CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

SCHOOL OF COMPUTER SCIENCE AND INFORMATICS

FINAL PROJECT REPORT

SESSION 2021-2022

Name:	**Robert Turton**
Student Number:	**C1890422**
Degree Programme:	**Computer Science MSc**
Project Title:	**Automated Generation of Pixel Art**
Supervisor:	**Yukun Lai**
Assessor:	
Date:	**19/09/2022**

Declaration:

I have read and understand Appendix 2 in the Student Handbook: "Some advice on the avoidance of plagiarism".

I hereby declare that the attached report is exclusively my own work, that no part of the work has previously been submitted for assessment (although do note that material in "Interim Report" may be re-used in the final "Project Report" as it is considered part of the same assessment), and that I have not knowingly allowed it to be copied by another person.

Contents

Abstract

Significant research has been put into the creation of automated algorithms to generate pixel art from high resolution images. This thesis explores a combination of these automated techniques with a genetic algorithm guided by a human in the loop to optimise the resultant pixel art image. A series of images from contemporary research on the automated generation of pixel art are used to generate pixel art images using the guided genetic algorithm approach; these are compared with the output of fully automated approaches and the work of pixel artists. It is found that genetic algorithms are effective at optimising the output of automated pixel art generation techniques to a standard that is comparable with, and in some cases superior to the automated techniques. Some further work is required to finally meet the standard of the best pixel artists and avenues are identified to improve the established GA approach.

Acknowledgements

The author like to thank his supervisor for taking the time to supervise this masters dissertation, and his parents and friends for their support during the whole process.

1 Introduction

Forms of pixelated artwork have existed since ancient times, the earliest examples perhaps being the mosaics dating back to 300BC [1]. In a more modern setting however the term “pixel art” largely refers to art created by the exclusive, deliberate, positioning of pixels typically on a computer screen. Pixel art is often confused with other forms of digital art when they are resized to very low resolutions, however pixel art is distinct as it is a stylised form of art that often loses the form and proportion of an image to convey the important features of an image e.g., Eyes, Hair, Smile [2]. This dissertation will be investigating the automated generation of pixel art with a human in the loop to guide the optimisation function used to achieve the desired outcome. In this case genetic algorithms will be used [3] to achieve optimisation, inspired by the Identikit system approach [4].

1.1 Aim

The aim of this thesis is to confirm or otherwise the hypothesis that a genetic algorithm with a human in the loop will be effective in guiding the automated generation of pixel art.

1.2 Objectives

In order to achieve this aim, the following objectives have been set:

- Perform a literature survey to establish the state of the automated generation of pixel art.
- Analyse and construct an evolutionary algorithm to guide the automated generation of pixel art
- Create a user interface through which the evolutionary algorithm can be guided
- Run tests to compare current best practice against automated pixel art generation to confirm or otherwise the effectiveness of the evolutionary approach

1.3 Layout of Thesis

This thesis is divided into eight sections which will be outlined here.

The first section is the introduction which gives a brief overview of the thesis topic, aims and objectives as well as the project plan.

The second section is a literature review of current research around pixel art and topics related to the methodology of the thesis.

The third section describes project analysis, design and requirements including a down selection of the options considered following the literature review.

The fourth section summarises the theory behind major concepts used in the thesis.

The fifth section lists the software libraries and hardware used in the creation of the thesis.

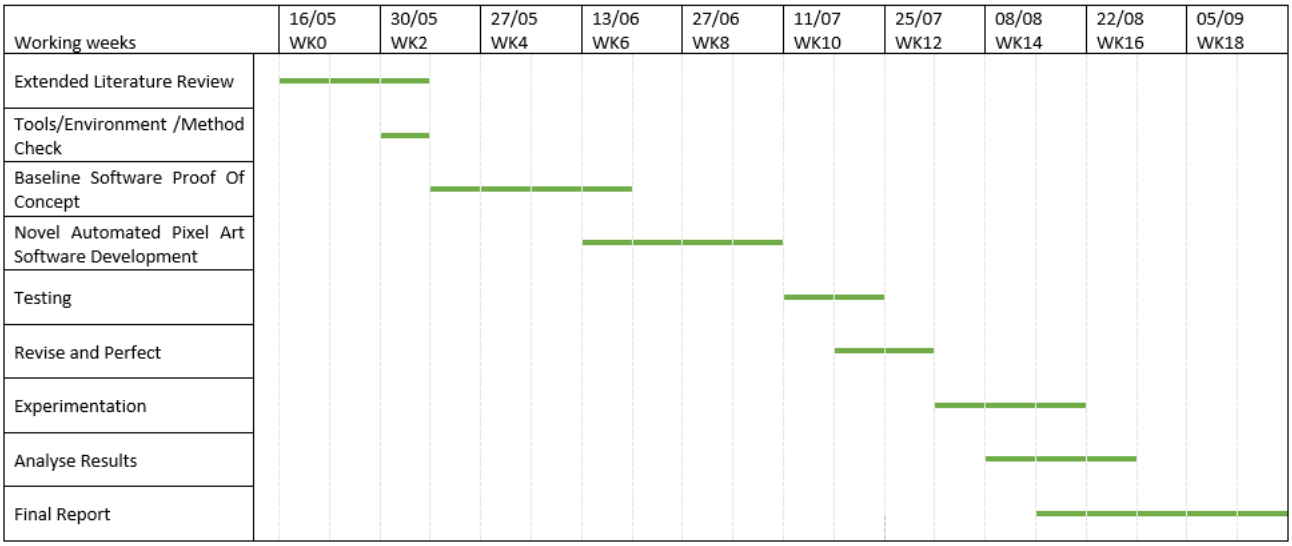
The sixth section details the specifics of implementation, outlining the algorithm and user interface created as part of the thesis.

The seventh section concerns the results and analysis of experiments performed to ensure the thesis has met its aims and objectives set out in section one.

The eighth section is a summary of the project and its results and provides the conclusions of the thesis and a reflection on the work done.

1.4 Project Plan Outline

The dissertation was planned and tracked using the Gantt chart below.



The headings in the chart are as follows:

Extended Literature Review- There was an initial literature review performed before the specific project topic had been set. A new more focussed literature review was required once the topic was selected, for inclusion in this thesis.

Tool/Environment/Method Check- The environment and tools used for the thesis were reviewed and tested before the bulk of work began to ensure they were well understood and provided the utility required for the thesis

Baseline Software Proof of Concept- Basic reproductions of algorithms relevant to automated pixel art generation were performed to ensure that similar methods could be introduced as part of the novel approach in this thesis.

Novel Automated Pixel Art Software Development- During this period the novel approach to automated pixel art generation was developed. This encompasses the creation of the evolutionary algorithm and its incorporation with current pixel art generation techniques and the creation of a user interface to guide and test the process

Testing- Once developed the software was tested for errors and appropriate results before moving to experimentation

Revise and Perfect- After noting any error in the code or output the algorithms were refined to improve efficiency and effectiveness as well as to fix any bugs before moving to experimentation

Experimentation- Experiments were performed to assess the effectiveness of the automated pixel art generation method.

Analyse Results- Results from the experiments were analysed and compared to current research.

Final Report- All sections were collated into a final report for the thesis.

2 Literature Review

The aim of this literature review is to provide a firm basis of knowledge of the current research into the automated generation of pixel art and related topics as required by the thesis. It begins with a brief section on the background and historical context of pixel art and then goes on to analyse key research areas related to the automated generation of pixel art.

From the definition of pixel art given by [5] the first two critical aspects of pixel art are a limited colour palette and a very low image resolution. Another aspect which commonly appears in papers concerning the automated generation of pixel art is edge detection as naïve down-sampling often fails to preserve the edges of features [6]. This is a particular issue with pixel art as singular misplaced pixels can have a disproportionate effect on the quality of a resulting image as the total number of pixels is so limited. In order to assist in the generation pixel art many methods of optimisation and different coordinate systems have been considered [3][7][8].

These main areas will form the basis for the key research headings in this literature review.

2.1 History and Background of Pixel Art

The term pixel was first published describing discrete picture elements in 1965 [9], however widespread use of the term did not begin until the 1980s where it began to appear in the titles of publications. One of the first uses of the term pixel art was a column on “pixel art” in 1982 [10]. Forms of pixelated artwork have existed since ancient times (fig.1) [1], however today pixel art is perhaps best known for its role in the graphics of old video games, where the limitations of the displays necessitated the use of low-resolution images [11], however more recently there has been interest in “hi-bit” pixel graphics as a chosen art style [12].

Fig.1 1st Century AD Mosaic- Example of early use of a pixelated art form [1].

The increased demand for pixel art comes with significant challenges however, as while the issue of displaying sufficient pixels on a screen has been solved, the principal challenge of generating pixel art remains; namely the ability to represent recognizable images with severely restricted resolutions and palettes.

Currently pixel art is produced principally by artists experienced in the style, and guides have been developed specifically for this art form [13], of course such methods of asset generation are expensive, so interest has been growing in ways to automatically generate pixel art.

Example areas of modern research relevant to the generation and manipulation of pixel art include:

- Down-sampling
- Edge detection
- Hex-Based pixels.
- Palette selection

2.2 Down-sampling

Pixel art images are typically much smaller than a more traditional representation of an image, and since often these larger images are used as a basis for the generation of pixel art they must be down-sampled for use as pixel art. The field of down-sampling images is very broad and not restricted to pixel art, however listed below are examples that were found that can be directly applied to the generation of pixel art:

- Content adaptive scaling to balance sharpness with avoiding artefacts [14]
- Unsupervised neural network with forward and backward learning enhancing sharpness [15]
- Using perceptual image approaches to optimise down-sampling [16]
- The use of convolutional filters to efficiently preserve visually important details [17]
- The maintenance of high frequency patterns by controlling aliasing [18]
- The identification and maintenance of “thin structures” [19]

Typically, in down-sampling images a low-pass 2D filter is applied to the image which prevents aliasing of the high frequencies into low frequencies as the image is sampled. An alternative is to fit a series of points and then sample from that fitted model.

When reading into this topic it is apparent that edge detection is important and indeed the professional artist’s approach [20][2] showed much clearer edges than seen in the automated down-

sampling technique when generating pixel art. Edge detection techniques are therefore an interesting potential area of study in the automated generation of pixel art.

2.3 Edge Detection

Much like down-sampling edge detection is a very broad field not restricted to its use in the generation of pixel art. Multiple approaches have been considered and applied to varying degrees of success when applied to pixel art, examples include:

- The application of Gaussian filtering algorithms to assist in edge detection [21]
- The use of Laplacian operation to detect intensity changes in 1D and 2D schema [22]
- The use of "thresholding" to introduce sharp contrast to detect edges [23]
- Considering the use of range and standard deviation of pixel intensities rather than traditional estimates of digital gradients for edge detection [24]
- The use of a combination of approaches which fundamentally integrates a direction operator and a maximum gradient detector whilst smoothing out noise [25]

Many of these papers involve complex mathematics, which would require significant study to fully understand and apply. As this is not a mathematics project, emphasis will be placed on simply reproducing proven edge detection methods for the purposes of this project if edge detection is necessary. However, this is only one aspect of automated pixel art generation and the main driver in this case is the maintenance of key features for which edge detection is just one tool.

2.4 Hex-based Pixels

Whilst most traditional approaches use rectilinear coordinate systems [26], another well documented approach is to use hex-based coordinate systems [8]. Given the prevalence of rectilinear systems in hardware conversion between square and hexagonal systems, there is a challenge [27], however hexagonal sampling systems have shown to be more computationally efficient that rectilinear [28][29].

In the context of pixel art the properties of hex co-ordinates may, for example, assist in edge detection as described in Fitz [29], or prove to produce aesthetically pleasing results.

Gerstner [20] remarks on how the Super-pixel approach seems to favour a hexagonal pixel arrangement, though a recti-linear approach is used in the end. The use of a hex-based approach is intriguing and may work well with a style of approach similar to that used by Canny [25] for edge detection.

2.5 Palette Selection

A critical part of creating effective pixel art is colour palette selection, particularly as the many colours in a high-resolution image must be selected down to a very limited selection for pixel art. Most methods of palette selection rely on the process of super pixelation and a method of selecting a colour for the palette within each pixel, an example of how each problem may be approached is given below.

- MCDA (Mass Constrained Deterministic Annealing) applied to colour selection [20] potentially with user input [2]
- K-means clustering for colour quantisation [30]
- The use of SLIC (simple linear iterative clustering) when generating super-pixels for palette selection [31]

Typically, papers which study the automatic generation of Pixel Art, focus on unsupervised techniques [20]. When human interaction is included in the process in even a small way, often significant improvements can be made to the end results [2].

One interesting approach used in the past to produce suitable images is the use of image creation for identikit purposes, curated by human intuitive knowledge. This has been successful in the past at producing significantly better results than automatic generation alone [4] through combination with genetic algorithms [3]. This 'human-in-the-loop' approach is very unusual and as far as can be seen has never been used for colour palette selection and is therefore of interest. A novel approach may be to use a similar combined method of automatic creation and manual curation in combination with genetic algorithms to improve palette selection (as well as to assign the colours to pixels). This would favour rapid pixel art generation techniques to support colour palette evolution area and is therefore of interest.

2.6 Supporting Papers

2.6.1 Evolutionary Algorithms

Evolutionary algorithms are a form of optimisation which work well with discontinuous search spaces [3]. Pixel art optimisation is an example of a discontinuous search space where very smaller changes in the image, such as the placement of a single critical pixel, can dramatically affect the quality of the output. 'Evolutionary algorithm' is the general term that covers genetic algorithms as a sub-set, in this thesis a genetic algorithm is applied to generation of pixel art. For the purpose of this thesis, it will be assumed that a genetic algorithm may use non-binary bit strings for the

chromosomes. There are published examples of their implementation in python, for example in Farrell's book [7].

Evolutionary algorithms are a well-researched field which many publications cover, one of the principal factors which determines the effectiveness of an evolutionary algorithm is the quality of the fitness function, that is the ability of the algorithm to identify good and bad results [32]. Due to the subjective nature of pixel art however, it is difficult to create a fully automated fitness function to evaluate algorithmic performance. The identikit paper [4] offers a potential solution to this issue for the case of pixel art, by including a human in the loop to guide the algorithm to create the desired results.

2.6.2 Signal Filtering

Given the damage noise could potentially cause to a pixel art image, and due to the outsized impact (when compared with more traditional high-resolution images) each individual pixel has on the quality of the final image, some papers on digital filtering to remove potential stray pixels have been investigated. There are several promising techniques which could be applied, however due to the very small number of pixels in any dimension, larger filters are not appropriate. The simplest filter of an appropriate type to remove individual 'outlier' pixels is the median filter [33]. This filter simply takes an area (mask) around a pixel and replaces its value with the median of the points within that area. The reference given covers an adaptation which optimises the mask area, however for this work the area must be minimised so the proposed optimisation is not required. Further options can be found in [34].

2.7 Literature Review Summary

In the literature review it was found that the pixel art generation lies at the intersection of many fields of research, primarily those fields concerned with image manipulation. Many well-known techniques have already been applied with some success, and automated systems are quite capable of producing passable pixel art. As of writing it appears that little research has been made into the combination of genetic algorithms in the generation of pixel art, possibly due to the difficulty in creating a suitable automatic fitness function to guide the algorithm to high quality solutions. Potentially with the inclusion of a human in the loop to guide the fitness function it will be possible to apply a genetic algorithm to the automated generation of pixel art. Other areas which must be considered are the use of non-traditional coordinate systems such as the hex-based approach, and further refinements to edge detection, down-sampling, or palette selection techniques. The practicality of each possible approach for the thesis is addressed in detail in the following section.

3 Project Analysis and Selected Approach

In this section the potential directions for the thesis arising from the literature review are considered and down-selected, based on the requirements of the thesis, and the assessed practicality of implementation of each approach.

3.1 Approach Option Analysis

3.1.1 Implementation of edge detection in combination with simulated annealing

It has been noted that in one particularly successful method for automatically generating pixel art using simulated annealing [20], the authors did not make use of sophisticated edge detection techniques. One potential direction for the thesis would be to replicate the simulated annealing approach and attempt to combine this process with suitable edge detection either by simple gaussian filtering [21] or thresholding [23].

This direction is attractive as the approaches have already been proven and there are good examples of how each method has been achieved in the past. The approach does however lack novelty beyond the combination of the two approaches and will require full understanding of two rather complex techniques, that of simulated annealing and edge detection. It is therefore 'high risk' especially given the mathematical complexity of many of the techniques involved.

3.1.2 SLIC based pixel art generation using hexagonal coordinate system

SLIC has been used previously as part of the process of automated pixel art generation [20] however the super pixel approach computationally favours a hexagonal coordinate system [28][29] which was not used in previous research. One direction for the thesis would be to replicate the simulated annealing approach used by previous papers but use the hexagonal coordinate system and test if the hypothetical improvements to edge detection [29] and computational efficiency [28] materialise.

This direction is attractive for similar reasons to the previous approach described as the technologies involved have already been proven and used in published literature. This would provide a solid basis on which to construct the thesis algorithms at the cost of novelty. Another issue with this approach in particular is the mathematical challenges of converting an unfamiliar and complex technique like simulated annealing with edge correction to a new coordinate system, with the risk that despite the added complexity the changed system does not result in a significant change to the overall results.

3.1.3 Use of a genetic algorithm with a human in the loop to guide pixel art generation

Taking inspiration from the identikit application of human-in-the-loop for the improvement of graphical images using a genetic algorithm [4] one potential direction for the thesis would be to use

a human-in-the-loop to guide a genetic algorithm to the generation of pixel art from high resolution images. This would require the creation of a palette selection and pixel colour allocation technique that generates a varied population of pixel art images according to variables that can be controlled by a genetic algorithm.

This approach is advantageous firstly due to its novelty as the literature review did not find papers that used a genetic algorithm as part of pixel art generation. Furthermore, by using the genetic algorithm approach the mathematics involved in pixel art generation may be simplified greatly, as the genetic algorithm can be relied upon to iteratively improve palette selection and pixel placement without the need for a single complex algorithm to create an acceptable result. The largest drawback to this technique is that the approach is unproven, and there may be good reasons why there is little research looking at this approach. Another issue is that by introducing a human in the loop the fully automated generation of pixel art is lost as we require human input to guide the pixel art generation algorithm

3.2 Down-selection

Ultimately it was decided that the novelty and relative reduction in complexity made the use of a genetic algorithm in combination with pixel art generation the best approach for this thesis.

4 Summary of Concepts

In this section major concepts which are later used in the implementation of the thesis code are summarised.

4.1 Genetic Algorithms

Genetic algorithms are a form of evolutionary algorithm which emulate the process of adaptation by living organisms to their environment in order to optimise a particular process. This method of optimisation relies, as in nature, on five critical processes:

- The generation of a diverse set of “chromosomes” containing “genes” which control characteristics which determine how the algorithm behaves and generates its output.
- The ability for the “fitness” of a function to be determined so that the algorithm may test how well any particular chromosome fulfils a desired set of objectives.
- The ability for the measured fitness to be used to competitively select chromosomes which result in more desirable outcomes.
- The process of “crossover” wherein chromosomes can exchange characteristics and in so doing create a new population with characteristics from both parents.
- The chance of “mutation” where a gene in a chromosome will spontaneously change to another value between generations.

The first obstacle therefore, to the construction of a genetic algorithm, is framing the problem to which the algorithm will be applied in a manner which constructs a set of defined variables which when changed can result in a range of possible solution outputs. An example would include each chromosome having a gene which when coded to a 0 would result in a red pixel being displayed on screen and when coded to 1 would display a green pixel on screen. If many such genes exist, which code for different behaviours, the combination of all these genes (which don’t have to control behaviour in such a binary manner, i.e. 2 could code for yellow in our previous example) could create a great variety of potential solutions.

The second obstacle is often the most challenging for such algorithms as it asks how we evaluate the effectiveness of a particular solution. Sometimes this can be simple, if a particular target number were being optimised for (for example), the result produced by a mathematical function acting on the chromosome could simply be checked for similarity to that target number. However, if the desired result is not easily mathematically defined, such as in the case of determining the quality a piece of artwork, the fitness function can be much more challenging to determine.

Once fitness has been determined there are many different methods of then allowing the chromosomes to compete. An effective method of competition is tournament selection. In this form of competition, a fixed number of chromosomes (typically four) are selected at random from the overall population and the winner selected for reproduction in the next generation. Other methods include the roulette wheel selection method, where members of a population are assigned an area of a wheel in proportion with their fitness and then a random point is picked on the wheel when selecting members of the next generation, with the chromosome to whom the selected part of the wheel was assigned being passed on. There are of course many more methods of selection and variations on the methods presented above [32].

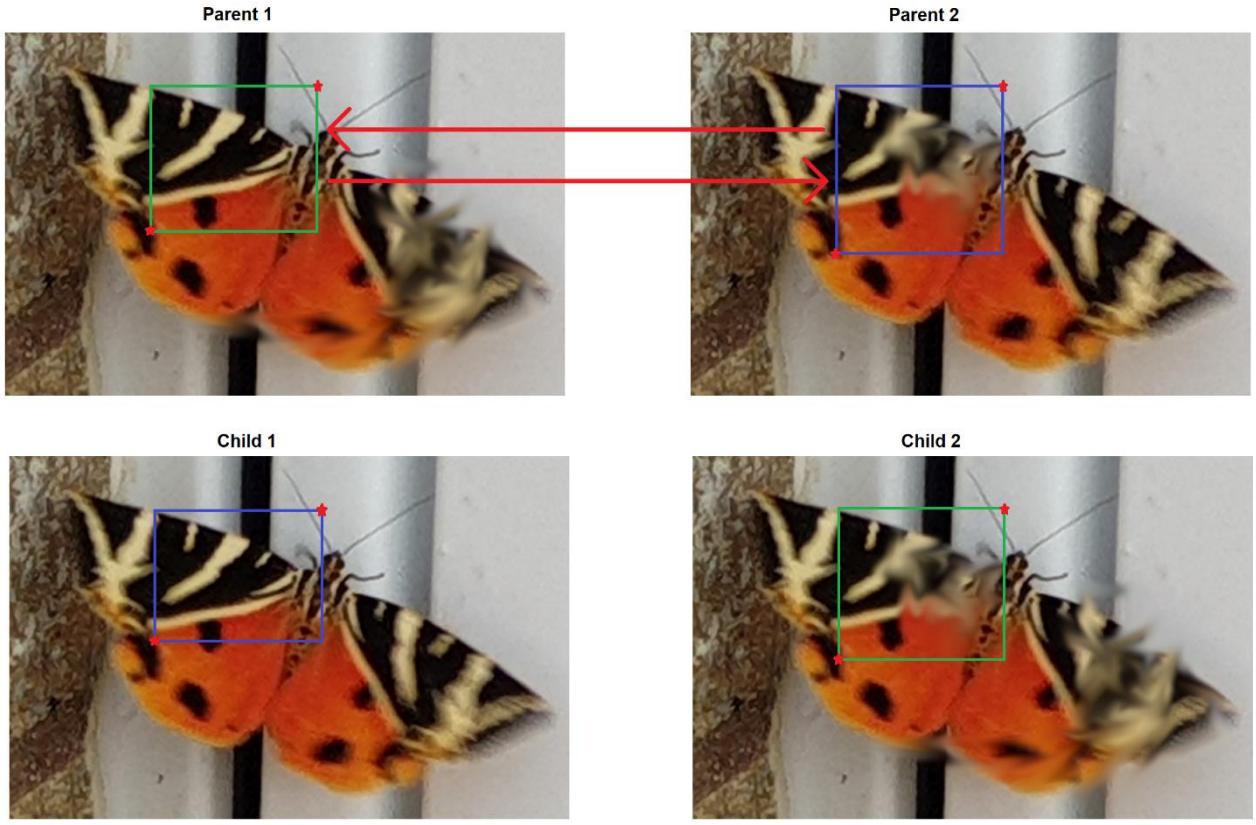


Fig.2 Two-Point image Crossover Example- Two images of a butterfly undergo two-point image crossover. Two points represented by red squares are used to draw boxes (green and blue) which are swapped to create the next generation.

Perhaps the most defining factor of a genetic algorithm is how once the members of the next generation are selected, they can swap the genes (crossover) that make up each chromosome to create a new population with a new combination of genes from the most fit members of the previous generation. When the genetic algorithm is successful this results in iterative improvement in output over many generations. A simple but effective method of crossover is uniform crossover. In

uniform crossover the two selected chromosomes are compared and genes that code for the same characteristic have some predefined chance (usually 50%) to swap between the chromosomes which results in two new unique chromosomes.

Another useful method is to perform a “two-point” crossover where two pixels are selected and all pixels in an imaginary box drawn between them (using the selected pixels as the corners) are swapped to create the next generation (fig.2). As in competition there are many options when considering crossover [32].

In the case of mutation, the effect and implementation are somewhat self-explanatory. A chromosome from the fit population will be selected and a random gene/s will be changed to a new value/s by some relevant mechanism.

Aside from variation in the methods of crossover, selection, and mutation, genetic algorithms can vary in the proportion of the population crossover and mutation are applied to. Some typical rates for crossover and mutation are 40% of the population performs crossover and 1% undergoes mutation.

4.2 Simple Linear Iterative Clustering (SLIC)

Simple linear iterative clustering is at its core a method by which groups of pixels may be clustered or placed into groups based on their proximity and colour. These groups are known as superpixels. This is achieved by first seeding the image with a number of initial centroids (centre-points) for each superpixel at regular positions across the image, these each have an area around them which is defined as their search region.

Pixels are assigned to the centroids based on distance, if the pixel is within the search region of the superpixel. This limited search space is part of what separates the SLIC methodology from other clustering methods as it reduced the number of distance calculation saving on computational efficiency. It should also be noted here that “distance” does not simply refer to proximity but rather also to the colour difference between the centroid pixel and a candidate for the superpixel group.

Following these assignments, the superpixel centroids are recalculated as the vector mean position and colour of all members of the group, and the process is repeated. Repetitions and movements of the superpixel centres continue until an equilibrium is reached where the superpixel centroids no longer change significantly between repeats. In this manner all pixels in the image will have been assigned to a superpixel group and the process is complete.

4.3 K-Means Clustering

K-Means clustering is a common and well-known clustering method. In K-means clustering the process begins with a set of randomly chosen centroids (centre-points) the number of which is decided by the user when the method is applied. Next, as in the SLIC algorithm, pixels are assigned to the closest centroid, however critically the search space is not limited in this more naïve clustering methodology. Following this step centroids are recalculated as the vector mean position of all cluster members and the process repeated until the centroids stabilise. As a result of the unlimited search space K-Means algorithms can be quite slow so a fair amount of research has been put into improving the speed of k-means clustering. [30]

4.4 Median Filters

The median filter is one of the simplest forms of smoothing filter where a mask is moved around a dataset which replaces noisy datapoints with the median value of its neighbours, the neighbours being defined by the shape of the mask. The shape of the mask is variable but is typically an NxN square, the smallest possible square median filter being a 3x3 square neighbourhood. There has been research into improving this simple idea, in particular how to determine when a pixel being analysed by the mask is a noisy datapoint in need of replacement by the median filter [33]

5 Hardware and Software Library Specifications

This section briefly lists the hardware and software specifications and libraries used in the creation of the thesis code.

5.1 Hardware

Processor- Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz 3.60 GHz

RAM- 16.0 GB

System Type- 64-bit, x64-based processor

5.2 Software

Operating System- Windows 10 Pro

Programming Environment- Spyder (Python 3.9)

Utilised Libraries-

- Pygame v1.9.6
- OpenCV-python v4.6.0.66
- SciPy v1.9.1
- Time v3.10.6
- Random v3.10.6
- Matplotlib v3.5.2
- Scikit-image v0.19.2 [35]

6 Implementation and Algorithm Design

In this section an overview is given of the implementation of the thesis code, including both a graphical flow chart representation, and a written explanation of the purpose of each section.

6.1 Overview

The code is effectively split into four sections:

- A loading section where the user specifies the image they wish to pixelate and defines the number of unique colours they desire in the final colour palette.
- A colour palette selection section which allows the user to manually edit the colour palette if desired based on some preview images and allows them to set the degree to which the colour palette varies in the initial genetic algorithm population.
- A genetic algorithm section where successive generations of the results from the GA are displayed and the best three images selected by the user assisting the GA each generation.
- A final confirmation section displaying the original image and the best three images generated by the algorithm in the final generation.

The code outputs the three best images along with the intermediate generational images in the 'jpg' format.

6.2 Loading Section

The loading section takes a user input of an image in any of the common image formats parsable by the OpenCV library. In the case of this dissertation 'jpg' and 'png' test image formats were used. The images to be selected must be placed in a folder titled "testImages" in the root directory containing the python code.

Once loaded the image undergos k-means clustering to cluster all pixels by colour alone, in order to restrict the palette to a user specified size. In order to create comparable images with current research, colour palettes of 8, 12 and 16 were selected for testing. The original image is then displayed against a full-size version of the original image using the restricted colour palette for final confirmation.

Fig.3 shows an example image "lena.jpg" being loaded with a palette size of 12, fig.4 describes the implementation of the code as a flow diagram.

Fig.3 Loading Section- Image file path is entered into the "input images file path" input box confirmed by a return keypress, palette size is entered into the "select palette size" input box confirmed by a return keypress.

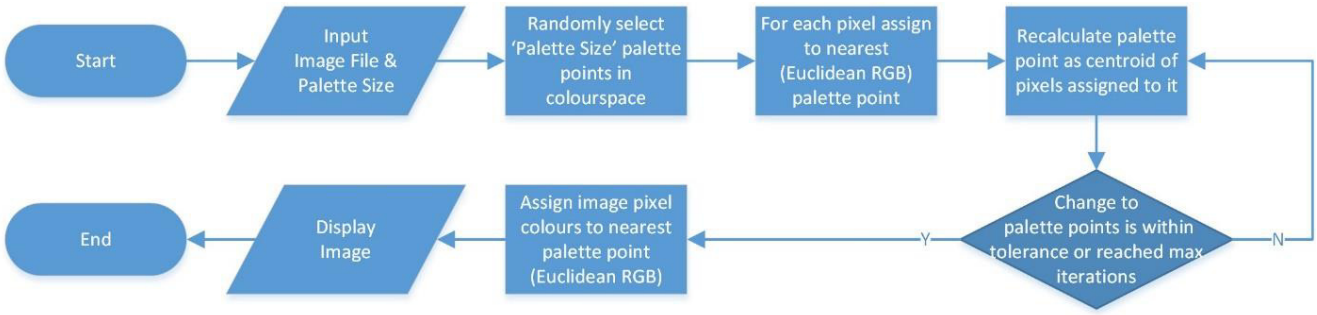


Fig.4 Loading Section Process- Flow chart representation of the implementation of the loading screen

6.3 Colour Palette generation

In this section of the code the colour palette is displayed alongside several representational images to allow the user to tailor the colour palette to their liking before initiating the GA process.

The visualisation screens include the full resolution restricted colour palette image, a naïvely down-sampled version of the image (using nearest neighbour down-sampling), and two SLIC images (created using the scikit-image library SLIC function) one with a large number of superpixels (864) the other with a smaller number (100). These images do not serve a functional purpose in the code but rather give the user an idea of how the changes they make to the colour palette will affect the results from the down-sampling process.

The entire colour palette is editable from this screen (fig.5), either by selection from a colour wheel or by editing RGB values directly for each colour. It is also possible to select colours directly from a copy of the original unedited image displayed next to the colour wheel. Another key element which is editable on this screen is the degree to which each colour in the colour palette may be varied by the genetic algorithm. The user also specifies the permitted range of variation (specifically the standard deviation) for the colours when generating the initial members of the GA population. Fig.6 describes the implementation of the palette ending screen as a flow diagram.

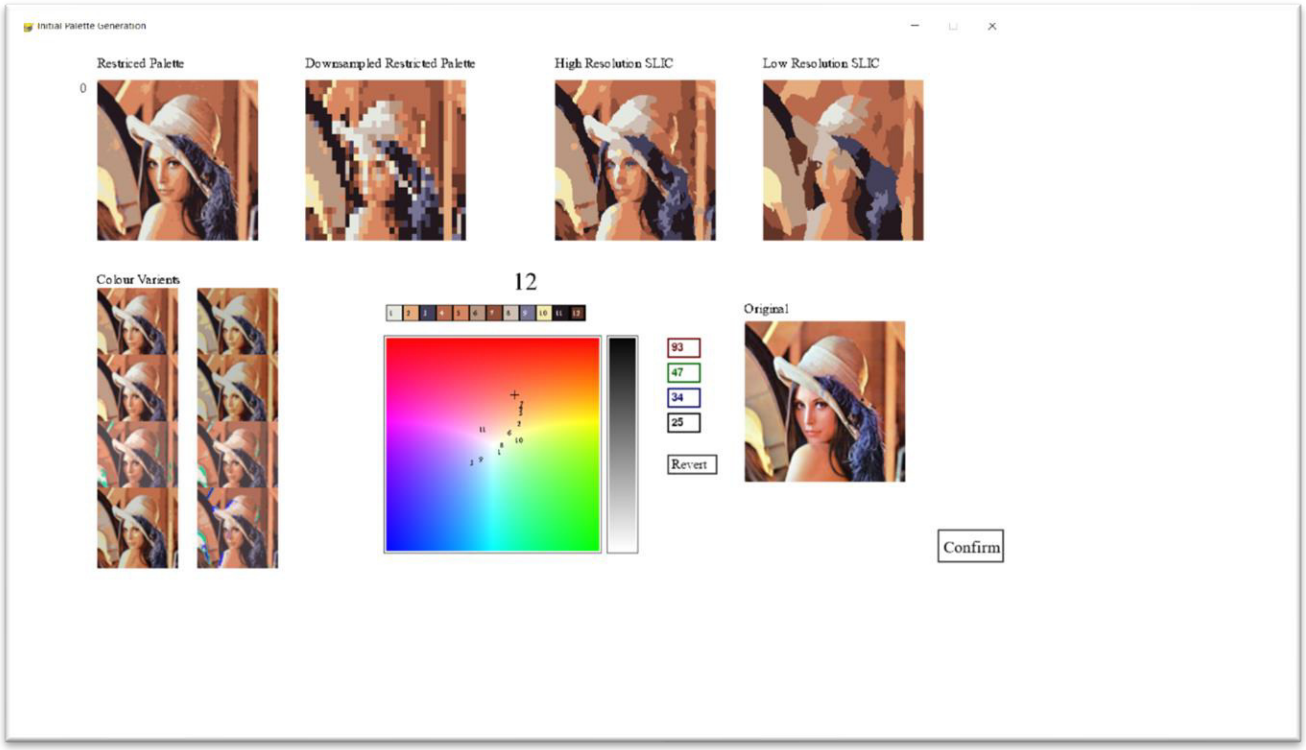


Fig.5 Palette Editing Screen- Individual palette colours can be edited by selection from the palette here palette colour 12 is selected. New colours may be entered by RGB value, selection from the colour wheel or selection from the "original" image. Palette variation may be controlled by the number in the black box below the RGB value entry boxes which are colour coded.

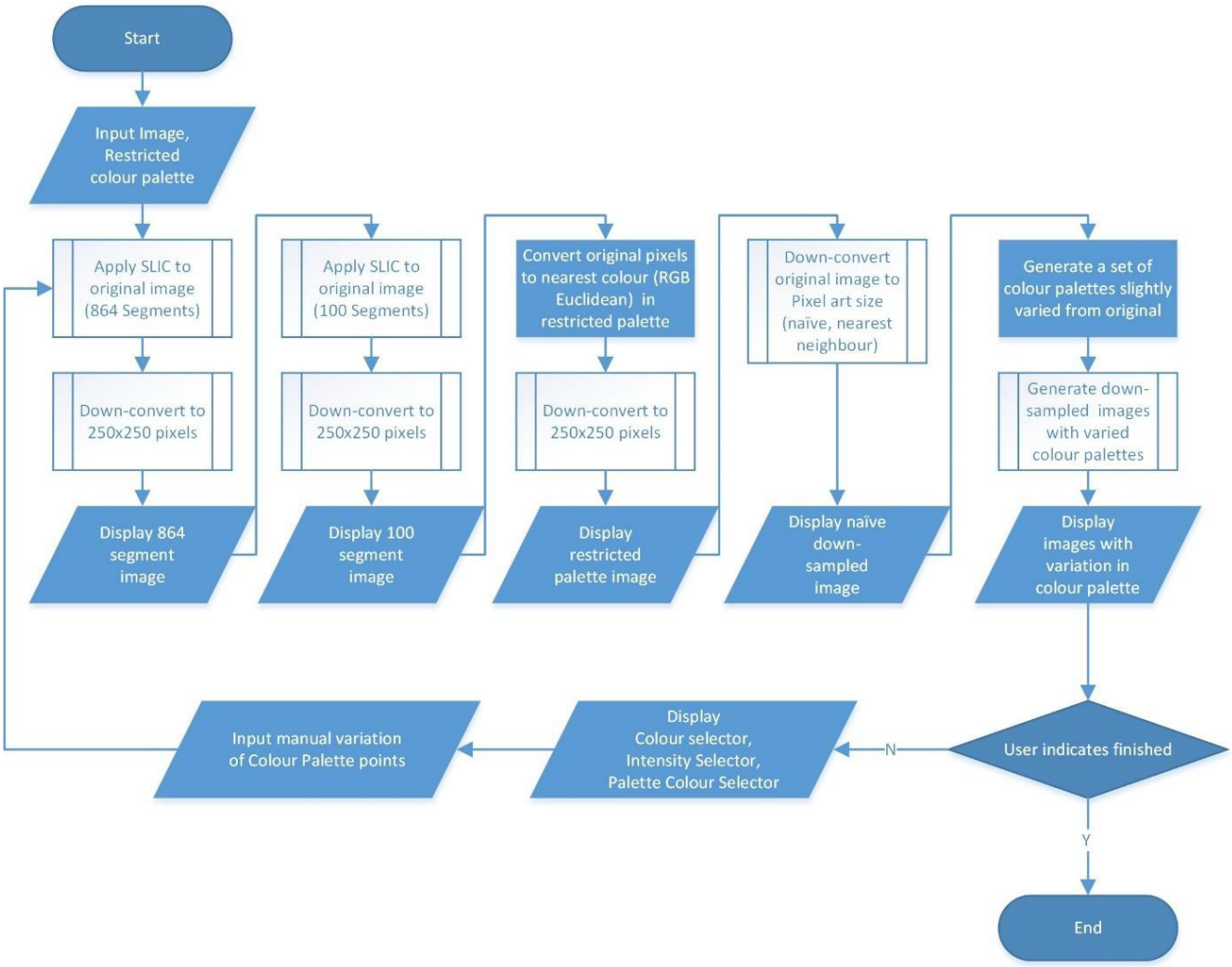


Fig.6 Palette Editing Process- Flow chart representation of the implementation of the palette editing screen.

6.4 GA Chromosome generation

Once the colour palette selections are confirmed a set of 49 GA chromosome are generated for use in the next section. The chromosomes are generated as follows:

A palette is generated for the chromosome

The palette generation process takes the palette defined in the previous step and varies each colour according to the standard deviation set by the user in the palette selection step. The variation is obtained using the original palette colour as a mean of a gaussian distribution with the standard deviation set by the user, a random set of RGB values is taken from this distribution and input as the colour in the palette. This is repeated until the colour palette has been fully generated for the chromosome. For the first 5 chromosomes generated the palette is not varied to ensure the selected palette values are well represented in the generated chromosome population.

The original image is subdivided into a smaller number of squares that will form the basis of the future image

The subdivision occurs based on a scale factor of the original image set by the user, for example a 1000x1000px image with a scale factor of 0.1 will result in a 100x100px pixel art image. Each low-resolution pixel is associated with a subdivision of the high-resolution image. In the case of the example each pixel in the 100x100 image is associated with a 10x10 subdivision of the 1000x1000 image.

A random pixel is selected in each subdivision to represent the pixel for that chromosome

For simplicity, and to achieve diversity in chromosome results, a random pixel is selected in each subdivision as the initial candidate pixel in the associated pixelart image

Selected colours are snapped to the nearest colour in the generated chromosome palette

For each subdivision the randomly chosen pixel colour will be snapped to the nearest chromosome colour palette colour. The image is now ready for display. An example of the first generation set of chromosomes is shown in fig.7 and the process is shown in flowchart form in fig.8

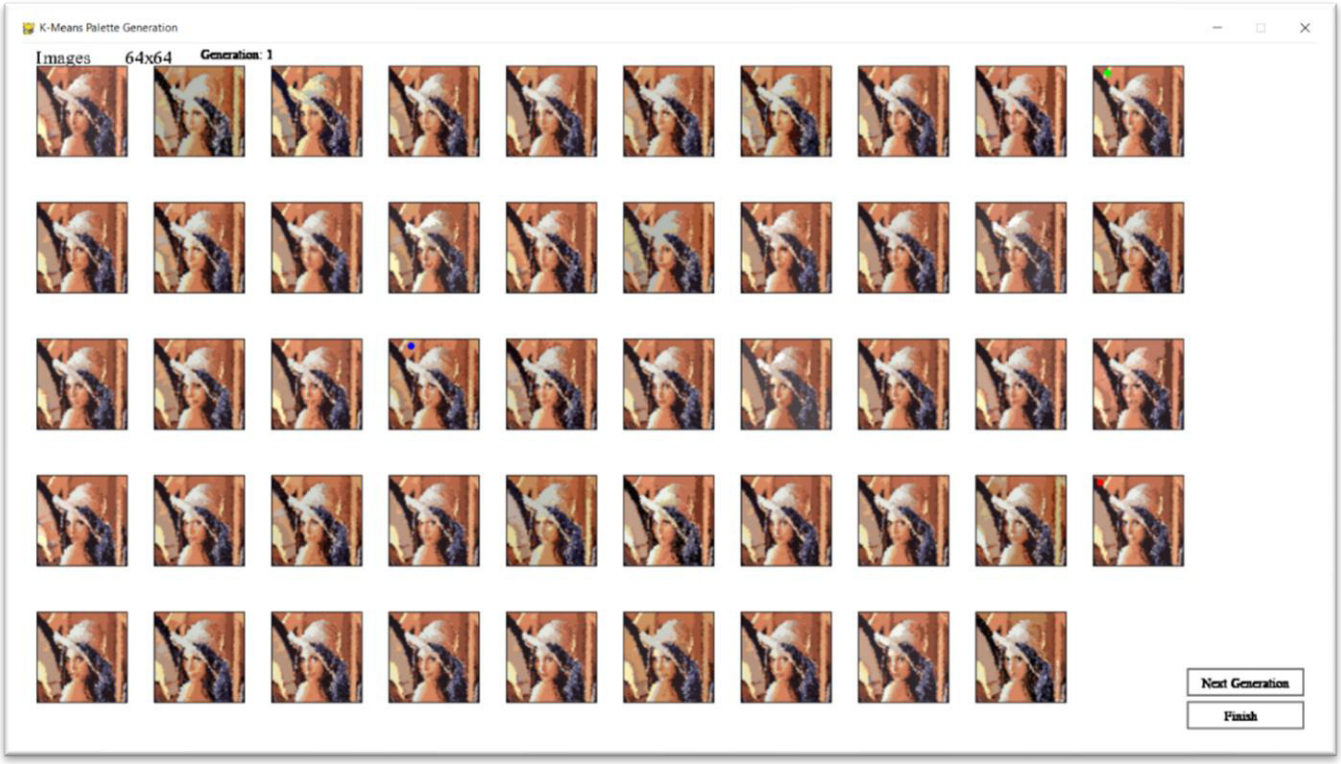


Fig.7 GA Refinement Process- Images on this screen are each generated by a GA chromosome, user selects the three best (marked by coloured dots in the top left corner) and clicks "Next Generation" to generate the next generation, or "Finish" to end the process.

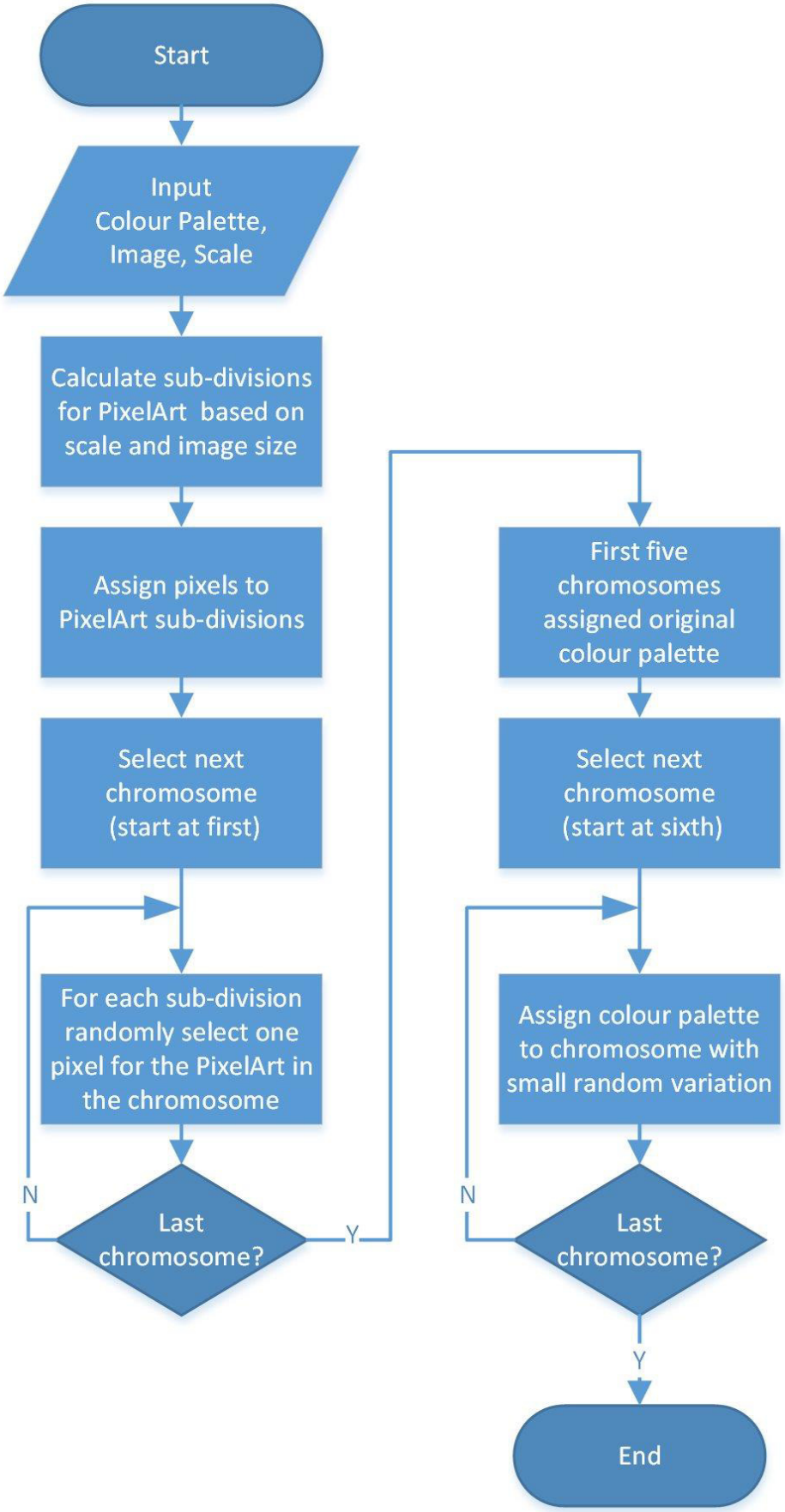


Fig.8 Chromosome Generation Process- Flow chart representation of the initial chromosome generation process

6.5 GA Refinement Process

6.5.1 Fitness calculation

The fitness calculation is based on user selected best images for each generation, with the user selecting the three best images each generation. Three images are selected as a user may wish to preserve more than one feature in an image, one image may have good eyes but poor ears whereas another may have good ears but bad eyes for example, using this methodology both desirable features can be guaranteed to be passed to the next generation.

With the three best images selected the current generation is split into thirds, excluding the three selected images. These thirds of the population are each compared for similarity to the three selected images, one third for each image. Similarity is measured by calculating the sum of the squares of the Euclidian distances between the RGB values of the pixels in the candidate image and the corresponding pixels in the selected best image. This measure defines the fitness of the chromosomes. Fig.7 shows how the most fit chromosomes are selected from the visualised population of chromosomes in a generation.

6.5.2 Competition

The method of selection chosen was a tournament selection with elitism. In this case this means that the best three chromosomes, as selected by the user, will always pass unchanged to the next generation, while the rest of the population will undergo tournament selection as described in 4.1 where for each member of the next generation population four random chromosomes are selected from the original population and the best chosen to go to the next generation. Subsequently pairs of the new population are chosen for crossover and some individual chromosomes chosen for mutation. The number of chromosomes involved in crossover or mutation is set in the program.

6.5.3 Crossover

The genetic algorithm used in this thesis uses a combination of uniform and two-point crossover. Based on the crossover fraction set, 7/10 was selected for this thesis, a fraction of the chromosomes in the fit population are paired with another chromosome in the fit population not including those selected for mutation or elitism. They then either undergo uniform or two-point crossover (at a ratio of 2:3).

In the uniform crossover every pixel in a chromosome is paired with the pixels assigned to the same location in the paired chromosome, each pixel has a 50% probability to swap positions with its paired pixel.

In two-point crossover two pixels are selected at random in one chromosome which represents a box on the equivalent image with these two pixels forming two of the corners. All pixels in this box are then swapped with all pixels in an identical area drawn on the paired chromosome. This was shown graphically in section 4.1 (fig.2).

In both cases the palette is then varied by uniform crossover and each chromosome pixel set is re-snapped to the new palette colours for the chromosome.

6.5.4 Mutation

The algorithm uses random mutation and a form of noise filtering mutation. In random mutation either the palette or a random pixel in the image is selected, once selected the value of the pixel or palette colour is changed by selecting a random number in a gaussian distribution around a mean which is the value of the original pixel/palette colour. The variation around this mean is applied to the colour in the palette closest to the original pixel/palette colour.

In the noise filtering mutation, a median filter is applied to the chromosome image to produce a reduced noise version of the chromosomes output. The median filter applied is the SciPy implementation of a median filter with a kernel size of 3 (meaning a 3x3 kernel footprint). This array is then crossed over with the original image using the same protocol for the uniform crossover described in 6.5.3. This is an attempt to remove noisy elements of the pixel art image whilst providing a chance to retain desirable elements that may be removed if the filter were applied uniformly, such as single pixels representing eyes in an otherwise uniformly coloured face. The refinement process is summarised in flowchart form in fig.9.

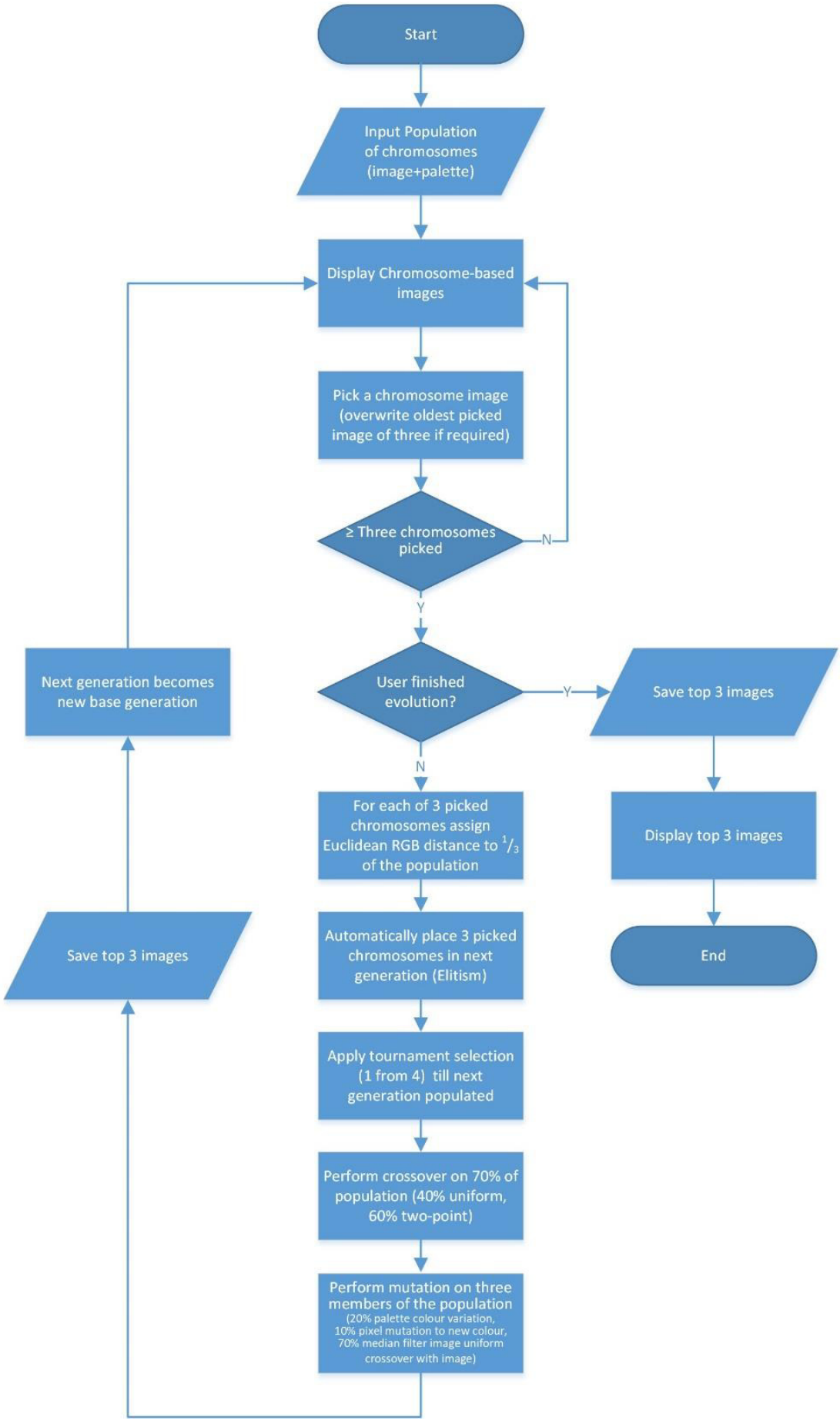


Fig.9 GA Refinement Process- Flow chart representation of GA refinement process.

7 Results and Analysis

This section outlines the results obtained by experimentation with the thesis code, including examples of the algorithms output compared against current research. Comparisons were made using the image specification laid out in Gerstner et al. [2] as this offered a good range of images to compare against as well as the ability to compare results to the work of pixel artists and naïve down-sampling techniques (cubic and nearest neighbour as described in Gerstner et al. [2]).

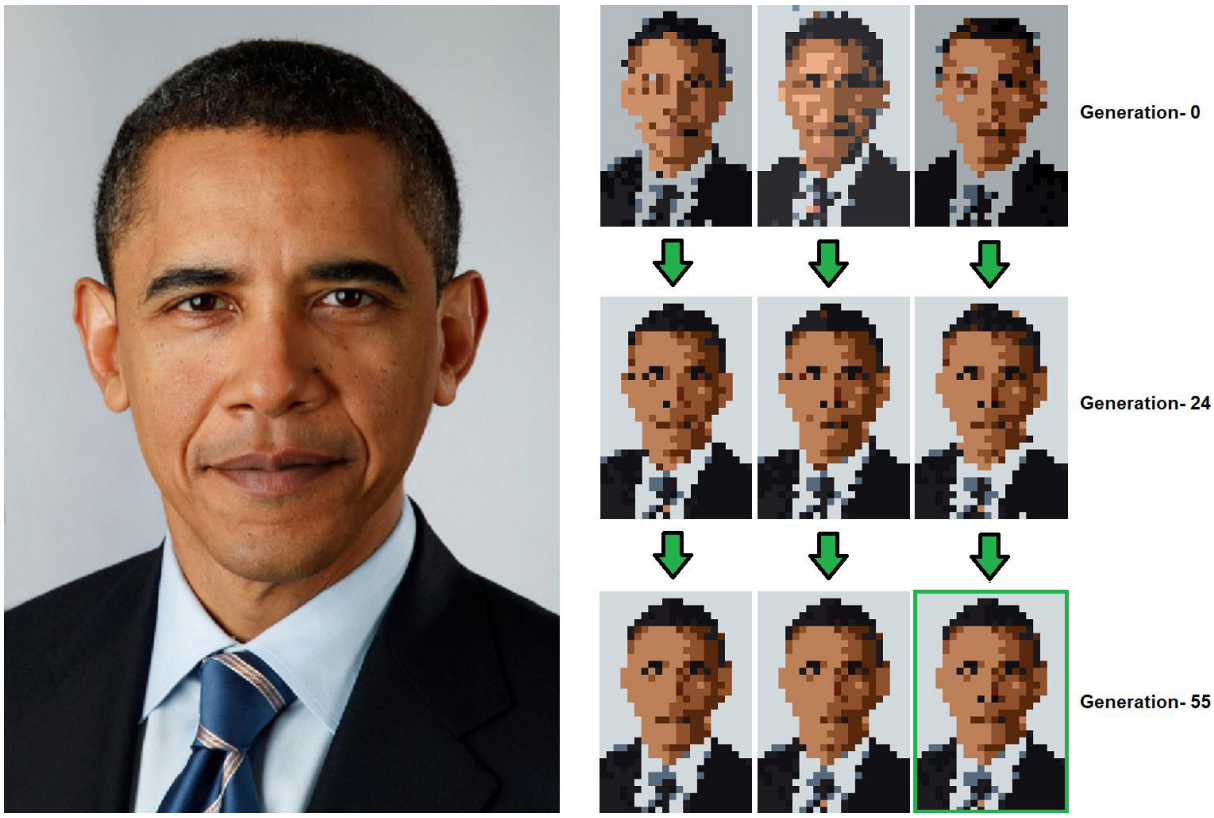


Fig.10 Obama Image- Original image 770x1120, Pixel art 22x32, Colour Palette Size 8

Fig.10 displays the iterative improvements of the Obama image over 36 generations, note that in each generation including the last 3 “best” images are selected out of a pool of 49, which are the three images displayed. Clear improvements can be seen over the generations. Direct comparison of the results is made in fig.11

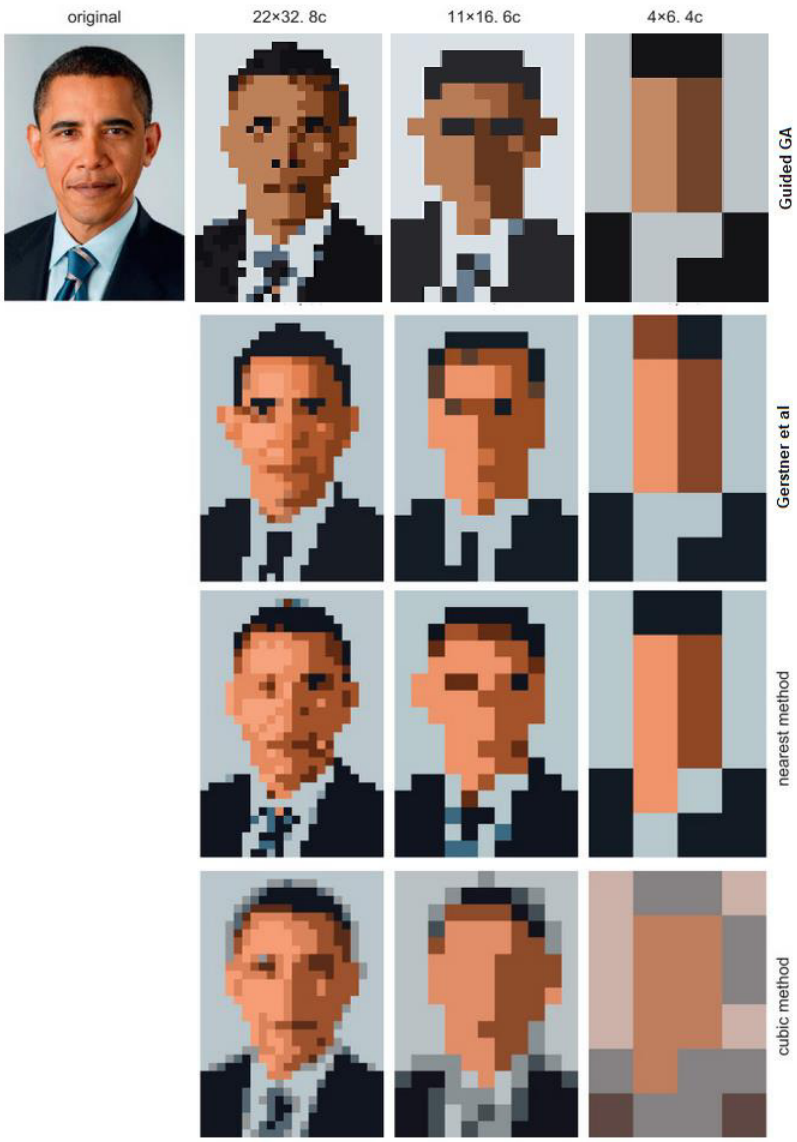


*Fig.11 Obama Image Comparison- Original image 770x1120, comparison images taken from Gerstn*er et al. [2]

When compared with naïve methods the guided GA shows clear advantages over both cubic and nearest methods particularly around the eyes and hair with less blurring and preservation of key features (eyes, nostrils). It is a closer comparison when looking at the method proposed by Gerstner, generally the guided GA results in a slightly noisier image particularly at the higher pixel resolutions around the tie. It is also quite interesting to note that the skin tone selected via the human guided GA has resulted in a darker skin tone. This is likely to be more representative of reality, if not the raw image data, as lighting when pictures are taken can often lighten skin tone which is not a fact that is considered by unguided pixelation techniques. The author notes that this has been an issue which has caused some controversy in recent times with automated imaging, particularly with regards to

facial recognition [36]. To demonstrate this point included in fig.12 is another picture of Obama in different lighting.

Fig.12 Obama Alternative Lighting- Edited from: "Barack Obama - Politics And Ascent To The Presidency". Encyclopedia Britannica, 2022, https://www.britannica.com/biography/Barack-Obama/Politics-and-ascent-to-the-presidency.

It is therefore noted that with user guidance corrections can be made when generating pixel art from reference images using information that is not present in the original image in order to cater to human perception. This is commented on in Gerstner et al when they reference pixel artist techniques such as dithering and edge highlighting. [2]

One failed case that was shown in Gerstner et al was that of a man wearing a backpack, however his face is strongly shadowed in the image. The guided GA was used to generate pixel art for this image and a comparison was made in fig.13.

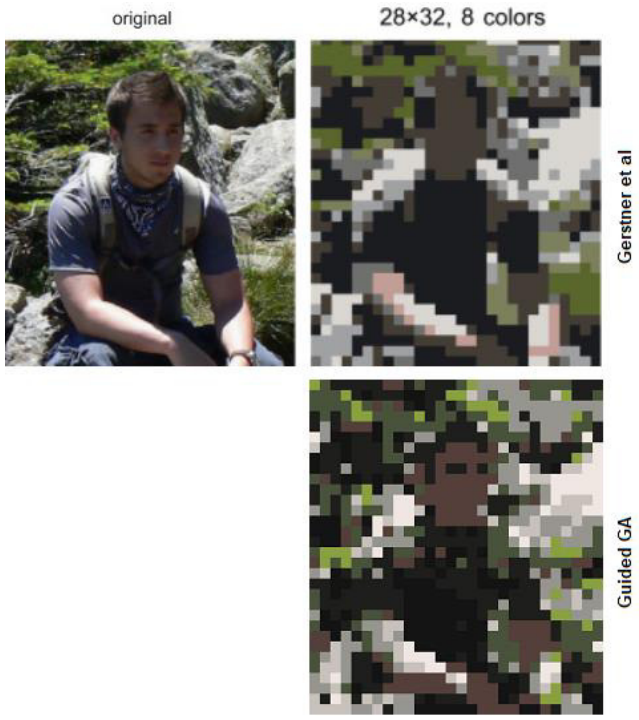


Fig.13 Backpack Failure Case- Original image 756x864, comparison image taken from Gerstner et al. [2]

As in Gerstner the guided GA struggles to deal with the shadowed nature of the detail in the image. Particularly, it is evident that the main focus of the image, which would be the man's face, struggles to compete with the background. Despite this however, the guided GA maintains important details that the automatic method does not (eyes, hair) although it struggles more when distinguishing the boundaries between the body and the background which gives a noisier appearance.

Another useful comparison to make is the results of the GA with the work of pixel artists. A comparison of this kind is made in fig.14.

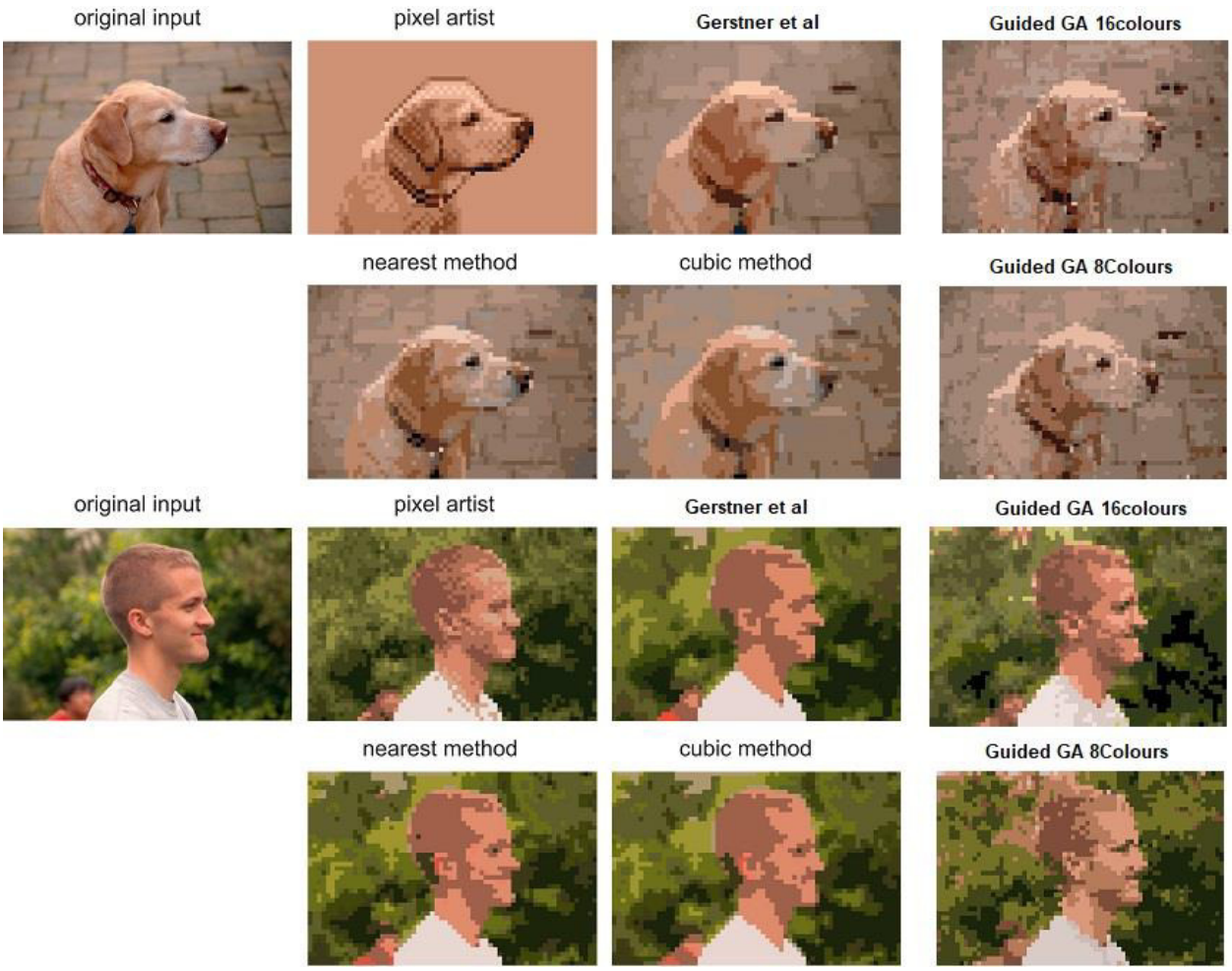


Fig.14 Comparison with Pixel Artists- Original images 832x559, 16 colour comparison images taken from Gerstner et al. [2]

It can be seen once again in fig.7 that the guided GA tends to produce slightly noisier results in larger images, however it is clearly better than naïve methods of pixel art generation. This is particularly evident in the case of the image of the man where the guided GA results are far closer to the pixel artist attempt than the attempt made in Gerstner, with more textural detail being available on the face. It should also be noted that both images can be represented with fewer colours than is represented in Gerstner which universally used 16 colour images as seen in fig.14. However, it is

recognised that without direct comparison images made with a more restricted colour palette the hypothesis that the guided GA works better with more restricted palettes due to the incorporation of human perception based on factors not in the original image, has not been proven. A further example of this is shown in fig.15.

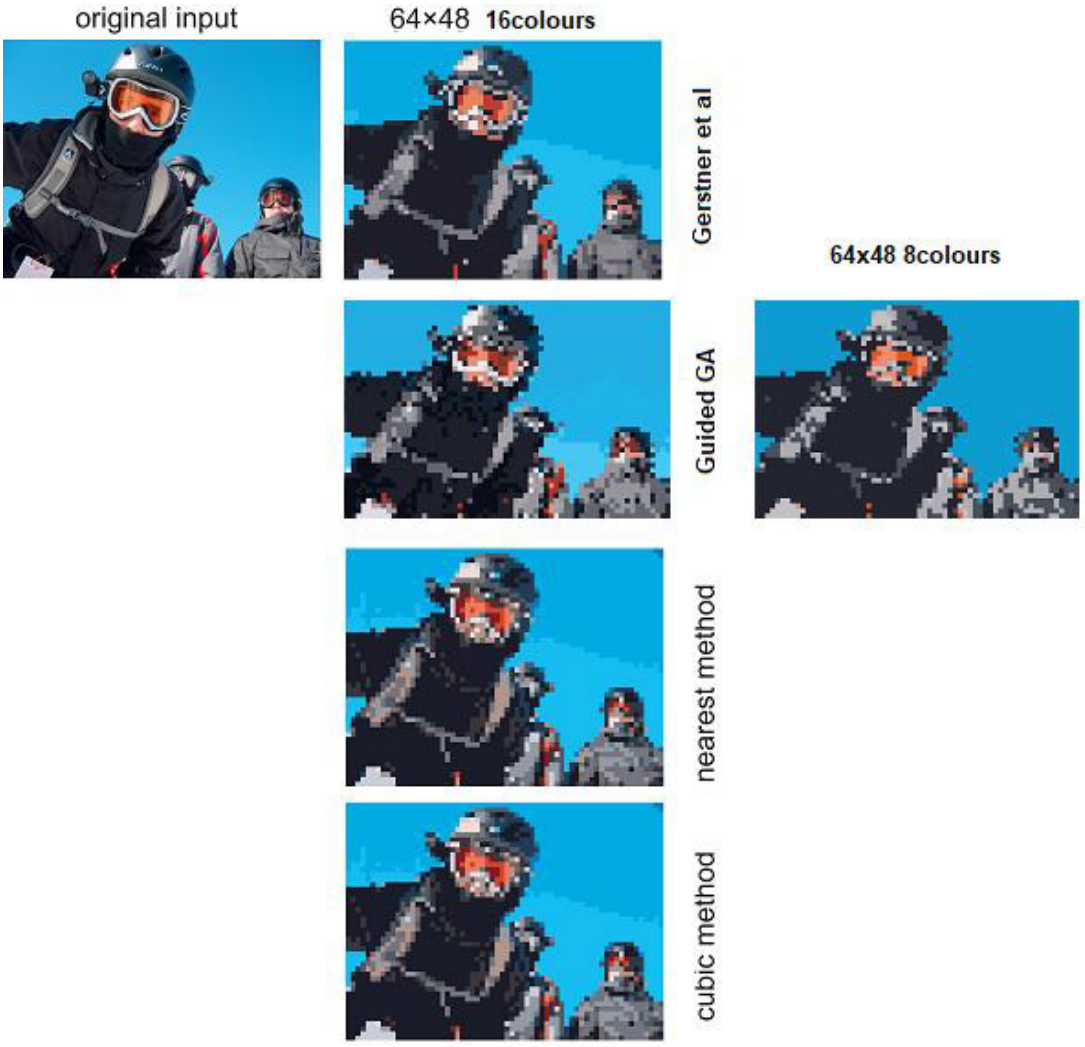


Fig.15 Colour Restriction Comparisons- Original image 768x516, comparison images taken from Gerstner et al. [2]

We can see that despite the continuing issues with mild increases in noise in the guided GA method when compared with Gerstner, the guided GA does tolerate more restricted colour palettes well.

By analysing the results from all the test cases that have been presented it can be seen that the guided GA method is capable of iteratively improving pixel art over multiple generations. When compared with current research this method produces results of comparable quality with both advantages (key feature preservation, colour palette curation) and drawbacks (general increase in noise). Despite this it must be noted that the human time investment is significant in the Guided GA method taking ~5mins to generate a pixel art image.

8 Conclusions, Reflections and Summary

8.1 Conclusion and Summary

This thesis has demonstrated that a human assisted genetic algorithm is capable of generating pixel art images of a quality comparable to current automated techniques. It has found there are some unique benefits for such an approach in the selection of colour palette and in important feature preservation. The main shortfall of the guided GA, beyond the human time investment required, is the tendency to create nosier images than other fully automated techniques used in current research.

It may be possible to address the shortfalls in the guided GA by implementing a more sophisticated set of noise reduction algorithms to deal with noisy pixels. The process might be further improved by integrating some of the current best practice automated pixel art generation techniques such as a gaussian or local threshold-based edge detection, or by refining the method by which the GA selects the initial colour of pixels away from a simple random selection (although this will require further experimentation to identify the best alternative). One approach may simply be to allow the human to 'mutate' one image each generation to snap several offending pixels to a different palette colour.

The issue with this approach is that in theory they could end-up doing the job of the pixel artist which makes objective assessment of the technique difficult. However, in practical terms this leaves the decision as to how much of the task to be left to the computer up to the human which would allow for rapid and effective interaction where the human sees an obvious correction to make. So as a tool such an approach would be entirely valid. Equally an edge detection and emphasising algorithm could be introduced as a form of mutation to increase the variety of potentially useful choices.

Another interesting direction the work could be taken in would be to attempt to develop a neural network that could identify a 'good' pixel art result which could then be used to guide the GA in place of a human, although granted the development of such a tool would be a worthy goal in its own right.

In summary, the thesis has fulfilled its aim to confirm the hypothesis that a genetic algorithm with a human in the loop is effective in guiding the automated generation of pixel art.

8.2 Reflections

The writing of this thesis has been a significant challenge and has given me some new insights about the applications of genetic algorithms. I chose to implement a GA in this thesis as I had used a similar approach for the control of a UAV swarm as part of my bachelor's thesis which gave me some familiarity with the approach, and I was interested to see if the methodology could be applied to a more diverse range of problems.

From my experience applying a GA in this thesis I have found that the approach is more of a framework than a set technique as it relies heavily on the fitness function, crossover, mutation, and the variables present in the chromosomes which can all vary greatly between problems. In this thesis in particular I developed an unusual form of mutation combining some elements of crossover and filtering for semi-random noise removal. This was significantly different to the standard random forms of mutation I have used previously as the mutation was more focussed on addressing a particular problem in the GA output. This might seem at odds with the purpose of mutation in a traditional GA (which is to introduce randomness into the population in search of more optimal solutions) however it was very effective in this particular application.

Another aspect of the work which was enlightening was the testing phase, I realised here that to truly assess the GA as a tool for others to use in the long term I would need to recruit volunteers to test the algorithm as humans in the loop or simply to assess output images. This would require ethical approval as well as improvements to the algorithm, so the interface is both easy to use and preferably higher speed to make it more acceptable to the average user. This highlighted to me how usability can be as important as the quality of the underlying algorithm when making a real-world tool and how care needs to be taken with human in the loop experiments to meet ethical guidelines.

I believe I have improved on my management of time when compared with projects I have undertaken in the past and have controlled the scope of the project well. In particular the decision to keep the palette colour selection process and down-sampling methods relatively simple by avoiding more complicated techniques such as edge detection was a good one. Had I not done this, I may have had significant issues with timing, and these were not necessary to address the main hypothesis.

I have found that I am reasonably competent in the python language and this project has given me opportunity to improve those skills. This dissertation has highlighted the importance of optimisation and particularly the advantages of using vectorisation in python to achieve this. Some of the image

arrays (up to 800x600 in size) would have been functionally impossible to manipulate otherwise as the time to complete processes would be prohibitively long for a real-time ‘human-in-the-loop’ tool.

I found it quite surprising to encounter a topical issue during the testing of my dissertation code such as that encountered when generating the pixel art of Obama. While pixel art generation algorithms not being able to recognise the skin colour of an image of Obama, due to imperfections in the lighting original image given to the algorithm, is not a particularly earthshattering observation; on reflection I believe it points to a larger issue that will face society in the future. It seems that as technology and algorithms develop societies will have to decide on how these algorithms are implemented and in what situations they are used. Potentially, human in the loop programmes will prove themselves useful simply because there is an individual capable of making ethical decisions or at least being responsible for the resultant effects of the algorithm.

9 References

[1] Elkheshen, Gamal Ahmed. "Pixel Art as a visual stimulus in graphic arts." Journal of Arts & Architecture Research Studies 2.3 (2021): 142-156.

[2] Gerstner, Timothy, et al. "Pixelated image abstraction with integrated user constraints." Computers & Graphics 37.5 (2013): 333-347.

[3] Holland, John H. "Genetic algorithms." Scientific American 267.1 (1992): 66-73.

[4] Kováč, Milan, and Peter Schreiber. "Identikit creation with the use of genetic algorithms." Applied Mechanics and Materials. Vol. 248. Trans Tech Pub. Ltd, 2013.

[5] Huang, Ming-Rong, and Ruen-Rone Lee. "Pixel Art Color Palette Synthesis." *Information Science and Applications*. Springer, Berlin, Heidelberg, 2015. 327-334.

[6] Takimoto, Hironori, Seiki Yoshimori, and Yasue Mitsukura. "Method for automatic generation of pixel art based on color-difference tolerance." *Kyokai Joho Imeji Zasshi/Journal of the Institute of Image Information and Television Engineers* 66.11 (2012): J399-J406.

[7] Farrell, P., 2019. Math adventures with Python: An illustrated guide to exploring math with code. No Starch Press.

[8] Staunton, Richard C. "The processing of hexagonally sampled images." Advances in imaging and electron physics 119 (2001): 191-265.

[9] Lyon, Richard F. "A brief history of 'pixel'." Digital Photography II. Vol. 6069. International Society for Optics and Photonics, 2006.

[10] Goldberg, Adele and Flegal, Robert, “ACM President's Letter: Pixel Art,” Commun. Assoc. Computing Machinery, Vol. 25, pp. 861–862, 1982.

[11] Grahn, Emma. "A study in GUI aesthetics for modern pixel art games." (2013).

[12] Heikkinen, Olli. "Hi-bit pixel graphics: new era of pixel art." (2021).

[13] Kry, Tobias. "How to develop graphic design for games with low-pixel density." (2013).

[14] Kopf, Johannes, Ariel Shamir, and Pieter Peers. "Content-adaptive image downscaling." ACM Transactions on Graphics 32.6 (2013): 1-8.

[15] Han, Chu, et al. "Deep unsupervised pixelization." ACM Transactions on Graphics (TOG) 37.6 (2018): 1-11.

[16] Oeztireli, A. Cengiz, and Markus Gross. "Perceptually based downscaling of images." ACM Transactions on Graphics 34.4 (2015):1-10.

[17] Weber, Nicolas, et al. "Rapid, detail-preserving image downscaling." ACM Transactions on Graphics (TOG) 35.6 (2016): 1-6.

[18] Gastal, Eduardo, “Spectral Remapping for Image Downscaling.” ACM Transactions on Graphics (TOG) 36.4 (2017): 145:1-16.

[19] Ma, Ruihua, and Gurminder Singh. "Large-scale infographic image downsizing." 2004 International Conference on Image Processing, 2004. ICIP'04.. Vol. 3. 2004.

[20] Gerstner, Timothy, et al. "Pixelated image abstraction." NPAR@ Expressive. 2012.

[21] Basu, Mitra. "Gaussian-based edge-detection methods-a survey." IEEE Transactions on Systems, Man, and Cybernetics, Part C 32.3 (2002): 252-260.

[22] Hildreth, Ellen C. "Edge detection." (1985).

[23] Ahmad, Muhammad Bilal, and Tae-Sun Choi. "Local threshold and boolean function based edge detection." IEEE Transactions on Consumer Electronics 45.3 (1999): 674-679.

[24] Bezdek, James C. et al. "A geometric approach to edge detection." IEEE Transactions on Fuzzy Systems 6.1 (1998): 52-75.

[25] Canny, John. "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence 6 (1986): 679-698.

[26] Tirunelveli, Girish, Richard Gordon, and Stephen Pistorius. "Comparison of square-pixel and hexagonal-pixel resolution in image processing." IEEE CCECE2002. Canadian Conference on Electrical and Computer Engineering. Conference Proceedings (Cat. No. 02CH37373). Vol. 2. IEEE, 2002.

[27] Fadaei, Sadegh, and Abdolreza Rashno. "A Framework for Hexagonal Image Processing Using Hexagonal Pixel-Perfect Approximations in Subpixel Resolution." IEEE Transactions on Image Processing 30 (2021): 4555-4570.

[28] He, Xiangjian, and Wenjing Jia. "Hexagonal structure for intelligent vision." 2005 International Conference on Information and Communication Technologies. IEEE, 2005.

[29] Fitz, A. P., and R. J. Green. "Fingerprint classification using a hexagonal fast Fourier transform." Pattern recognition 29.10 (1996): 1587-1597.

[30] Huang, Shu-Chien. "An efficient palette generation method for color image quantization." Applied Sciences 11.3 (2021): 1043.

[31] Achanta, Radhakrishna, et al. "SLIC superpixels compared to state-of-the-art superpixel methods." IEEE transactions on pattern analysis and machine intelligence 34.11 (2012): 2274-2282.

[32] Goldberg, D.E., 1989. Genetic algorithms in search. Optimization, and Machine Learning. Addison Wesley

[33] Youlian Zhu, Cheng Huang, “An Improved Median Filtering Algorithm for Image Noise Reduction”, Physics Procedia, Volume 25, 2012, pp. 609-616

[34] Buades A., Coll B., Morel J-M. “A review of image denoising algorithms, with a new one.”, Multiscale Modeling and Simulation”, SIAM, 2005, 4 (2), pp.490-530.

[35] Van der Walt, Stefan, et al. "scikit-image: image processing in Python." PeerJ 2 (2014): e453.

[36] Nast, Condé. "The Best Algorithms Still Struggle To Recognize Black Faces". *WIRED*, 2022, https://www.wired.com/story/best-algorithms-struggle-recognize-black-faces-equally/.