhotoFunction code organisation, explaining the functionality of each class. The information in Appendix-2 and the online repository can be consulted by the reader in case instances arise where further clarification is needed.

1.3.2 **Neural Network Regression**

Neural network training (specifically regression) will feature as the most prominent area of research and development. This is the main technology upon which the PhotoFunction implementation has been built, and provides the largest space for potential improvement. Specifically, pre-training, data pre-processing, and adaptive network growth are the areas of machine learning that will be explored. Development in each of these areas will facilitate a database of efficiently stored and broadly applicable, pre-trained networks that can be retrieved, fine-tuned and adapted based on image characteristics.

1.3.3 **High Performance Computing**

As a means of significantly improving the speed of SFG conversion, high performance computing concepts such as concurrency, memory caching and precomputation will be investigated and utilised. In particular, designing a concurrent system for SFG conversion is likely to introduce several problems involving memory inconsistency between threads. This could result in significant discontinuities within the mathematical model of an SFG, causing unwanted visual artifacts. Trial and error may be involved to find an effective balance between memory consistency (using volatile access modifiers) and runtime speed.

1.4 Shape of Argument

1.4.1 **Overview of Industry Standard Image Scaling Techniques – Chapter 2**

Chapter 2 will introduce the widely used and state-of-the-art class of image scaling algorithm known as spline Interpolation. Significant differences exist in the way that SFGs and splines express intermediate pixel values, although both systems work by establishing a set of interpolation coefficients. This overview will therefore feed into a discussion of differentiating factors (in Chapter 3**,** relating to Runge's Phenomenon and the nature of pixel interpretation).

Chapter 2 will additionally discuss vector graphics and fractal images as resolution independent image formats. Resolution independence describes the quality of information that is encoded in a way that does not prescribe a particular scale (as with SFGs). Establishing the benefits and limitations of these pre-existing formats will further feed into Chapter 3, which places PhotoFunction and SFGs in context.

Finally Chapter 2 will explore significant research in the area of neural network-based image enlargement (or super-resolution). As with interpolation, super-resolution is a term used in certain contexts to refer to a process of image enlargement, such as where convolution neural networks (CNNs) are involved. CNNs differ fundamentally in structure and approach from SFGs, but have recently achieved industry standard (or superior) results in terms of high-resolution image reconstruction.[7]

As such this chapter, establishes a varied assembly of state-of-the-art image scaling techniques (S-splines, fractals, and CNNs), and sets up a discussion of the relative advantages and limitations of the current SFG format and conversion process (in Chapter 3).

1.4.2 **Introduction to Scalable Function Graphics – Chapter 3**

Building on the information presented in Chapter 2, Chapter 3 introduces details of the current PhotoFunction/SFG implementation, establishing relative benefits as well as inherent limitations of representing image information as a network of artificial neurons (compared to fractal coefficients). It also serves to emphasise the novelty of SFGs by pulling together a discussion of spline interpolation and neural network regression. This will essentially act as a snap-shot of PhotoFunction and SFGs as they are now (before embarking on the summer implementation project).

Together this information will be used to explain why SFG conversion can be very slow, and why SFG file sizes can be very large. Yet, a visual comparison of the previously discussed image scaling approaches will demonstrate the aesthetic quality and potential of SFGs. This chapter will therefore establish impetus for further research and development, and act to frame an in-depth discussion (in Chapter 4) of improvements that can be made to PhotoFunction software and the SFG format.

1.4.3 **Improvements to PhotoFunction Software – Chapter 4**

Chapter 4 will explain how the application of certain machine learning and high performance computing practices can help improve SFG conversion speed and file efficiency.

In the area of machine learning, such practices include neuron growth (adapting network sizes to image complexity), neural network pre-training (providing a head start on bitmap conversion), and data pre-processing (filtering training data to better relate to neural network learning requirements). Together these alterations will help facilitate a database of efficiently stored, broadly applicable, miniature neural networks that can be adapted to a variety of images before subsequent fine-tuning. In particular, this will reduce SFG file redundancy (an identified weakness, compared to fractal images) while reducing the need for live training (an identified weakness, compared to super-resolution CNNs).

This machine learning discussion will be followed by an overview of high performance computing concepts that can be employed to better optimise the neural network fine-tuning process for runtime speed. These include parallel computing and the use of precomputed lookup tables. Problems that may be encountered with these approaches will be discussed, in addition to measures that will be employed in worst case scenarios.

Finally, addressing aesthetics as opposed to efficiency, a brief overview of two image filters will be presented. This section will explain how leading image enlargement software packages employ finishing-touch image filters to further enhance image quality; specifically unsharp and noise. These will be described as simple, worthwhile features to add to the PhotoFunction implementation.

Chapter 4 will therefore establish improvements relating to runtime performance, file storage efficiency, and aesthetics, which will become the focus of an updated PhotoFunction implementation. This will be followed by a short chapter that discusses further considerations pertaining to the development of this implementation.

1.4.4 **Further Considerations – Chapter 5**

Having identified the intentions for further development, Chapter 5 will briefly justify Java as the choice of programming language after presenting an overview of the intended approach to evaluating this project. A preliminary investigation of the mean squared error of SFGs will be presented, comparing against identified state-of-the-art image scaling algorithms and revealing reasons for involving human participants in the final evaluation process.

Chapter 5 will be followed by a conclusion that summarises and connects the ideas that have been reviewed in the body of this document, in addition to providing an overview of possible future work that could further improve SFG technology. A work-plan presented at the end of the document will then show the road map for the upcoming process of research and development, establishing staged time allocation and project specific contingency plans.

Overview of Industry Standard Image Scaling Techniques 2.0

As discussed in the introduction, bitmap graphics provide relatively little information that can be used to directly guide image enlargement. For this reason algorithms exist that interpolate between pixels when bitmap graphics are required to be displayed at higher resolutions.[8]

The first section of this chapter (Spline Interpolation) will introduce two important algorithms that use splines (piecewise polynomials) to enlarge images. This will lead directly onto the section Resolution Independence, describing a circumstance where images are instead stored in a format that naturally supports rendering at practically any scale. Finally, neural network-based image scaling approaches will be discussed; specifically convolution neural networks, which have recently demonstrated an improvement on state-of-the-art approaches to super-resolution (focused specifically on error minimisation). Each of these varied approaches represent the industry standard in terms of photographic image enlargement, while directly relating to the mechanisms of PhotoFunction (either through the utilisation of machine learning, the establishment of resolution independence, or the activity of curve fitting).

Each will act as visual and algorithmic points of comparison to SFGs later in the document (in Chapter 3 and Chapter 5), and will set the context for an in-depth discussion of SFG functionality and weaknesses that require further development.

2.1 Spline Interpolation

2.1.1 **Bicubic Splines**

One of the most frequently used, freely available image scaling approaches (set as default by most image editing packages such as Photoshop and GIMP[9]) is bicubic spline interpolation. In this approach, sub-pixel values are rendered based on a weighted sum or average of 16 surrounding pixel values. These are weighted by two cubic functions, described by the equation and illustration below (Fig. 1); sourced from Digital Image Processing, William K. Pratt[10]. ***F(p',q')*** denotes the sub-pixel value, ***F(p+m, q+n)*** denotes each of 16 known pixel values, $R_c()$ denotes the cubic functions, and **a** and **b** represent distances from the nearest pixel.

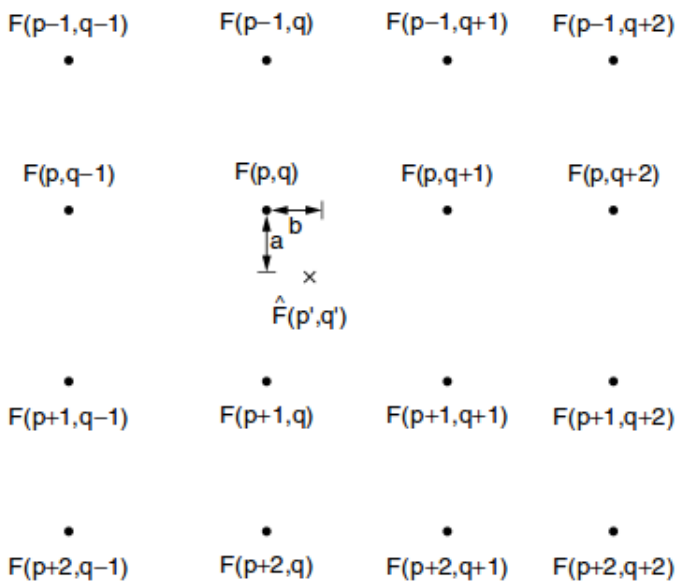


Fig. 1 Bicubic interpolation[10]

$$F(p', q') = \sum_{m=-1}^{2} \sum_{n=-1}^{2} F(p+m, q+n) R_C\{(m-a)\} R_C\{-(n-b)\}$$

The cubic functions, $\boldsymbol{R_c}$ (together forming a bicubic function), are created by a process of spline fitting, in which coefficients are calculated that force the spline to meet each data point while maintaining continuity. This process requires sets of cubic equations to be solved simultaneously, such that they are equal where they join with continuous derivatives (degrees of slope).

While bicubic interpolation is popular and freely available, a significantly higher quality, spline-based commercial algorithm is known as S-spline interpolation, developed and utilised by PhotoZoom Pro.[11]

2.1.2 **S-spline Max**

The granted patent, published in 2000,[12] reveals that a weighted pixel system is similarly employed. However, in addition to taking into account distances **a** and **b**, this system also takes into account a local minimum and maximum pixel value, in addition to an "associated dynamic value" that provides a measure of local pixel hardness (or sharp contrast). S-spline Max is the latest generation of this patented system, touted as the industry leading algorithm for photographic image enlargement. Unlike the bicubic spline approach, S-spline algorithms are proprietary and costly.

Image scaling techniques, such as bicubic spline and S-spline Max, are useful because of the prevalence and limitations of bitmap graphics, which store image information in an intuitive manner. However they suffer from the limitation of resolution dependence (overly dependent on the resolution at which the image data has been sampled). As illustrated by the bicubic expression above, pixel values directly feature as variables in the calculations used to establish new, interpolated pixels (resolution dependence is evident in the formula used to model the image).

However, resolution independent alternatives to bitmaps do exist. Resolution independence means that the image is stored as a mathematical descriptions of features, without any reference to pixels. Instead, all pixel data is generated when the image is displayed based solely on the mathematical description, which gives no indication of a prescribed resolution or scale. The more sophisticated the mathematical description, the greater the capacity for generating visually plausible high-resolution information.[13] Vector graphics and fractal images are both types of resolution independent image formats[2][14] that have relative strengths and limitations. These will be discussed in the following section.

2.2 Resolution Independence

2.2.1 **Vector Graphics**

Vector graphics are a popular alternative to bitmaps, where image data is encoded as mathematically defined geometric objects. Representative ellipses, polygons, lines, and curves are stored as precise vectors, and so the image can be rendered at any resolution, sampling pixel values from the vector graphic's mathematically defined geometric objects. Vector graphics typically require significantly less storage space than bitmaps, but this comes at the expense of capturing photo-realistic detail. This format is therefore only currently useful in cases where exact photorealism is not required.[2]

A relatively recent vector graphic feature (of editors such as Adobe Illustrator and CorelDraw) is referred to as a gradient mesh.[15] Instead of relying on geometric primitives, a gradient mesh represents image data as an intricate deformed grid, allowing for a higher degree of photorealism. Each vertex specifies a gradient and colour that are interpolated over grid cells (typically using a bicubic interpolation algorithm).[16] Gradient mesh tools tend to require manual guidance (as a labour and time intensive task), requiring skill to produce good results.[16] During this process a user drags and assigns colours to control points (or vertices), aligning with a reference bitmap image and sampling colours from appropriate pixels.[17]

In 2007 a patent was filed (and later granted in 2014)[18] for a semi-automated approach to generating "optimised gradient meshes". This requires the user input to identify separate objects in the image (using a cut-out or lasso tool) after which an energy minimsation algorithm fine-tunes mesh vertices (minimising the residual error between the bitmap image and the rendered gradient mesh).[15] Fig. 2 shows an original bitmap image (left) that has been rendered as an optimised gradient mesh (right). Acknowledged in the original paper, this optimisation tool is well suited to modeling smooth objects, but it is not as well suited to modeling details that are characterised by rapid changes in colour intensity.[15] As can be seen in Fig. 2, details such as eye lashes and hair have become blurred during the conversion to gradient mesh.

In 2009 a completely automated approach to gradient mesh generation was presented in the paper "Automatic and Topology-Preserving Gradient Mesh Generation for Image Vectorization",[16] along with a filed patent application.[19] This approach is nevertheless still incapable of modeling photographic detail, as acknowledged by the authors.[16]

Research that has focused on this particular problem includes Jeschke et a 2011,[20] in which textures are created by fitting Bezier curves in addition to parameters for procedurally generated noise. Fig. 3a shows an original bitmap image, while a version that has been vectorised using this approach is given in Fig. 3b, and Fig. 3c shows the result after manual retouching. Photographic detail is still lost and manual editing is required to produce satisfactory results. Also, as described by the authors, overall texture characteristics are mimicked instead of faithfully modeled.[20]

These varying approaches to vectorisation are currently insufficient to produce resolution independent formats that can allow accurate scaling of photographic images. Generating high quality sub-pixel information requires precise modeling at the level of pixels. An entirely different approach to resolution independence known as fractal image enlargement achieves this, advertised as an industry standard in photographic resolution independence.[21]

Fig. 2, Optimised Gradient Mesh[15]

Fig. 3a, original chick[20]

Fig. 3b, vectorised chick[20]

Fig. 3c, retouched chick[20]

2.2.2 **Fractal Images**

Fractals are mathematically defined patterns that exhibit self-similarity at increasingly diminishing scales, [22] facilitating infinite zooming based on a recursive mathematical description. Their application for scaling photographic images relies on the fact that photographs posses a degree of self-similarity that can be defined in term of affine transforms. These are transformations that preserve affinity, in terms of ratios and collinearity between lines and points[23] (such as rotating, scaling, skewing and reflecting).

Fig. 4 shows a fractal image that has been generated with two affine transformation functions (using constants obtained from the book "Fractal Geometry: Mathematical Methods, Algorithms, Applications"[24] - the fractal image has been rendered using author's implementation).

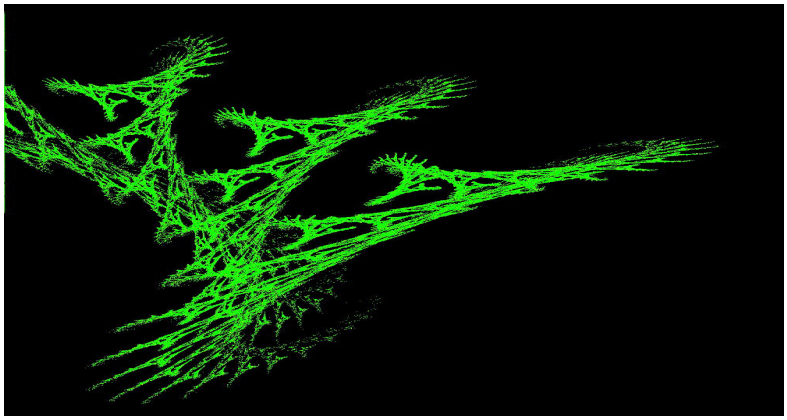


Fig. 4, Fractal IFS

As can be observed, Fig. 4 exhibits a pattern that repeats itself at increasingly smaller scales. The complex structure (which has been encoded using only twelve values) can be rendered at any scale without any loss of quality. An iterated function system (IFS) has been applied to render this fractal (as is the case with fractal image enlargement software). Iterative functions feed their resulting values back as function input a certain number of times, transforming the values. Previous X and Y co-ordinates are therefore fed into the iterated function to produce new X and Y co-ordinates as outputs. RGB values can also act as iterated inputs to more complex IFSs (necessary for photographic image enlargement).[24] But in the case of Fig. 4, an arbitrary colour has been rendered at the X Y co-ordinates of each iteration.

Describing a photographic bitmap as fractal coefficients is a non-trivial problem. Effectively, existing self-similar affine transformations are identified between partitioned blocks or tiles in the image. The image can then be encoded as the constants that describe these transformations. A special property of IFSs are the irrelevance of initial pixel information (as initial input to the iterated functions). The output will always converge over multiple iterations on an identical and stable result (a rendering of the encoded image) regardless of the initial input. The original bitmap information is therefore no longer relevant once the image has been encoded by iterated affine transformations.[24]

Described in terms of self-similar characteristics, sub-pixel values can now be rendered with respect to an identified fractal pattern, producing high quality results in addition to requiring less information than the original bitmap.

2.2.3 Fractals vs. Vectors for Resolution Independent Photographs

In addition to requiring less information that the original bitmap, fractal image conversion can be performed in a matter of seconds, while gradient meshes typically require manual guidance (taking significantly longer).

Gradient meshes are also encoded using significantly less information than original bitmaps. But, this is an unfair comparison given that the format is not capable of representing detailed photographic information. The more complex vector graphic gradient meshes are required to become, the more information required to describe them.[25]

Aside from file compression, fractal images scaling software (such as Perfect Resize[21]) represent an industry standard for photographic image enlargement, drawing on self-similar features within the image as a whole. These relationships do not necessarily reflect the nature of original sub-pixel information, but tend to convince the eye of perceiving high clarity visual information. For this reason, in the book Fractal Geometry: Mathematical Methods, Algorithms, Applications,[24] resolution enhancement is purposefully used as a term to refer to fractal image scaling as opposed to resolution enlargement.

However, there are also image scaling approaches that focus more on reconstructing original information, such as deep convolution neural networks, where the training criteria is to map low resolution to high resolution ground truth images. Convolution neural networks are discussed in the final section of this chapter.

2.3 Neural Networks & Super-resolution

Neural networks can be used to model either discrete or continuous relationships between inputs and outputs. The former scenario is typically referred to as classification. In the field of computer vision, RGB values typically form the inputs to a neural network where the output acts as a discrete classification of image content (such as cat or dog). Mapping a continuous relationship, as opposed to a discrete relationship, is referred to as regression, where inputs relate to a continuum of real numbers instead of a discrete set.[26] Neural network-based image enlargement can utilise either classification or regression, as discussed in the following section.

2.3.1 Convolution Neural Networks

Much work has been conducted using convolution neural networks (CNNs) to enlarge photographic images. CNNs take pixel data as input, as they are structured to act as a complex filter that convolves over an image, providing a transformation from input to a particular output.[27]

Classification-based image scaling approaches include Gustafson and Meyer,[28] in which spline interpolation parameters are adapted according to the classification of objects in corresponding parts of the image. The results show improved interpolation quality but are compared only to standard bicubic interpolation. Tree-based resolution synthesis[29] is another approach in which image segments are first classified by a CNN, after which standard interpolation algorithms are applied (such as bicubic or bilinear) that best suite the characteristics of each image class.

However, as opposed to classification, more successful work has been conducted in the area of regression, where low resolution pixel data acts as input, and higher resolution pixel data is directly generated as output (often described in this context as super-resolution). Earlier research in CNN super-resolution resulted in scaled images that appear marginally sharper or more detailed than standard interpolation algorithms, such as Go et al 2000,[30] Hu 2004[31] and Aokage et al 2005.[32] Each of these approaches use relatively shallow

network architectures consisting of 1 to 3 layers. However more recently, an online 2015 post[33] in the engineering section of Flipboard (the social-network aggregation platform) reveals that the company has been conducting research in the use of deep learning (many layered networks) for the purpose of super-resolution. According to the post, Flipboard's engineering team have used a CNN consisting of 8 layers trained on low resolution/high resolution pairs. Good results were reported, but bicubic interpolation was used as the only basis of comparison, with a tentative comment about the resulting network; "While this wont have its place everywhere within our product, we feel it was a good cursory step forward to improving quality."[33]

Certainly, other deep learning architectures recently appear to offer significant potential in the area of super-resolution. Reporting far better results Chao Dong et al 2014[33] trained a deep CNN to map low to high resolution with considerable accuracy. As mentioned by the authors, "the image super-resolution problem has not witnessed the usage of deep learning techniques to the best of our knowledge." Their results show an improvement on prior state-of-the-art super-resolution algorithms based on sparse coding,[34] anchored neighbourhood regression,[35] and others.[36][37] Additionally, as mentioned in the paper, the trained CNN produces high quality output, yet far from converges on the global minimum (minimum error between input and output). The authors therefore conjecture that further improvements could be made in the area of CNNs with the use of a larger network and dataset. Their work appears to be the latest significant research in the area of super-resolution, particularly using neural networks.

Neural network projects do also exist that directly map X Y co-ordinates to RGB values, instead of low resolution to high resolution pixel information, however these tend to be toy projects or have unrelated applications.

2.3.2 **Other Neural Network Projects**

ConvNetJS, for example, is a JavaScript neural network library developed by Standford PhD student Andrej Karpathy. A browser-based demo of the ConvNetJS library, entitled "Neural Network 'Paints' an Image",[38] shows a image of kitten being mapped as an X Y – RGB relationship. The result is a representation of the image that is very rough or lossy. The demo page mentions "It's a bit like compression, since the image information is encoded in the weights of the network, but almost certainly not of practical kind." No discussion is given to the potential for image scaling with greater network precision.

A website called Picbreeder is another example of neural network X Y – RGB mapping, but instead of error backpropagation (as is typical of neural network training) an interactive genetic algorithm is used. This requires visitors of the website to rank abstract neural network-generated images based on aesthetic preference. The objective is not to encode pre-existing images but instead to generate new ones through interactive evolutionary computation, with aesthetic appeal as the fitness criteria. This continues from earlier work conducted by Karl Sims in the 1990s.[39] The resulting images are relatively abstract and significantly lack detail, but the developers do briefly mention that "pictures evolved in Picbreeder have infinite resolution (because they are stored as mathematical objects)."[40]

2.4 Summary

A variety of state-of-the-art image scaling techniques have been established in this chapter (S-splines, fractals, and CNNs). Each approaches the problem of photographic image enlargement in a different way while relating to SFGs, either through the use of neural networks, the achievement of resolution independence, or establishment of curve fitting coefficients. These algorithms will act as critical points of comparison in the following chapter (Chapter 3), which introduces details of SFG conversion and establishes this technique's relative advantages and limitations. Visual comparisons will be presented, demonstrating the quality and potential of SFGs, while leading to a discussion of the parts of the SFG conversion process that require further development.

Introduction to Scalable Function Graphic Conversion 3.0

SFGs are an experimental approach to high-quality image scaling, currently in their prototype stage of development. This chapter will provide an overview of the mechanisms of SFGs and the PhotoFunction conversion software, going into further detail in areas that help to explain SFG advantages, limitations and distinctiveness, as compared to previously discussed algorithms. This will act to frame a discussion of specific implementation improvements to be made PhotoFunction software and the SFG format, which will take place in chapter 4.

3.1 Overview

3.1.1 **The Conversion Process**

SFGs store image data as the connection strengths (or weights) of a neural network, otherwise described as coefficients of a continuous, bivariate, vector-valued mathematical function. This function is bivariate in that it takes two inputs - a single pixel's X and Y co-ordinates; and vector-valued in that it gives multiple outputs - the pixel's red, green and blue (RGB) colour channels. The function thereby describes a continuous relationship between pixel co-ordinates and corresponding colour channels.[4] Crucially the variables bear no reference to a specific resolution, in contrast to the spline approach discussed in Chapter 2.

$$V(x,y) = \langle\ R(x,y),\ G(x,y),\ B(x,y)\ \rangle$$

Mapping this abstract relationship as a continuous function is a regression problem, requiring a time-consuming gradient descent algorithm. During this process errors are backpropagated (fine-tuning the mathematical function's constants such that the error value between output RGB values and expected RGB values decreases to within an acceptable range). After this learning process is complete, the function can be fed X and Y coordinates of any pixel in the image, and return the correct RGB values. The image can now be stored in a resolution independent manner, encoded as the constants of the mathematical function. The function's structure is that of a fully connected, feed-forward, artificial neural network.[4]

3.1.2 **Extra-pixel Generalisation**

Fig. 5a, original sunset41

Fig. 5b, extrapolated sunset[42]

As a process of neural network generalisation (or educated guessing), feeding the SFG X and Y co-ordinates that were not defined in the original image results in high quality pixel estimations. This capability of generalisation can be illustrated by observing a function graphic (a general case of SFGs developed by the author) that has extrapolated outside the borders of the original bitmap, synthesising extra-pixel values (see

Fig.5b). Features and colours that existed inside the bitmap are transformed into the extended image. An 'Extrapolatable Function Graphic' such as this requires the entire image to have been mapped as a single, continuous mathematical function.[42] However the larger and more complex the original bitmap image, the increasingly large the mathematical function required to adequately model it. In this case it can be seen that the original bitmap (Fig. 5a) possesses features that the function graphic (Fig. 5b) has not been able to accurately represent.

However, for the purpose of interpolating (rendering sub-pixel values) a single mathematical function is not required to model the entire bitmap. Instead the image can be divided into small tiles (10 by 10 or 20 by 20 pixels), with a single function required only to accurately model a single bitmap tile. This significantly reduces the necessary size of each function, and necessary time for conversion.[43]

3.1.3 **High-quality Sub-pixel Generalisation**

Essentially, due to tiling, a bitmap can be completely modeled in all of its detail as an SFG (a collection of neighboring continuous functions), which can then be rendered at a much higher resolution while maintaining smooth gradients, sharp lines and details. Sub-pixel values are synthesised by generalising about local pixel patterns, resulting in aesthetic quality that rivals or even surpasses fractal image enlargement – the industry standard for photographic resolution independence. Fig. 6a highlights a 20 by 20 pixel tile that has been scaled up in figures 6b – e using a nearest neighbor, bicubic, SFG, and fractal algorithm (Perfect Resize software[21]), respectively.

Fig. 6a, Identified tile, original sunset[41]

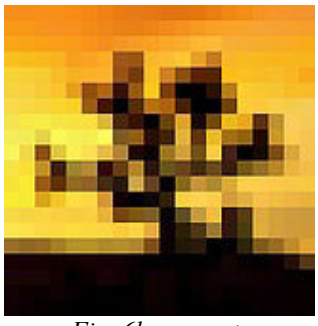


Fig. 6b, nearest neighbour[43]

Fig. 6c, bicubic[43]

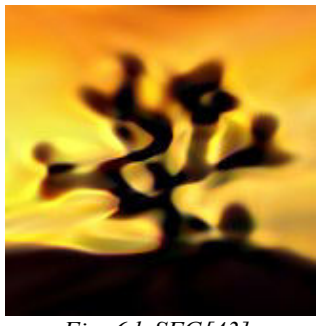


Fig. 6d, SFG[43]

Fig. 6e, Fractal[43]

3.2 Relative Advantages & Limitations

3.2.1 **SFGs vs. Fractals: File Format Redundancy**

The SFG format is capable of accurately describing a bitmap's pixel-level details, in a way that results in high-quality generalisation about sub-pixel information. However the process of converting a bitmap to an SFG representation can take hours or days, even for small thumbnails.[4] This is compared to the seconds it takes to generate a fractal representation. Also, while both fractal and SFGs achieve resolution independence, significantly more information appears to be required to encode images as an abstract X Y – RGB relationship than in terms of self-similar transformations. Fractals essentially act as a form of compression, while SFGs require as much as 30 times more information than the original bitmap.[4]

Nevertheless redundancy in the SFG file format is likely a contributing factor to this problem. Chapter 4 (Improvements to PhotoFunction Software) will discuss means of reducing redundancy by adapting neural network sizes to the complexity of specific images, instead of relying on a fixed network size for all image tiles.

The conversion-speed disparity between fractals and SFGs is also a large problem, with spline interpolation similarly capable of producing results in a matter of seconds as opposed to SFGs. Significant differences exist in the way that splines and SFGs model pixel data, and these highlight the main reasons for PhotoFunction's slow conversion speed (discussed in the following three sections).

3.2.2 SFGs vs. Splines: A Pixel as a Domain

As mentioned by Siu and Hung 2012,[44] spline interpolation techniques tend to treat pixels as points that belong to a precise X Y co-ordinate. This is only the case if the photographic image is highly aliased (where pixels reflect sparsely sampled information). Arguably, it is more accurate (but processing intensive) to treat a photographic pixel as a domain (a region of possible co-ordinates), each with an unknown range (set of possible colours) except for an average range across the domain as a whole.

Essentially, as illustrated in Fig. 7, one grey pixel at the co-ordinate (0,0) does not necessarily indicate that a grey object existed there as photographic subject matter. Instead this pixel value suggests that for any set of objects that existed within the pixel domain (between the co-ordinates {0,0}, {1,0}, {0,1} and {1,1}) the only known value is the average colour, which is grey. Some objects may have been white and some may have been black (but all averaged to from a single pixel value).

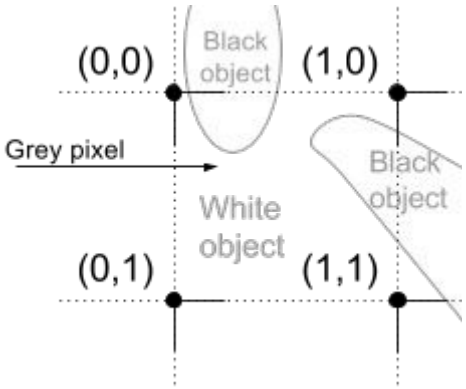


Fig. 7, Pixel domain

Interpreting the data such that pixels represent domains, as opposed to precise points, adds complexity and time to the modeling process, but allows underlying colour relationships to emerge more clearly. This is achieved in PhotoFunction by a form of batch gradient descent that has been adapted for SFG conversion. During conversion, a batch of sub-pixel co-ordinates are fed as inputs to SFG neural networks at regular intervals within the domain of a single pixel (5 by 5, 25 sub-pixel co-ordinates, see Fig. 8). For each sub-pixel co-ordinate the input values reaching each neuron in the network are recorded, in addition to the output values for the network as a whole. After 25 evenly spaced sub-pixel co-ordinates have been fed through the network, an average output value for the network as a whole is calculated, in addition to an average input value for each neuron. These averages are then used to backpropagate error through the network, updating weights accordingly.

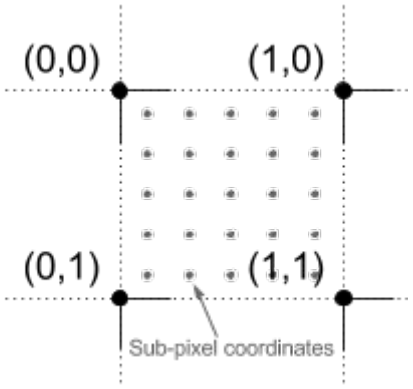


Fig. 8, Sub-pixel

This improvised technique is slower due to multiple sub-pixel co-ordinates needing to be fed as inputs, as opposed to a single pixel co-ordinate if pixels were to be treated as normal training data points. However it ensures that the network treats pixels appropriately, as an average range over a domain. Additionally, this batch approach provides some degree of resistance to Runge's Phenomenon.

3.2.3 SFGs vs. Splines: Runge's Phenomenon & the Variable n

Runge's phenomenon describes a circumstance in which, for high values of ***n*** (number of data points to be mapped by a single continuous function), interpolated values are inclined to oscillate significantly between data points, resulting in visual artifacts.[45] For SFGs, modeling the pixel as a domain means that sub-pixel

co-ordinates feature as artificial data points during the training process. This reduces the spaces within which Runge oscillations are more inclined to emerge. Conversely, spline (or piecewise polynomial) interpolation overcomes Runge's phenomenon by reducing ***n*** to as few as 2 for each spline (each spline together constitutes a larger piecewise defined function).[46]

The benefit of modeling high values of ***n*** as a single continuous function (as with SFGs) is that it allows for more representative generalisation (based on a larger amount of data). On the other hand, this directly contributes to the slowness of SFG conversion due to a high number of training data points to process.

Another significant aspect of SFG conversion that helps to overcome Runge's phenomenon is the application of curriculum learning. This further contributes to the slowness of SFG conversion. As discussed in the following section.

3.2.4 SFGs vs. Splines: Runge's Phenomenon & Curriculum Learning

There are various approaches to avoiding Runge's Phenomenon that still allow for large numbers of data points to be mapped by a single continuous function. These are referred to as regularisation methods, where restrictions are imposed on the degree of complexity allowed for the fitted or trained function.[47] Several such processes have been explored to help avoid Runge's phenomenon in SFGs (such as the early stopping of training, a reduced number of neurons, and altered error functions) but in each instance this severely limited the amount of photographic detail or sharp edges that could be modeled.

Instead, the main approach through which Runge's Phenomenon is avoided is by applying curriculum learning as a continuation method. Continuation methods are optimisation strategies where the error function has been smoothed in an attempt to better frame the problem, and curriculum learning is a term used to describe a staged process where simplified training data is presented to a neural network first before moving on to the proper training data.[48]

Effectively, SFGs are first trained on an image that has been scaled up using bicubic interpolation. This produces smoothly inflated training data, filling in the space between original data points with non-oscillating values. Training on this data first ensures that the network avoids Runge's phenomenon. The next stage of the learning process requires that the smooth training data is switched for the original training data. At this final stage in training it is essential that pixels are treated not as individual points but as domains (as discussed previously). This allows the image to be dramatically sharpened, significantly surpassing the visual quality of bicubic interpolation.

At this stage the neural network is not as resistant to Runge's phenomenon as in the first stage of training, but this is necessary. A degree of Runge's phenomenon appears to provide the illusion of greater detail in the rendered image. Fig. 9 compares a scaled SFG to bicubic interpolation. One of the various SFG 'details' that can be referred to as an artifact of Runge's phenomenon has been highlighted by a red square.

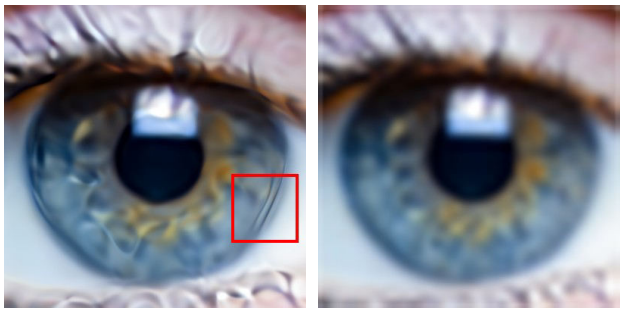


Fig. 9, Runge's Phenomenon

Runge's phenomenon can be described as the over complication of an interpolant, but the idea of SFGs is that they should indeed attempt to synthesise complex sub-pixel information (so long as it is based on a learnt pattern from surrounding pixel data). By ensuring that the image has first been smoothly modeled in its entirety, later instances of Runge's phenomenon (during the sharpening phase) are intended to be based on or influenced by learnt image content.

Constructing two learning phases reduces the speed of SFG conversion, but is necessary to minimise the emergence of unwanted artifacts. Therefore, the interpretation of pixels as domains, and the high number of

data points modeled by a single continuous function, in addition to this multi-stage learning process, all contribute to the slow conversion process of SFGs.

However, the most significant draw backs of SFGs relate to the way in which the learning or conversion process takes place compared with CNN image scalers. These issues are discussed in the final two sections of this chapter, before a summary that consolidates the information that has been presented so far.

3.2.5 SFGs vs. CNNs: Training on Demand

Unlike SFGs, a significant benefit of CNNs are that they are only required to be trained once (using low resolution and high resolution image pairs as training data) and can then be applied to upscale any image given as input.[7] This is contrast to the SFG approach, in which training is required to take place on demand for each image as a conversion process (the trained neural networks essentially become the converted image representation). Measures for reducing demand on the SFG conversion process (by utilising pre-trained neural networks) will be discussed in Chapter 4.

Nevertheless, there are also fundamental differences in the nature of the input/output relationships that SFGs and super-resolution CNNs attempt to model. The connection between low resolution and high resolution data is intuitively less abstract than the relationship between arbitrary X Y co-ordinates and RGB values. Most importantly unlike super-resolution CNNs, where only a general data fit is necessary (or even enforced through a process of regularisation),[7] SFGs are required to model training data with very high precision to actually represent the detail of the photographic image. As is typical of neural network training, progress significantly slows as the global minimum (minimum error) is approached.

3.2.6 SFGs vs. CNNs: Approaching the Global Minimum

The simplest reason for this decline in the speed of progress is that the size of network updates are based on the size of errors encountered. Error values decrease as the global minimum is approached, so network updates reduce and progress slows proportionately.

Further, 100% of the training data contributes to learning initially, as the network encounters a high error value for each data point. Towards the end of the training process only a fraction of the training data present high error values, and so opportunities for learning occur less frequently. This problem is referred to as data sparsity or “the curse of dimensionality.”[49] The current SFG implementation reduces this problem through importance sampling, training high error networks more frequently than lower error networks over the course of the conversion process. Further measures that might reduce the likelihood of error disparity between data points are discussed in the following chapter (Improvements to PhotoFunction Software).

Nevertheless, this problem of slowed error minimisation is not such an issue for super-resolution CNNs, where there is an inherent limit to how far error can even be minimised (given that the information needed for 100% high-resolution ground truth accuracy can almost never be contained in low resolution input images).

3.2.7 Aesthetic Comparison

As described in Chao Dong et al's 2014 paper, deep super-resolution CNNs represent the state-of-the-art in super-resolution. An open source implementation (based on the research of Chao Dong et al) is available on GitHub,[50] with a live version hosted at a website called Waifu2x.[51] With ease of access to a high quality CNN, results from the Waifu2x can be used as a benchmark against which the current and further developed SFG implementation can be compared, along with leading fractal and spline-based commercial algorithms.

As can been seen from Figures 10a-f, SFGs produce compelling results that are on par with, or variably appear to exceed, the visual quality of these state-of-the-art approaches (see Chapter 5, Evaluating Results, for mean squared error). SFGs represent a novel technique for high quality image scaling which, given further development, has potential for significant improvements in conversion speed and storage efficiency, as will be outlined in the following chapter (Improvements to PhotoFunction Software).

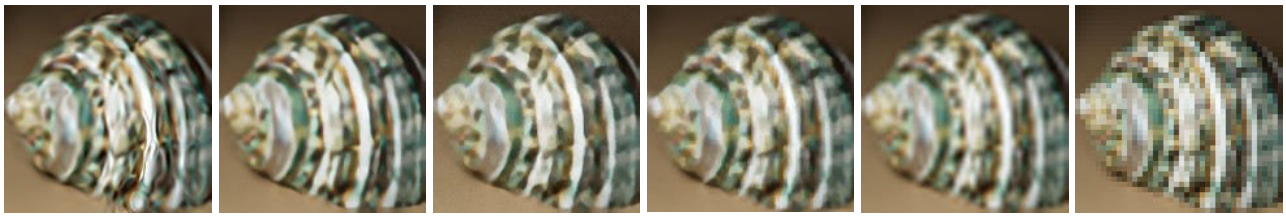


Fig. 10a, SFG *Fig. 10b, Waifu2x* *Fig. 10c, S-s max* *Fig. 10d, Fractal* *Fig. 10e, Bicubic* *Fig. 10f, Nrst N.*

3.3 Summary

This chapter has provided an introduction to the mechanisms of SFGs, and established the conceptual strengths and weaknesses of encoding image information as a network of artificial neural connections (determined through sub-pixel-batch gradient descent and curriculum learning).

Particular problems have been identified relating to SFG file redundancy (compared to the compression capabilities of fractal images) and the time consuming conversion process, which takes place on demand (as opposed to super-resolution CNNs which are trained in advance). The following chapter will outline how further research and development in the areas of machine learning and high performance computing can improve these issues. Chapter 4 will also explain how the addition of simple image filters can help to further improve image quality (filters that are already employed by S-spline Max and fractal enlargement software).

Improvements to PhotoFunction Software 4.0

4.1 Machine Learning

4.1.1 **Tile Size & the Global Minimum**

As discussed in the previous chapter, the high number of data points (***n***) modeled by SFGs in comparison to spline techniques directly contributes to the slowness of SFG conversion. It has also been discussed that segmenting an image into tiles (as opposed to modeling an image with as a single continuous function) is what allows SFGs to accurately model a photographic image within an achievable time-frame. But an aspect that has not been adequately explored in the development of PhotoFunction is the degree to which aesthetic quality is effected by further reducing tile sizes and therefore the value of **n** (below 10 by 10 pixels). This would reduce the number of data points to be processed and could act as a means of significantly improving conversion speed and file size, utilising smaller neural networks that model smaller tiles.

The use of smaller tiles would also decrease the likelihood of error disparity between data points, by reducing the potential for relatively high error data points to be visited less frequently. This problem has been discussed in the previous chapter as contributing to decreasing training progress as the global minimum is approached.

On the other hand, the basis of SFG image scaling is that the neural networks have a broad enough experience of data to be able to generalise (based on that data) and render plausible sub-pixel values. The trade-off relating tile size and generalisation quality requires further investigation. It may be the case that similar quality images can be produced through the use of much smaller tiles (and neural networks), resulting in significant speed and file storage improvements.

The consideration of smaller tile and neural network sizes underpins the other machine learning-based improvements that are intended to be made to PhotoFunction software. These are:

- The establishment of a database of small (efficiently stored) pre-trained neural networks, reducing the amount of on demand training during the SFG conversion process. This was identified in the previous chapter as a significant limitation of SFGs in comparison to CNNs, which are trained in advance.[7] The establishment of a pre-trained database of neural networks will be discussed in the following two sections, Initialisation & Pre-training and Data Pre-processing & Feature Space reduction.
- The development of a fine-tuning algorithm that facilitates neural growth, ensuring that network sizes are based on the complexity of the respective image tile (reducing redundancy in the SFG file format, and adapting small pre-trained networks to increase capacity for greater modeling complexity). This will be discussed in the section, Neuron Redundancy & Adaptive Growth.

4.1.2 **Initialisation & Pre-training**

Pre-training is the process of training a neural network in a way that encourages initial features to be learnt in a data set that are expected to provide a useful starting point for the main training process (to avoid local minima[52] and other problems[53]). The current SFG software initialises weights using a heuristic approach, where random variance in the intialisation of each neuron's output weights is capped relating to the number of that neuron's input weights. This prevents every neuron from initially having drastically different output values (convergence on a good solution can take longer or is less likely to take place if neuron outputs begin with significant disparity between them[54]). A further improved approach to initialisation would be to pre-train networks on a set of features that are known to be beneficial starting points to the general learning process.

However, given that the training data (photographic image tiles) can contain a wide range of characteristics in terms of colour, geometry and texture (with some image tiles contrasting with others) the existence of a general set of features on which all neural networks should be pre-trained is not apparent. This has led to the decision to establish a database of pre-trained networks, where each will have been pre-trained on a specific set of features and can be retrieved from the database by closest match to image characteristics.

There is, however, a potentially huge number of image patterns on which neural networks could be pre-trained. Determining which are the most useful and representative would be problematic. Instead, this potentially huge space of features can be reduced in a variety of ways, such as if smaller image tiles were to be used (thereby increasing the representativeness of sampled pre-training data). To even further improve the broad applicability of pre-training data, a pre-processing trick can be performed, as discussed in the following section.

4.1.3 **Data Pre-processing & Feature Space Reduction**

Pre-processing filters are applied to training data before learning takes place in order to present it in ways that are more compatible with neural network learning.[55] A simple example is normalising image data from the range 0 to 255 to the range -1 to 1 (more in line with neuron activation functions). This is a pre-processing step performed by the SFG conversion software (both for RGB values and for X Y coordinate values). Typically data pre-processing increases the chances of good quality convergence on the global minimum (minimum error).[56]

Currently this pre-processing filter is applied to RGB training data irrespective of specific image tile characteristics. However image tiles (particularly smaller ones) often only occupy a fraction of the possible 0 – 255 range. To better accommodate this variety, a simple analysis could be performed to determine the RGB ranges for each image tile, allowing for tile specific data normalisation. This would ensure that each image tile ends up occupying the full -1 to 1 range, as opposed to just a fraction of it. Centering the data at zero in this way is useful to maximise the initial size of neuron activation function derivatives, which directly correlates with learning speed.[57] Also, performing this pre-processing step independently for each colour channel will result in a relatively monochromatic appearance, as the differences between colour channels are minimised (particularly for smaller tiles, which initially occupy a smaller colour depth range). See Fig. 11b for an example of the described colour channel independent pre-processing filter, though still in the range 0 - 255 (generated using author's code).

Fig. 11a, Lenna, unprocessed

Fig. 11b, Lenna, 4 by 4 preprocessed tiles

This pre-processing step could be reversed in final renderings of trained SFGs to obtain full colour scaled images. Most importantly, with tiles that are monochromatic, the feature space for possible pre-training data can be reduced from $\mathbf{256}^{(\boldsymbol{3n})}$ to $\mathbf{256}^{(\boldsymbol{n})}$, where **256** is the pixel colour depth, ***n*** is the number of pixels and the coefficient is the number of colour channels.

This is still to much data to be represented by a pre-trained database. To reduce this feature space further, it will be necessary to decrease the colour depth represented in training data (from ***256***), in addition to the

image tile resolution (***n***). Retrieved neural networks can then be fine-tuned to higher precision or resolutions as necessary. This will require a facility for adding complexity to networks, increasing their size and modeling capacity during fine-tuning (as addressed in the following section).

4.1.4 **Neuron Redundancy & Adaptive Growth**

Image tiles that are modeled by SFG neural networks vary greatly in terms of complexity, from almost complete pixel uniformity to significant differentiation. Instead of relying on a fixed network size (as is currently the case) it would be better to ensure that a minimal network is used for the given pixel data. It is known that smaller networks with fewer parameters can be trained significantly faster, generally requiring fewer training epochs to minimise errors,[58] in addition to requiring a reduced number of parameter updates for each epoch. Smaller networks in relation to training data also tend to have better generalisation capabilities.[59]

On the other hand networks need to be large enough to model sufficient complexity in the given data. Trial and error has been the approach used to determine a generally suitable network size for the current SFG conversion software (7 layers, 30 neurons each; 5732 parameters). This size is applied to every SFG network regardless of the characteristics of the image tile on which the network is trained. As an improvement, one of two approaches could be used to strike an optimal network size dynamically, ensuring that each is only as large as required.

Pruning is one such approach, in which an intentionally over-sized neural network gradually has weights removed until overall performance is adversely affected. Weights can be selected for removal based on their estimated importance, for which various heuristic techniques exist (such as optimal brain damage[60] and optimal brain surgeon[61]). However, an alternative and simpler approach is to begin with a network that has a minimal number of neurons. Extra neurons are added to the network during the learning process whenever improvement is no longer taking place with the provided number of neurons. Training is deemed complete once overall error has been minimised below a required threshold. The network then contains a minimal number of neurons required to fit the training data.[62]

There appears to be far more literature on the topic of pruning. This is likely due to the advantages of larger networks in that they tend to be more flexible and less sensitive to the choice of initial weights.[63] However neural growth is the preferable option in this instance, as it facilitates efficient database storage (small initial networks) and subsequent adaptability to achieve the necessary modeling capacity and precision.

By normalising training data appropriately for each image tile, utilising efficiently stored pre-trained neural networks, and reducing neuron redundancy through tile specific network growth, it is anticipated that improvements can be made in terms of the efficiency of SFG conversion. Most importantly, segmenting an image into smaller tiles has the potential to dramatically speed up the learning algorithm in addition to reducing SFG file size (by reducing the necessary size of neural networks). Exploring the effects this will have on the aesthetic quality of SFG images will determine the extent to which tile sizes can be reduced.

The first section of this chapter has focused specifically on improvements that can be made to machine learning aspects of PhotoFunction software, such as pre-processing, intialisation and training procedures. The next section will focus on high performance computing concepts, before a brief discussion of aesthetic enhancing image filters in the final section of this chapter.

4.2 High Performance Computing

In terms of general code structure, the current SFG conversion software is not very well optimised for runtime efficiency (see Appendix-2). Four general areas that can be improved upon are identified in the following sections; Concurrency, Memory Caching, Precomputation and Compiler Optimisation.

4.2.1 **Concurrency**

As described by Amdahl's Law,[64] the theoretical speed increase that is made possible through parallel computing is determined by the proportion of processing time that is spent executing a parallelisable task. As an example, PhotoZoom Pro (the software that hosts S-spline Max) is highly parallelised, and has been praised for featuring scalable run-time performance proportional to the number of available processor cores. [65]

The process of SFG conversion is also highly parallelisable, with image tiles capable of being processed simultaneously on separate cores over the entire conversion process. Yet despite this, the current conversion process is completely sequential and handled by a single thread. Therefore, a relatively simple means of increasing speed is to determine the number of cores available at runtime, and to distribute a group of image tiles to separate threads that can run independently on each core.

However, neighbouring neural networks are required to share some information during the second stage of conversion (ensuring seamless overlap). Therefore it will need to be determined whether or not memory inconsistency between separate threads (that handle neighbouring networks) becomes an issue in this project. In such case volatile access modifiers may need to be employed on the weights of fringe located networks (located a the fringes of an area handled by a specific thread). This would have the affect of avoiding possible discontinuities between the output of fringe networks, but will negatively affect runtime speed due to the time it takes to access main memory as opposed to cached memory.[66]

4.2.2 **Memory Caching**

Cached memory is significantly faster to access than main memory. Caches store information based on the likelihood that it will need to be accessed by a program in the future. This can be be estimated based on the principle of spatial locality, suggesting an increased likelihood that a program will need to access a data element that exists at a close storage location to another data element that has just been accessed. For this reason data elements that are stored in close proximity are often fetched and cached as a block of contiguous memory, or cache line.[67]

The expectation of spatial locality could be better utilised by the current SFG software. Image tiles are currently retrieved in random order, with the neural network for each tile only trained on one data point before another tile/network is randomly selected. A better approach would be to select tiles sequentially, and to only move on to the next tile once the previous tile has been sufficiently mapped. This would increase the frequency with which closely stored data elements are accessed, taking advantage of contiguous memory caching.

Unfortunately this approach cannot be applied to the second phase of SFG training, at which point neighbouring neural networks are required to share information during the training process. At this point all networks must make progress together.

4.2.3 **Precomputation**

There are also a significant number of calculations that are preformed during SFG conversion that could benefit from being precomputed and stored as a lookup table. This would reduce the overhead of having to perform identical calculation repeatedly (a technique that can provide significant run-time performance gains[68]).

The most obvious place in which this technique could be implemented is in calculating the output of certain neurons. Two kinds of neuron activation functions are used in SFG neural networks. These are sine (for the majority of neurons) and arctangent (for the last hidden layer of neurons). While arctangent is not periodic and has a domain that spans all real numbers, sine has a period of 2π, beyond which output values display a repeating pattern. A lookup table could therefore be efficiently precomputed for the sine function, with the need to only store outputs for the domain 0 to 2π.

4.2.4 **Compiler Optimisation**

In addition to increasing runtime speed through the use of precomputed values, the SFG source code in general could benefit from significant refactoring (for the purpose of readability as well as optimisation). Aside from containing redundant variables that are no longer used, the majority of methods are currently extremely long (see Appendix-2). In order for Java's just-in-time (JIT) compiler to effectively perform code optimisation such as inlining (which avoids the overhead of method invocation), it is necessary for methods to be kept small and compact.[69] It is therefore anticipated that speed increases could be gained by tidying code and reducing the size of methods.

4.3 Image Filters

Finally, having achieved an updated PhotoFunction implementation that has been optimised for efficiency through the application of both high performance computing and machine learning concepts, aesthetic enhancing image filters will also be implemented to further improve perceived image quality. These filters are known as noise and unsharp, both of which are employed as features of PhotoZoom Pro (S-spline Max software) and Perfect Resize (fractal image scaling software).[70]

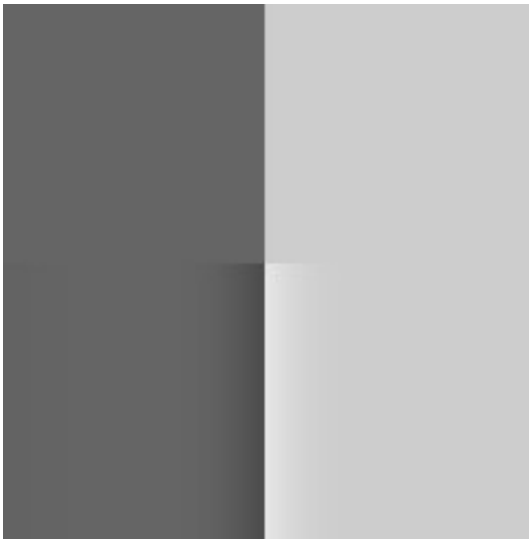


Fig. 12, Unsharp example

Unsharp increases contrast at the edge of objects, convincing the eye of greater clarity. Fig. 12 is an example image that has been created using the image manipulation software GIMP. The top half of the image represents the original and the bottom half has had an unsharp filter applied. Unsharp subtracts from pixels the positive or negative difference between a blurred version of the image and the original version of the image.[71]

Additionally, a small amount of random noise (simulated film grain) will be explored as a means of suggesting further photographic detail or granularity for scaled SFGs. As described by Perfect Resize documentation, “adding a modest amount of Film Grain can make your image appear sharper visually.”[70]

Nevertheless, the topic of aesthetics is a subjective one. This will need to be taken into account when evaluating the results of this project. The following chapter will discuss how the research results will be evaluated, highlighting the need for human participants and the problem of relying purely on an algorithmic error calculation.

Additionally, the choice of programming language will be justified, and this will lead to the conclusion and summary of this document.

Further Considerations
5.0

5.1 Evaluating Results

In order to gauge the prediction error of scaled images, mean squared error (MSE) will be used to compare against corresponding ground truth, high-resolution images. The Kodak lossless true colour image suite[72] will be used due to their popularity as standard test images in general image processing research. This will provide an accurate measure of how closely scaled SFG images have managed to reconstruct lost information.

However it would be problematic to rely solely on this algorithmic approach to graphic evaluation. A preliminary investigation reveals that industry standard image scaling software, such as Perfect Resize, produce scaled images that are not as accurate as simpler algorithms (such as bicubic interpolation) at reconstructing an original high definition image. Yet the results of Perfect Resize are visually sharper and appear to have significantly more clarity than bicubic interpolation.

Mean Squared Error (2x scaled shell image compared to high-resolution ground truth)	
Algorithm	**MSE**
Waifu2x (SRCNN)	373
S-spline Max	456
Bicubic spline	473
Scalable Function Graphic	523
Nearest Neighbour	610
Perfect Resize (Fractal)	778

The table to the right shows the results of low resolution 24-bit images that have been scaled up by a factor of two and compared to the corresponding high-resolution ground truth images. As can be seen, according to MSE (MSE values correlate positively with the degree of error), Perfect Resize and SFGs are roughly as poor as nearest neighbour interpolation (the simplest interpolation algorithm) at reconstructing the original high-definition image.

This can be understood in that the fractal approach to image scaling is fundamentally different to spline interpolation and super-resolution CNNs (for which the direct basis of training is high-resolution reconstruction).[7] The aim of fractal image scaling is to synthesise visually plausible sub-pixel information based on self-similar features within the image as a whole (whether or not such information existed at an exact location in the original image). This is discussed in chapter 2, referring to the term resolution enhancement over resolution enlargement.[24] Similarly, SFGs aim to synthesis sub-pixel information by utilising patterns that have been identified in other parts of the image. These relationships need not exactly reflect the nature of original sub-pixel information, but tend to convince the eye of perceiving high clarity visual information.

Essentially, human perception is a success criteria that is difficult to accurately define algorithmically. For this reason a sample of members of the public (who will be asked to rank scaled images in terms of clarity and attractiveness) will be critical to the aesthetic evaluation for this project. This will provide a measure of the visual plausibility of sub-pixel information, and the degree to which scaled images posses apparent clarity and detail. MSE will nevertheless provide a valuable metric that can give further insight into the effects that updates to the SFG conversion software have had on scaled image quality, particularly regarding the reduction of tile sizes and the effect on generalisation ability.

Other than evaluating the aesthetic aspects of the updated SFG conversion software, efficiency improvements will need to be determined. This will involve establishing benchmarks for comparing the updated and previous versions of SFG conversion software, in terms of conversion speed and resulting SFG file sizes. Ideally these benchmarks will also be run against industry standard, proprietary algorithms. In terms of performance speed, this will depend on obtaining command line versions of commercial image

scaling algorithms, such that completion can be automatically detected and timed. This may not be possible in each case, but given the latency with which SFG conversion software currently achieves its results, significant improvement on the previous implementation will still certainly be quantifiable.

5.2 The Jave Language

Finally, Java is the programming language that will be used in this project. This is largely due to the fact that the current SFG implementation has been programmed in Java. Nevertheless, the language is platform independent, allowing for runtime optimisations to be performed that are platform specific.

Additionally the JAR (Java Archive) file type features optional compression and can act as a resource container, accessible from a Java program.[73] This will allow for pre-trained neural networks to be stored in an optionally compressed format, as a single Java friendly file. As the pre-trained networks are only required to be retrieved (not updated and stored), a fully functioning database is not necessary. The option to turn JAR compression off could also result in faster runtime access. This trade-off between JAR file compression and runtime speed will need to be investigated.

Conclusion
6.0

The intention of this document has been to set the context for a research project that aims to build on PhotoFunction software; a prototype system for high quality photographic image enlargement. Image upscaling is a non-trivial problem in the field of digital signal processing that requires the spatial resolution of information to be artificially inflated in a way that is plausible to the eye and/or accurate to a high-resolution ground truth image. The main challenge revolves around the fact that photographic images are not expected to represent smooth functions.[74] They are generally composed of discontinuous edges and granular textures. Predicting or synthesising these patterns realistically at the sub-pixel level requires the utilisation of context specific information acquired from surrounding data.

PhotoFunction and the SFG file format are an experimental means of addressing this challenge, synthesising new pixel data based on a continuous mathematical model of image patches (providing a resolution independent image format). There are many other high-quality image scaling techniques, the most pertinent of which have been discussed in this review; from fractals which efficiently encode self-similarities within an image, to CNNs which learn context specific transformations from low-resolution to high-resolution images (both introduced in chapter 2). There are also techniques that have not been discussed, such as multi-scale texture synthesis,[75] which facilitates infinite example-based texture zooming, though it appears to be unsuited to photographic image enlargement (see appendix).

By comparison to fractals, CNNs and splines, SFGs are currently limited by impractically slow conversion times and large file sizes. This is due to a variety of factors.

- As opposed to fractal images, SFGs establish resolution independence as an abstract X Y – RGB relationship that appears to require more storage information than the original bitmap (discussed in Chapter 3).
- Unlike CNNs, SFGs require live training specific to individual images and are required to accurately minimise the error function (Chapter 3).
- Also, unlike splines, an SFG models a large number of pixels continuously, necessitating a multi-stage learning process to avoid problems with Runge's phenomenon. Further, SFGs interpret pixels as domain averages, necessitating sub-pixel batch training which takes longer (Chapter 3).

Nevertheless, the aesthetic quality of SFGs rivals (or even surpasses) these industry standard techniques. To determine the extent to which SFGs can represent a more practical approach to high-quality image scaling further development to improve runtime and storage efficiency is necessary.

Chapter 3 and Chapter 4 have provided an overview and theoretical basis for the development intentions of this project, building on the content of previous chapters and referring to established machine learning and high performance computing concepts. As discussed in Chapter 4, a database of broadly applicable pre-trained neural networks (trained on a reduced feature space) will be developed. Retrieved networks will be adaptable to specific image tiles (through channel specific pre-processing filters and neural growth). This will act to provide a head start on the SFG conversion process (reducing the amount of live training required), which will be optimised for runtime efficiency by applying concepts such a concurrency and precomputation.

Specific questions that require further investigation have also be identified. These are:

- The relationships between image tile size and neural network generalisation capabilities. It is anticipated that smaller tiles will allow for faster conversion and smaller file sizes (as mentioned in Chapter 4), however it is not clear the extent to which tile sizes could be further reduced without negatively affecting network generalisation capability.
- The extent to which multi-threaded conversion results in thread memory inconsistency errors between fringe located neural networks.

- The extent to which JAR compression of pre-trained neural networks effects their retrieval speed, and therefore runtime performance.
- The extent to which aesthetic quality can be further improved by using image filters such as noise and unsharp.

Plans for evaluating this upcoming project have also been discussed in Chapter 5, involving speed and file size benchmarks, human feedback regarding aesthetics, and ground truth error calculations.

An expected time-line and risk analysis for this project can be found in the following pages.

6.1 Future Research

Future work (beyond the scope of this research) is likely to focus on the development of an integration based training algorithm that calculates mean neural network output over a specific domain of inputs (instead of approximating the average based on sub-pixel coordinate sampling, as discussed in Chapter 3, which is assumed to be slower and produce less accurate mathematical models). The challenge this presents is that the backpropagation of error through a network requires that a record is kept of neuron inputs that correspond to network outputs. It is not obvious how these combined values could be determined through the process of integration, which is the standard approach to calculating the mean output of a function over a given domain.

Other work might involve further exploring the effects of altering neuron activation functions, and establishing a specialised topology that can more efficiently encode image information than fully connected networks.

Work Plan
7.0

Expected Time-line

The estimated time required to complete the tasks below and reach each milestone has been illustrated using a colour-coded Gantt chart. Work on the project implementation will begin on 23rd May, and the intended end date for the project is the 22th August (3 months).

Milestone 1: Database of pre-trained neural networks (20th June)
The first work-plan milestone is the completion of a database of broadly applicable pre-trained networks (of minimal size) that can be efficiently stored and retrieved by closest match to image characteristics. Utilising pre-trained networks in the SFG conversion process prevents the need to train networks from scratch, increasing conversion speed.

Task	Deliverable
1.1 Explore the relationship between network size, and SFG conversion speed, storage efficiency and aesthetic quality. **(30th May)**	Data describing a continuous relationship between network size and conversion/storage efficiency, and a qualitative relationship between network size and aesthetics.
1.2 Determine, through experimentation and using the findings from task 1.1, an efficient structure for pre-trained neural networks, in terms of size/complexity and broadly applicable image features on which each network in the database should be trained. **(20th June)**	A program for automatically generating training data (image features) and for pre-training neural networks on such data.
	An altered SFG conversion implementation, featuring a database of pre-trained neural networks, and a facility for retrieving applicable networks from the database.

Milestone 2: Adapted learning algorithm, facilitating neural growth (11th July)
Having established an efficient database of broadly applicable pre-trained networks, the SFG backpropagation algorithm will need to be adapted to facilitate the addition of new neurons during the fine-tuning process. This is to allow a database of neural networks of minimal size, pre-trained on simple features, that can each be fine-tuned on demand to model more specific or complex image tiles.

Task	Deliverable
2.1 Adapt the backpropagation algorithm so that, as fine-tuning progress significantly slows, new neurons can be added to the network to extend modeling capabilities. **(11th July)**	A learning algorithm that can fine-tune and grow miniature pre-trained neural networks, to model more complex image features.

Milestone 3: Adapted implementation for concurrency (25th July)
Now that SFG conversion software has been adapted to provide a head start on the conversion process (in addition to reducing neuron redundancy), the implementation needs to be adapted to run concurrently on multiple cores, and refactored for efficiency, to increase run-time speed.

Task	Deliverable
3.1 Adapt SFG conversion implementation to make use of multi-core processors by allocating sections of a bitmap to separate threads, and refactor code for efficiency **(18th July)**	A multi-threaded SFG conversion implementation that counts the number of available cores, and divides a bitmap into sections that can be efficiently and robustly
3.2 Experiment with volatility to determine a suitable means of minimising edge-effects (which might occur at the fringes of image	

sections due to thread memory inconsistency) while maximising run-time speed. **(25th July)**	handled by separate threads.

Milestone 4: Addition of aesthetic-enhancing image filters (28th July)
Having taken measures to improve the efficiency of SFG conversion, a set of image filters can be applied as finishing touches to potentially enhance the perceived aesthetic quality of output images.

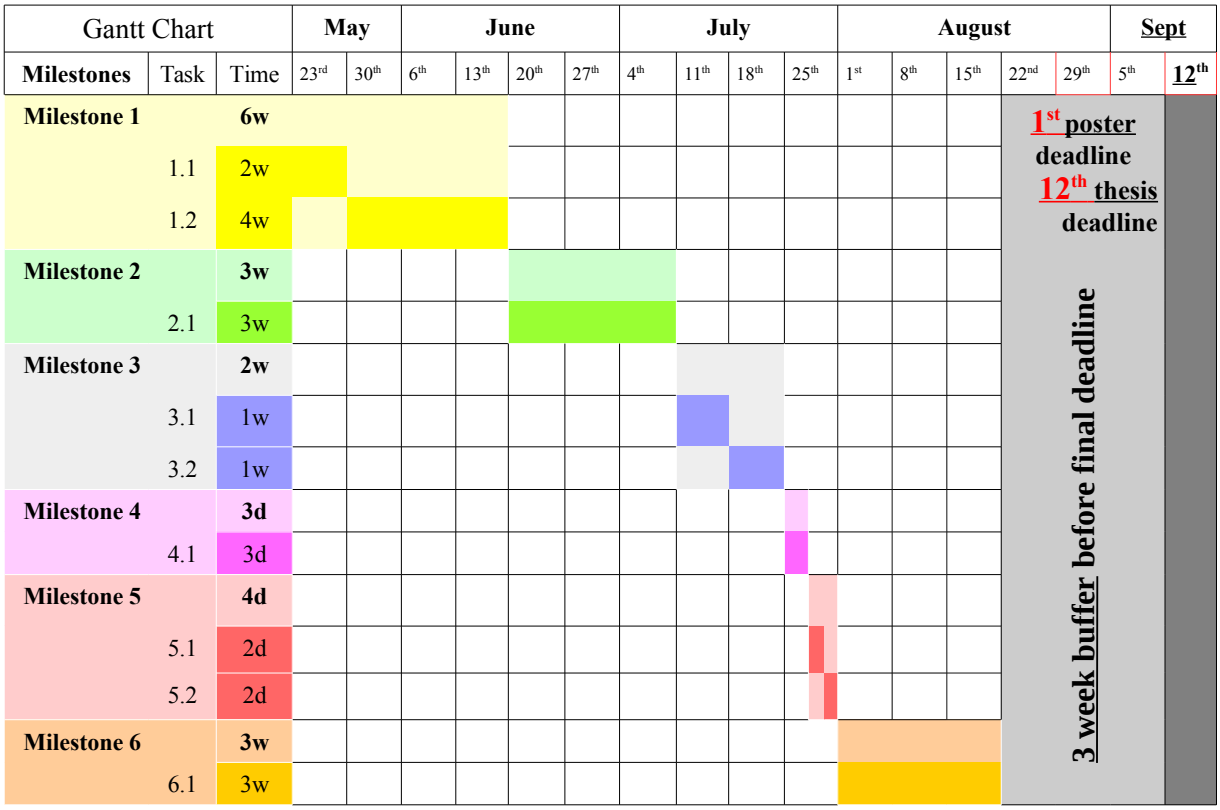
Task	Deliverable
4.1 Experiment with known image filters, such as unsharp and noise, to determine their effect on scaled SFG aesthetics. **(28th July)**	An updated SFG conversion implementation with optional image filters that can be applied as a finishing touch when rendering SFGs

Milestone 5: Final evaluation (1st August)

Task	Deliverable
5.1 Gather feedback from randomly sampled members of the public, assigning a rating (in terms of clarity and aesthetic quality) to scaled photographic images. **(29th July)**	Data that gauges the success of this project in relation to its aims, comparing the updated implementation with the previous version and with industry standards.
5.2 Gather data comparing the accuracy, speed and file storage efficiency of the updated and previous SFG conversion implementations, and to leading commercial image scaling software. **(1st August)**	

Milestone 6: Written report (22nd August)

Task	Deliverable
6.1 Write up and finish the report and accompanying poster. **(22nd August)**	An MSc Computer Science (Conversion) project report
	A poster, summarising main findings and deliverables



Risk Analysis & Contingency Plans

Unforeseen hurdles run the risk of setting the project back, leading to an incomplete implementation or evaluation by the time of the deadline. This section will describe possible hurdles, ordered by severity and likelihood, along with planned counter-measures. The following page also provides a risk register graph to illustrate likelihood and severity.

A. Acquiring a reliable participant sample

A sample of members of the public (who will be asked to rank scaled images in terms of clarity and attractiveness) will be essential to the aesthetic evaluation for this project. Ethical approval has already been granted for this investigation. However it may be a challenge to acquire a significant enough sample of participants to provide reliable results. A three week buffer has been built into the work-plan to help account for any additional time spent gathering enough participant feedback to provide a clear aesthetic ranking between images.

Severity – high Likelihood – medium

B. Demand Characteristics

The possibility of demand characteristics during aesthetic evaluation cannot be ruled out. Implicit communication between the experimenter and participants could sway preference decisions that are made. To reduce the possibility of this taking place, the experimenter will inform participants that they should not ask any further questions between the point that images are displayed to them and they cast their ranking decisions.

Severity – high Likelihood – medium

C. Data loss and hardware damage

Data loss due to theft or hardware failure is an ever present risk that needs to be actively mitigated against. An external storage device will be utilised in conjunction with an online data storage service, ensuring that both a local and cloud-based copy of work is regularly updated.

Severity – high Likelihood – low

D. Technical implementation difficulties

Unforeseen aspects of the implementation could present difficulties which exceed my current technical knowledge regarding neural network regression. As a starting point I have an in-depth understanding of each aspect of the current SFG conversion implementation. Challenges encountered in the scope of this project are therefore less likely to fall outside of my capabilities. This is taking into account the fact that textbooks and online resources are widely available and contain a wealth of knowledge on the subject of machine learning.

As a precautionary measure, I have been in communication with specialist staff in the areas of computer graphics and machine learning (via email and/or meetings) to discuss project details and setting up future meetings, (in addition to meetings with my supervisor, Dr. Chris Preist).

Severity – medium Likelihood – low

E. Establishing Speed Benchmarks

Establishing speed benchmarks against which to compare the updated SFG implementation with the previous implementation should be relatively easy, with complete access to the source code. However this will be more difficult or may not be possible for industry standard, proprietary algorithms. In an attempt to overcome this, command line versions of software will be sought that allow for algorithm completion to be automatically detected and timed. In any case, file size comparisons will certainly be possible, and significant speed improvements on the previous SFG implementation will still be detectable.

Severity – low Likelihood – high

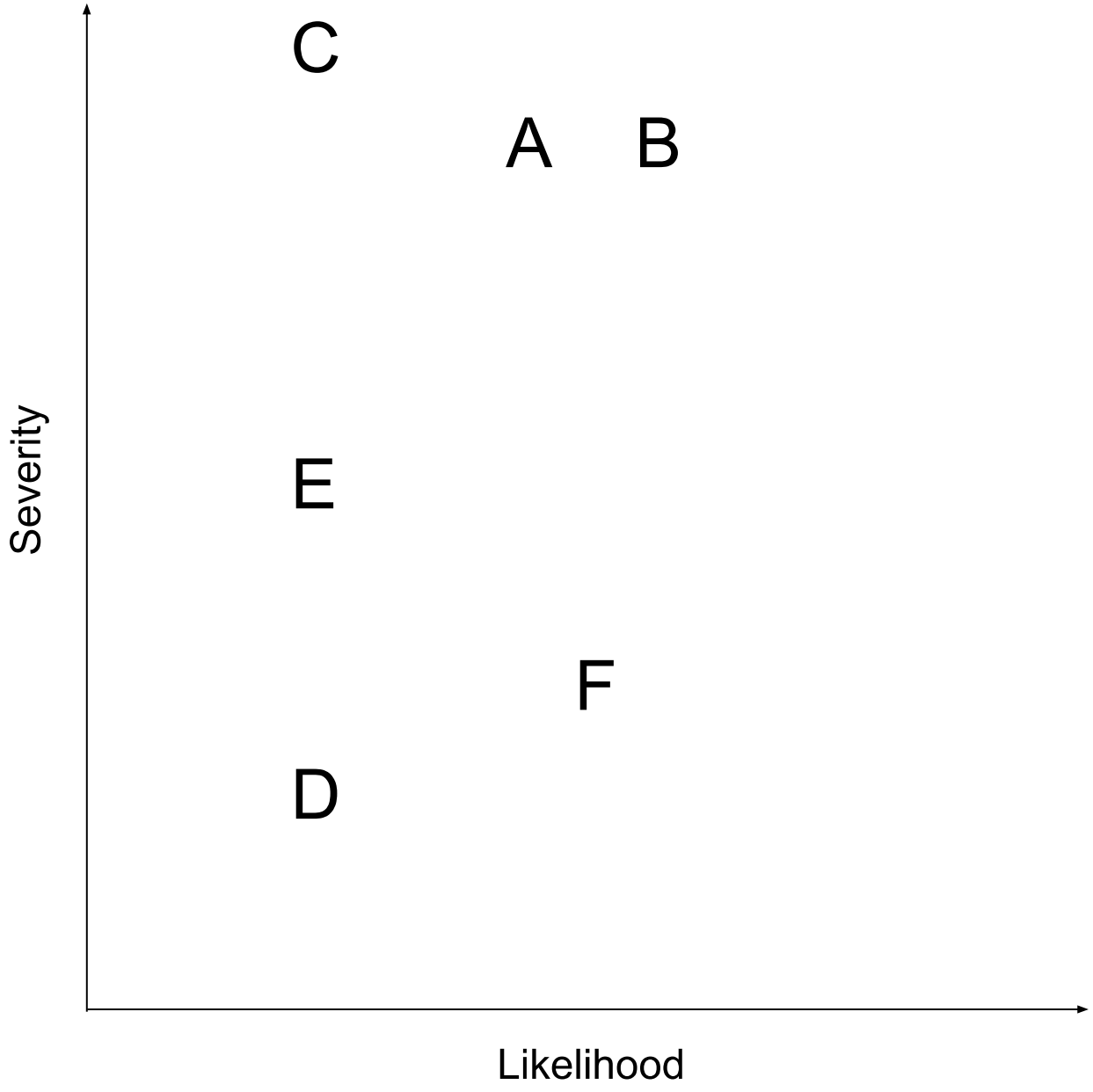
F. Unforeseen investigation outcomes

During the development process results that are not easy to anticipate could direct the project in slightly unanticipated directions, potentially to overcome unforeseen problems. This kind of uncertainty arises from the investigatory nature of this project (40% type II), and is not regarded as a significant problem.

The buffer that has been built into the work-plan will help absorb any additional time that might be spent implementing additional software features.

Severity – low Likelihood – medium

Risk Register Graph



Bibliography
8.0

1] V. Santhi, D. P. Acharjya, and M. Ezhilarasan, *Emerging technologies in intelligent applications for image and video processing*. IGI Global, p.51, 2016.

2] J. J. Parsons, *New Perspectives on Computer Concepts 2016, Introductory*. Cengage Learning, pp. 41-52, 2015.

3] J. J. Merelo, A. Rosa, J. M. Cadenas, A. Dourado, K. Madani, and J. Filipe, *Computational intelligence international joint conference, IJCCI 2014, Rome, Italy, October 22-24, 2014: revised selected papers*. Cham: Springer, p.308, 2016.

4] A. Lorimer "Scalable Function Graphics," AODLorimer. [Online]. Available at: http://www.aodlorimer.com/#!scalable-function-graphics/c1olm. [Accessed: 1-May-2016]

5] D. Leonard, "Scalable Function Graphics," *Titanium Alley Graphics*. [Online]. Available at: http://www.titaniumalleygraphics.com/#!sfg/c1nuq. [Accessed: 10-May-2016].

6] "SFG conversion, xBRZ, and similar weirdness [Archive] - Doom9's Forum," *SFG conversion, xBRZ, and similar weirdness [Archive] - Doom9's Forum*. [Online]. Available at: http://forum.doom9.org/archive/index.php/t-171861.html. [Accessed: 10-May-2016].

7] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution," *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, pp. 184–199, 2014.

8] A. Polyakov and V. Brusentsev, *Graphics programming with GDI & directX*. A-List, p.132, 2005.

9] E. Tapp, *Photoshop workflow setups: Eddie Tapp on digital photography*. Sebastopol, CA: O'Reilly Media, p.83, 2006.

10] W. K. Pratt, *Digital image processing: PIKS inside*. New York: Wiley, p.400, 2001.

11] "Enlarge images and increase photo resolution at high quality with S-Spline Max interpolation," *BenVista PhotoZoom Pro 6*. [Online]. Available at: http://www.benvista.com/photozoompro. [Accessed: 10-May-2016].

12] R. Eijkelhof, "Method for data processing." World Intellectural Property Organization Patent, WO 2000073993 A1, May 25, 2000

13] J. G. Williams, A. Kent, and C. M. Hall, *Encyclopedia of computer science and technology*. New York: M. Dekker, p.195, 1995.

14] S. T. Welstead, *Fractal and wavelet image compression techniques*. Bellingham, WA: SPIE Optical Engineering Press, p.65, 1999.

15] J. Sun, L. Liang, F. Wen, and H.-Y. Shum, "Image vectorization using optimized gradient meshes," *ACM SIGGRAPH 2007 papers on - SIGGRAPH '07*, pp. 1-7, 2007.

16] Y. K. Lai, S. M. Hu, and R. R. Martin, "Automatic and topology-preserving gradient mesh generation for image vectorization," *TOG ACM Trans. Graph. ACM Transactions on Graphics*, vol. 28, no. 3, pp. 1-8, 2009.

17] P. Barla and A. Bousseau, "Gradient Art: Creation and Vectorization," *Computational Imaging and Vision Image and Video-Based Artistic Stylisation*, pp. 149–166, 2012.

18] J. Sun, L. Liang, F. Wen, and H. Y. Shum, "Creating optimized gradient mesh of a vector-based image from a raster-based image." US Patent, US 8773423 B2, 8 July, 2014

19] S. Hu and Y. Lai, "Method and system for rapidly vectorizing image by gradient meshes based on parameterization." US Patent, US 20120113098 A1, 10 May, 2012

20] S. Jeschke, D. Cline, and P. Wonka, "Estimating Color and Texture Parameters for Vector Graphics," *Computer Graphics Forum*, vol. 30, no. 2, pp. 523–532, 2011.

21] "ON1, Inc.," *Resize 10*. [Online]. Available at: https://www.on1.com/apps/resize10/. [Accessed: 10-May-2016].

22] R. N. Aufmann, J. Lockwood, R. D. Nation, and D. K. Clegg, *Mathematical excursions*. Cenage Learning, p.454, 2012.

23] M. D. Adams, *Multiresolution signal and geometry processing: filter banks, wavelets, and subdivision*. Michael Adams, p.343, 2013.

24] J. M. Blackledge, A. K. Evans, and M. J. Turner, *Fractal geometry: mathematical methods, algorithms, applications*. Chichester, West Sussex: Horwood Pub. for the Institute of Mathematics and its Applications, p.150, 2002.
25] P. Bauer, *Special edition using Adobe Illustrator 10*. Indianapolis, IN: Que, p.346, 2002.
26] P. M. Nugues, *Language processing with Perl and Prolog: theories, implementation, and application*. Springer, p.110, 2010.
27] S. Krig, *Computer vision metrics: survey, taxonomy, and analysis*. Apress, p.186, 2014.
28] S. C. Gustafson and G. J. Meyer, "Spline-based neural networks for digital image interpolation," *Applications of Artificial Neural Networks in Image Processing VI*, 2001.
29] C. B. Atkins, C. A. Bouman, and J. P. Allebach, "Semantic tree based resolution variants," *Resolution Methods for the Decision Problem Lecture Notes in Computer Science*, pp. 93–129, 1993.
30] J. Go, K. Sohn, and C. Lee, "Interpolation using neural networks for digital still cameras," *IEEE Transactions on Consumer Electronics IEEE Trans. Consumer Electron.*, vol. 46, no. 3, pp. 610–616, 2000.
31] H. Hu, P. M. Hoffman, and G. D. Haan, "Image Interpolation Using Classifi cation-based Neural Networks ," *Proc. ISCE*, pp. 133–137, 2004.
32] H. Aokage, K. Kameyama, and K. Wada, "Image interpolation using feed forward neural network, " *Proc. 23rd IASTED Int. Multi-Conf. on Artificial Intelligence and Applications*, Innsbruck, Austria, pp. 861-866, 2005.
33] N. Tasfi, "Image Scaling using Deep Convolutional Neural Networks — Flipboard Engineering," *Flipboard Engineering*. [Online]. Available at: http://engineering.flipboard.com/2015/05/scaling-convnets/. [Accessed: 10-May-2016].
34] J. Yang, J. Wright, T. S. Huang, and Y. Ma. "Image super-resolution via sparse representation." *TIP*, vol.19, no.11, pp. 2861–2873, 2010
35] R. Timofte, V. De, and L. V. Gool, "Anchored Neighborhood Regression for Fast Example-Based Super-Resolution," *2013 IEEE International Conference on Computer Vision*, pp. 1920-1927, 2013.
36] R. Zeyde, M. Elad, and M. Protter, "On Single Image Scale-Up Using Sparse-Representations," *Curves and Surfaces Lecture Notes in Computer Science*, pp. 711–730, 2012.
37] H. Chang, D.Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," *IEEE Conference on Computer Vision and Pattern Classification (CVPR)*, vol. 1, pp. 275-282, 2004.
38] A. Karpathy, "ConvnetJS demo: Image 'Painting,'" *ConvNetJS demo: Image Painting*. [Online]. Available at: http://cs.stanford.edu/people/karpathy/convnetjs/demo/image_regression.html. [Accessed: 10-May-2016].
39] K. Sims, "Artificial evolution for computer graphics," *ACM SIGGRAPH Computer Graphics SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 319–328, Feb. 1991.
40] "Picbreeder, Behind the Scenes," *Picbreeder: Collaborative Art Evolution*. [Online]. Available at: http://picbreeder.org/behind.php. [Accessed: 10-May-2016].
41] J. Eastland, "Feathered Dusk," Wiki Commons. [Online]. Available at: https://commons.wikimedia.org/wiki/File:Feathered_Dusk.jpg. [Accessed: 1-May-2016]
42] A. Lorimer "Extrapolatable Function Graphics," AODLorimer. [Online]. Available at: http://www.aodlorimer.com/#!extrapolatable-function-graphics/cedj. [Accessed: 1-May-2016]
43] A. Lorimer "The Development of Scalable Function Graphics," AODLorimer. [Online]. Available at: http://www.aodlorimer.com/#!function-graphics/cvht. [Accessed: 1-May-2016]
44] W. C. Siu, "Review of image interpolation and super-resolution," *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1–10.
45] X. He, *MultiMedia modeling: 21st International Conference, MMM 2015, Sydney,NSW, Australia, January 5-7, 2015: proceedings*. Springer, p.16, 2015.
46] J. C. Pinoli, *Mathematical Foundations of Image Processing and Analysis, Volume 2*. John Wiley & Sons, 2014.

47] P. Nicholas, *Scala for machine learning*. Packt Publishing, p.226, 2014.
48] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pp. 1–8, 2009.
49] Y. Bengio and J.-S. Senecal, "Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model," *IEEE Trans. Neural Netw. IEEE Transactions on Neural Networks*, vol. 19, no. 4, pp. 713–722, 2008.
50] Nagadomi, "Waifu2x," *GitHub*. [Online]. Available at: https://github.com/nagadomi/waifu2x. [Accessed: 10-May-2016].
51] Nagadomi, "Waifu2x," *Waifu2x*. [Online]. Available at: http://waifu2x.udp.jp/. [Accessed: 10-May-2016].
52] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, 11, pp. 625–660. 2010
53] L. Pasa, and A. Sperduti, "Pre-training of recurrent neural networks via linear autoencoders." *Advances in Neural Information Processing Systems*, 27, pp. 3572–3580. 2014
54] A. Karpathy, "CS231n Convolutional Neural Networks for Visual Recognition," *CS231n Convolutional Neural Networks for Visual Recognition*. [Online]. Available at: http://cs231n.github.io/neural-networks-2/#init. [Accessed: 10-May-2016].
55] A. S. Pandya and R. B. Macy, *Pattern recognition with neural networks in C++* . CRC Press, p.47, 1996.
56] R. Klette, *Image and video technology: 6th Pacific-Rim Symposium, PSIVT 2013, Guanajuato, México, October 28 - November 1, 2013 ; proceedings*. Springer, p.511, 2014.
57] M. Nielsen, "Improving the way neural networks learn," *Neural networks and deep learning*. [Online]. Available at: http://neuralnetworksanddeeplearning.com/chap3.html. [Accessed: 11-May-2016].
58] S. Momami, *2013 International Conference on Electrical, Control and Automation Engineering(ECAE2013)*, DEStech Publications, p.568, 2014.
59] D. Mclean, Z. Bandar, and J. O'Shea, "Improving Generalisation Using Modular Neural Networks," *Artificial Neural Nets and Genetic Algorithms*, pp. 35–39, 1999.
60] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," *Advances in Neural Information Processing Systems*, pp. 598–605. 1990
61] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," Proceedings, IEEE Int. Conf. Neural Networks, pp. 293–299, 1992.
62] A. W. Jayawardena, *Environmental and Hydrological Systems Modelling*, CRC Press, p.223, 2014
63] R. Reed, "Pruning algorithms—A review." *IEEE Transactions on Neural Networks,* pp. 740–747. 1993
64] G. M. Amdahl, "Computer Architecture and Amdahl's Law," *IEEE Solid-State Circuits Newsl. IEEE Solid-State Circuits Newsletter*, vol. 12, no. 3, pp. 4–9, 2007.
65] L. Chambers, "Mac Pro Westmere Performance with PhotoZoom Pro," *Mac Performance Guide*. [Online]. Available at: http://macperformanceguide.com/reviews-macprowestmere-photozoompro.html. [Accessed: 10-May-2016].
66] D. Lea, *Concurrent programming in Java: design principles and patterns*. Addison Wesley, p.119, 1997.
67] D. A. Padua, *Encyclopedia of parallel computing*. Springer, p.1053, 2011.
68] V. M. Krasnopol'sky, *The application of neural networks in the earth system sciences: neural networks emulations for complex multidimensional mappings*. Springer Science & Business Media, pp.14–15, 2013.
69] P. Haggar, *Practical Java: programming language guide*. Addison-Wesley, p.127, 2000.
70] "Perfect Resize User Manual," *On1 Software*, p.49, 2007
71] W. Birkfellner, *"Applied Medical Image Processing, Second Edition: A Basic Course"*. Taylor & Francis, p.126, 2014.

72] “True Color Kodak Images,” *Kodak*. [Online]. Available at: http://r0k.us/graphics/kodak/index.html. [Accessed: 10-May-2016].
73] R. Englander, *Developing Java beans*. O'Reilly, p.128, 1997.
74] M. J. P. Cullen, M. A. Freitag, S. Kindermann, and R. Scheichl, *Large scale inverse problems: computational methods and applications in the earth sciences*. Walter de Gruyter, o.136, 2013.
75] C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun, “Multiscale texture synthesis,” *ACM SIGGRAPH 2008 papers on - SIGGRAPH '08*, 2008.

Appendices
9.0

9.1 Atromatix Communication – Appendix 1

Personal email communication with the startup company Artomatix, as a query regarding the suitability of multi-scale texture synthesis for photographic image enlargement.

Multi-scale texture synthesis is a tool offered commercially by the company (founded in 2014 by Dr. Eric Risser, who co-authored the 2010 research paper Multiscale Texture Synthesis).[75] There technology produces impressive results when synthesising infinite sub-pixel textures, but it is currently limited to texture-based images that exhibit a very high degree of structural repetition.

Alex Lorimer
3:26 am, March 24th 2016
I'm interested in what the texture synthesis tool allows me to do. If it were to be applied to an image such as a human face, what would the zoomed in version end up looking like? Can it be applied to infinite photograph enlargement?

Tom (Artomatix)
9:26am, March 24th 2016
Our CTO has done research in that area (infinite photograph enlargement), but it isn't available at the moment as a product. I managed to root out an old video of infinite zoom on Van Gogh's Starry Night from an old paper of his here: http://www.cs.columbia.edu/cg/mts/starry.wmv

9.2 PhotoFunction Code Organisation Overview – Appendix 2

This section can be used in conjunction with the online repository discussed in the introduction of this document. The Login details and URL for the repository are provided below. Login is necessary before visiting the URL.

Temporary Username:
[redacted]
Temporary Password:
[redacted]
Repository URL
[redacted]

The classes that compose the current PhotoFunction implementation are listed and described in terms of overall functionality below. Please note that PhotoFunction was developed as part of an experimental, learning-based process. The arrangement and organisation of classes is not ideal, and is intended to be altered and improved during the summer project.

ThePackage/...

GUI.java
The GUI class contains the main method, along with code that lays out the program window and its content. GUI also contains the event listeners that respond to the user's input in terms of clicks and field entries. GUI delegates to Iterate.java, Target.java, LoadNN.java, to perform more specialised tasks.

Iterate.java
The iterate class is called directly from GUI, handling the bitmap-SFG conversion process in the background so that the user interface is still responsive. Iterate divides the Target.png image into tiles (which are actually stored in the Target class) and organises the conversion process in terms of learning stages, selecting data points, and querying neighbouring networks to share information during the conversion process. Iterate delegates further to NN.java, Target.java, Test.java, EvaluateFitness.java, and BoxBlurFilter.java to perform more specialised tasks.

NN.java
NN.java is the neural network class, composed of NeuronLayer objects, which themselves are composed of Neuron objects. The process of constructing a neural network is handled in the NN constructor method, which iteratively delegates down to the constructor of NeuronLayer.java and subsequently to the constructor of Neuron.java. NN also features a host of different activation functions that have been experimented with (while only sine and atan, and their derivative functions, are used). It also handles the process of forward and backpropagation, which are methods called by the Iterate class.

Neuron.java
Acts as a container for neuron related information such as a record of input, output, and error values; referred to by NN.java. These values are necessary to perform error backpropagation.

NeuronLayer.java
A simple class that acts as a container that organises neurons into an array that represents a single layer in the neural network, referred to by NN.java.

Target.java

The constructor of the Target class is called by GUI.java, prompting Target to load the Target.png image from a folder called Convert (this is the bitmap image that is to be converted to an SFG). Target also contains an array of BufferedImage objects that are empty until Iterate.java assigns an image tile to each one. Iterate then delegates to Target.java during the conversion process by querying for the pixel values at specific co-ordinates.

Test.java

The Test class handles the process of saving SFG files (which are essentially serialised array objects which contain neural network information). Test also loads SFG files, deserialising and returning them as arrays. Test.java is called by Iterate periodically and at points of disruption (such as when GUI.iteratePause = = true), in order to save progress. It is also called by LoadNN.java.

LoadNN.java

LoadNN.java is a class that handles the process of rendering an SFG, essentially converting it back to bitmap format at a specific scale. This is called by the GUI class in response to user input (specifying the render dimensions). LoadNN.java delegates to Test.java to perform the action of loading the appropriate SFG file (called SFG.sfg) in the folder called Render.

BoxBlurFilter.java

BoxBlurFilter is a class that blurs a BufferedImage by a specified amount. This is called in the Iterate class. (A slightly blurred target image can be useful in the first phase of SFG conversion, resulting in increased SFG sharpness after the second phase of conversion, where pixels begin to be treated as domains as opposed to absolute data points).

EvaluateFitness.java

Finally, the EvaluateFitness class is called by Iterate.java periodically to determine how accurately rendered SFGs match the target image. This is actually also determined in the Iterate class (as the difference between actual output and target values, driving the training process). However EvaluateFitness.java performs this actions across an entire image in one go (calculating error based on the SFG and Target.png image as a whole rather than on randomly selected, individual data points). This provides a more accurate measure of whether or not a conversion process can be deemed complete.