Computing for Graphics
Programming Project Report

i7208422
Sim Chun You

Project Option 4: Generation of Roy-Lichtenstein-like images from Photographs

Contents

Introduction of the project.
Generation of Roy-Lichtenstein-like images from photographs

What the program is supposed to do
Roy Lichtenstein was an American pop artist famous for his boldly-coloured parodies of comic strips and advertisements. (The Biography.com 2015) His works usually consists of a combination of outline art, mix with flat and bold colours and Ben Day dots.

My program was to read one or more images and produce a resulting image similar to the works of Roy Lichtenstein.

Approach to solving the project
My approach to solving this is to firstly duplicate the image and work on them separately to achieve the various aspects of Roy Lichtenstein's iconic art before combining them all together.

Reducing of Colour Palette
Using one of the image duplicated, I would analyse the pixels of that image and proceed to use a colour quantization algorithm to get a reduced colour palette. For my project itself, I have chosen a colour palette of 16. With regards to the algorithm, I have chosen to try using the Median Cut algorithm for colour quantization. This is because it was the better algorithm that allows for a reduction of image colours while minimizing the impact of its appearance whereas other algorithms such as the Uniform Quantization fails to capture the distribution of colours and the Populosity Algorithm ignores rare colours.

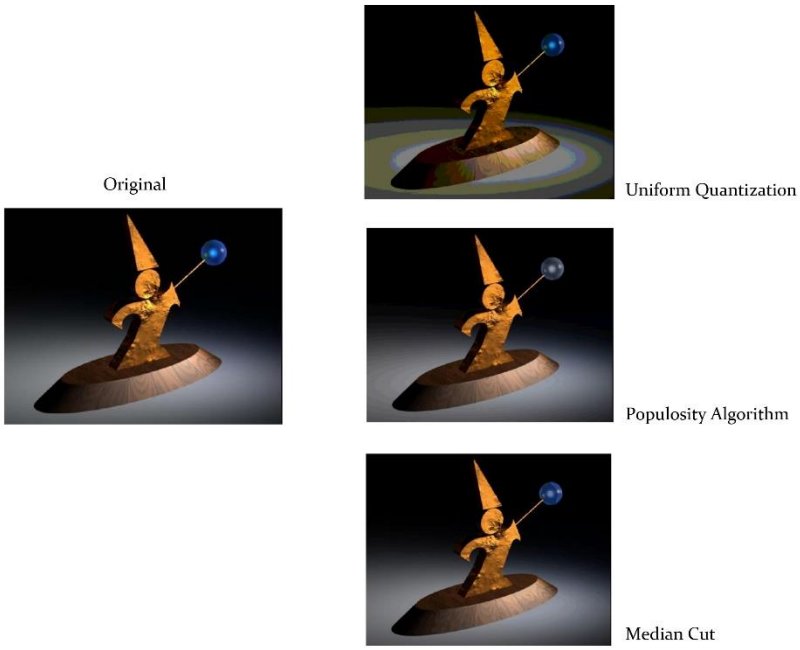


Image taken and Modified from University of Wisconsin (2002)

After getting the reduced colour palette, I went on to assign the image pixels with the newly created reduced colour palette based on the Euclidean distance between the current colour the pixel possess and each colour in the palette.

I would later replace colours that are closer to certain colours like red, blue, yellow, black and white with solid flat colours and the rest of the colours not assigned to the above mentioned, a Ben Day dots template.

Edge Detection

I would take 2 other duplicated images and proceed to do an edge detection method on them. The method that I used was the algorithm that was taught during our lectures which is the difference of Gaussian Blur method. (Eike Falk Anderson 2015). This involves me having to do a light convolution blur with a 3x3 kernel on one of the duplicated image and a heavy convolution blur with a 5x5 kernel on the other duplicated image. I would then subtract the light convolution blur from the heavy convolution blur to get the edges of the image.

Combination of Ben Day image and Edge Detection

With the surfaces that shows the Ben Day image and the Edge Detection, I would proceed to combine them using either of 2 methods. The first method is the replacement method which goes through the Ben Day image and the Edge Detected image and replace the pixel of the Ben Day image with black whenever the Edge Detected image has the corresponding pixel black. The other method is the multiply method whereby I take the pixels of the Edge Detected image and multiply them onto the Ben Day image.

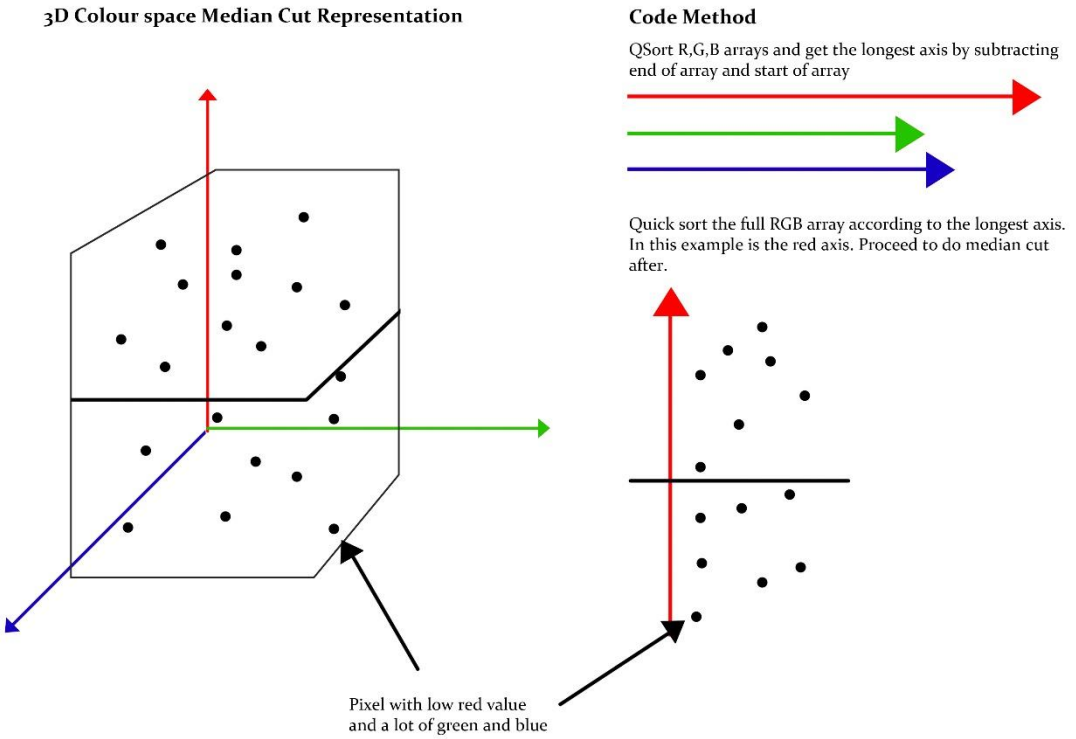
Implementation of Algorithms

Colour Quantization using the Median Cut Algorithm

Looking at the steps of the Median Cut Algorithm, the colours are represented in a 3D colour space. The longest axis is later determined and the colour space undergoes a median cut along that axis, creating 2 smaller colour spaces. This action is performed until we have the same number of colour space as the number of colour palette we need. As I cannot project the colours in 3D space as easily, I approach the Median Cut Algorithm one step at a time, looking at it in a 2D point of view.

To approach the Median Cut Algorithm, I analyse and get the RGB values of the image and put them in a 2D array called colour. I then check how many elements a colour space should have before it returns the reduced colour palette. This is done by getting the MaxElementCount = (total pixels) / (colour palette number). If the (total pixels) % (colour palette number) is > 0, I would increase the MaxElementCount by 1. I would also create a 2D array called Colour Palette that stores this newly created reduced colour palette.

I created a function for the Median Cut Algorithm. The function requires several parameters, such of which is a start and an end value. I create 3 arrays, R, G and B which stores all the red, green and blue values between colour[start] and colour[end]. After that, I quick sort the R, G and B arrays using the C library qsort function and get the last index of the array and subtract it with the first to get the longest axis. I proceed on to quicksort the colour array according to the longest axis so that I could 'median cut' the array along that axis. This is done by using a direct recursion of the function with the first recursion having the start as start and the end as (((end-start)/2) + start) or ((end+start)/2) and the next recursion as start being (((end-start)/2 + start) or ((end+start)/2) and end as end.

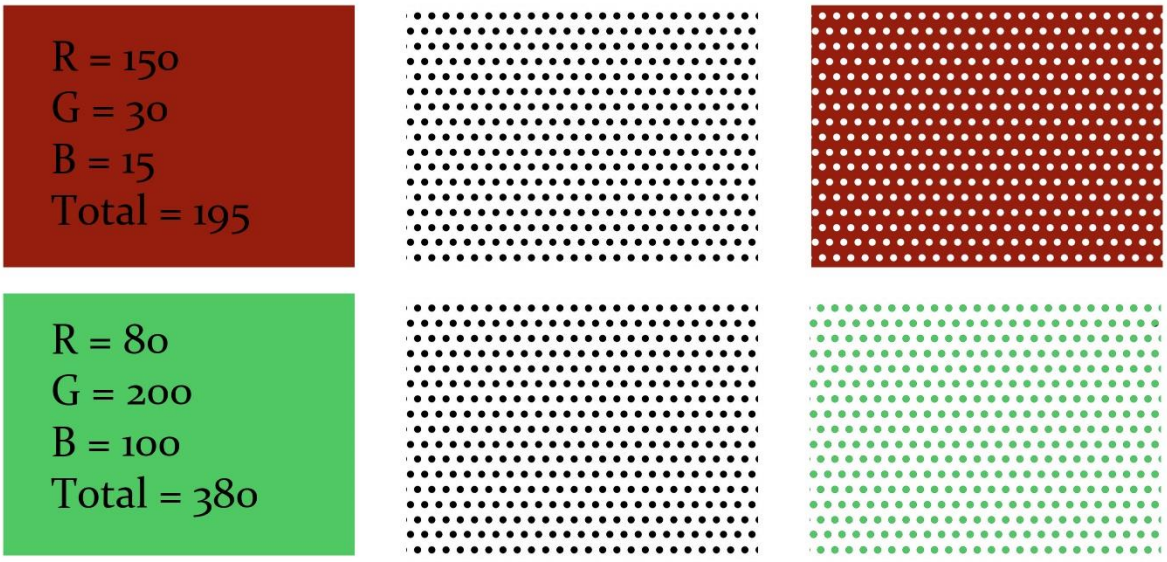


The recursive function will continue until we have as many colour spaces as we need. This is determined by whether the (end-start) < MaxElementCount.

After getting the reduced colour palette, which in my program it is a colour palette of 16. I went back to look through each pixel RGB values of the image and assign it with the new colour palette according to the smallest Euclidean Distance between its colour value and each of the colour palette colours

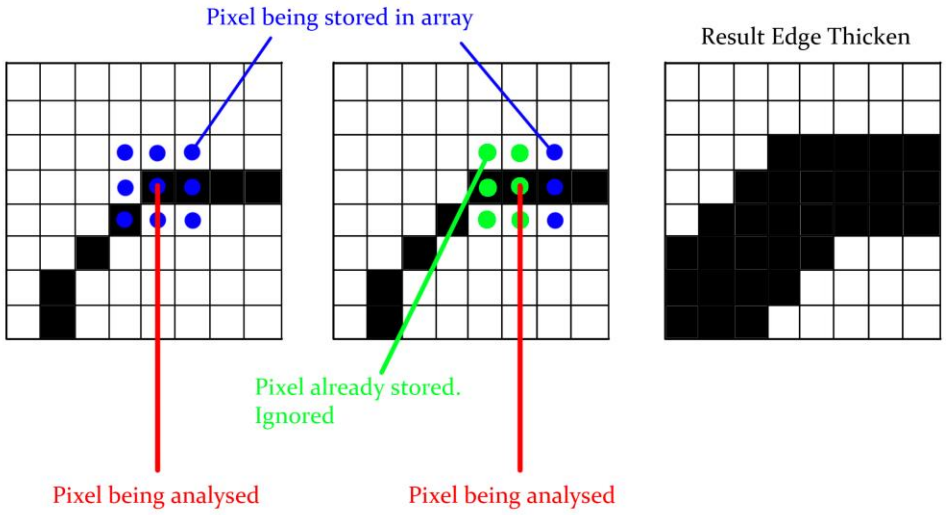
Ben Day replacement method

With the colour quantized image with a palette of 16, I then look through each pixel colour of the image and for those that are close to a certain value of red, blue, yellow, black and white, I will assign that pixel to a pre-determined red, blue, yellow, black and white colour. For the remaining colours not being assigned to any of the pre-determined palette, I went to give them a Ben Day look. To allow a varied Ben Day template variation, I chose not to use a coloured Ben Day dots template but a black and white one. The colours will be set according to the quantized colours the pixel possess. Looking through each pixel as well as the corresponding pixel in the Ben Day template, if the quantized colour has a total that is less than 200, which means it is either darker or more saturated, for each corresponding pixel in the Ben Day template that is white, I would assign it with the quantized colour. For each corresponding pixel in the Ben Day template that is black, I would assign it with white. If the quantized colour has a total value that is more than 200, I would assign areas where the Ben Day dots template that is white, white, and areas that the Ben Day dots template black, the colour quantized colour.



Edge Detection
The next thing I looked into was the detection of the edges of the image. I created 2 other surfaces using the original image pixels and proceed to colour quantize the image using the above algorithm to a colour palette of 2. This allows for a clearer and more defined edge detection. I continued to grayscale both images and did a light convolution blurring to one surface with a 3x3 kernel and a heavy convolution blurring to the other surface with a 5x5 kernel. I then subtracted the pixels of the surface with the light convolution blurring from the surface with the heavy convolution blurring which results in an image with the outlines of the image.

As the outlines that resulted is a little thin, I went on to try to thicken the edge detection. I created an array that was to store the pixel indexes that will be used to thicken the edges. I then loop through each pixel in the edge detection and for every black pixel that it comes across, it will store the surrounding pixel indexes based on a 3x3 kernel in the array. However if that index has already existed in the array, I would skip that index. I then went through each pixel of the image again and change the pixels with matching pixel indexes in the array with black.

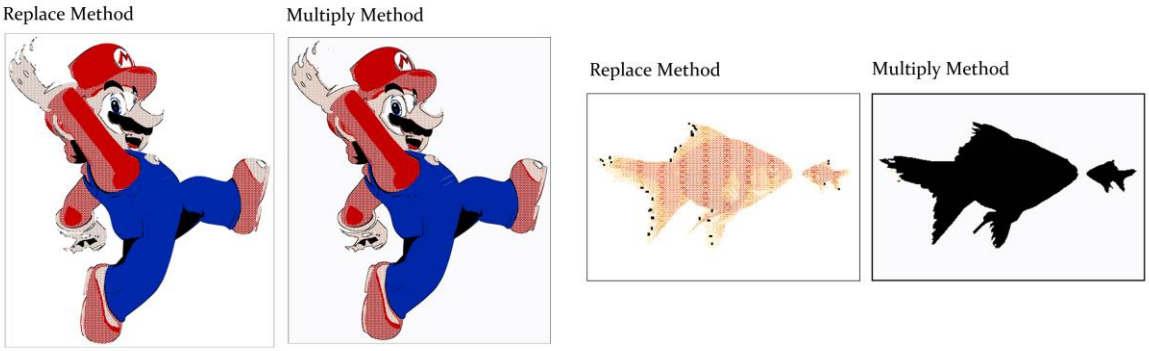


Combining of Edge Detection and Ben Day image
With the resulting Ben Day image and the Edge Detected image, I combined them while trying 2 different formulas.

The first formula is a simple replacement method. I would loop through every pixel in the surfaces and for each pixel index in the Edge Detected image that is black, I would replace the exact same pixel index in the Ben Day image with black.

The other method that I tried was the multiply method. I first typecast the RGB values of the Edge Detected image to float and divide it with 255. I then multiply the resulting value with

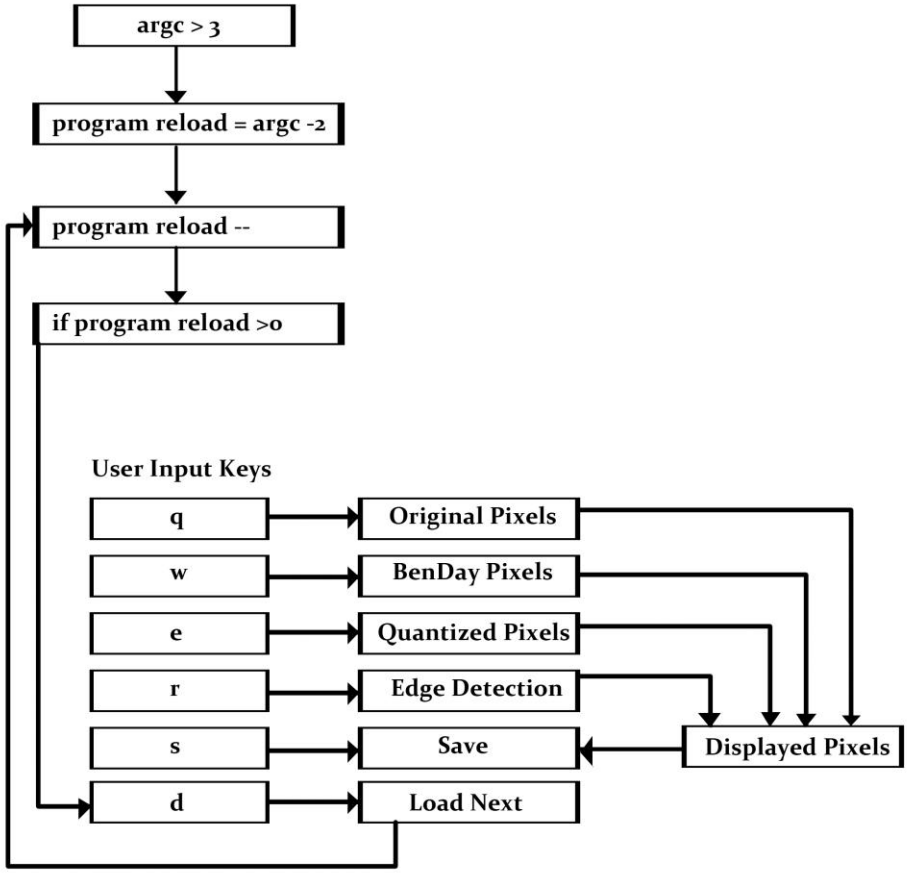
the float typecast of the RGB values of the Quantized image to get the new multiplied pixels. I then assign the Quantized pixels with the new value.

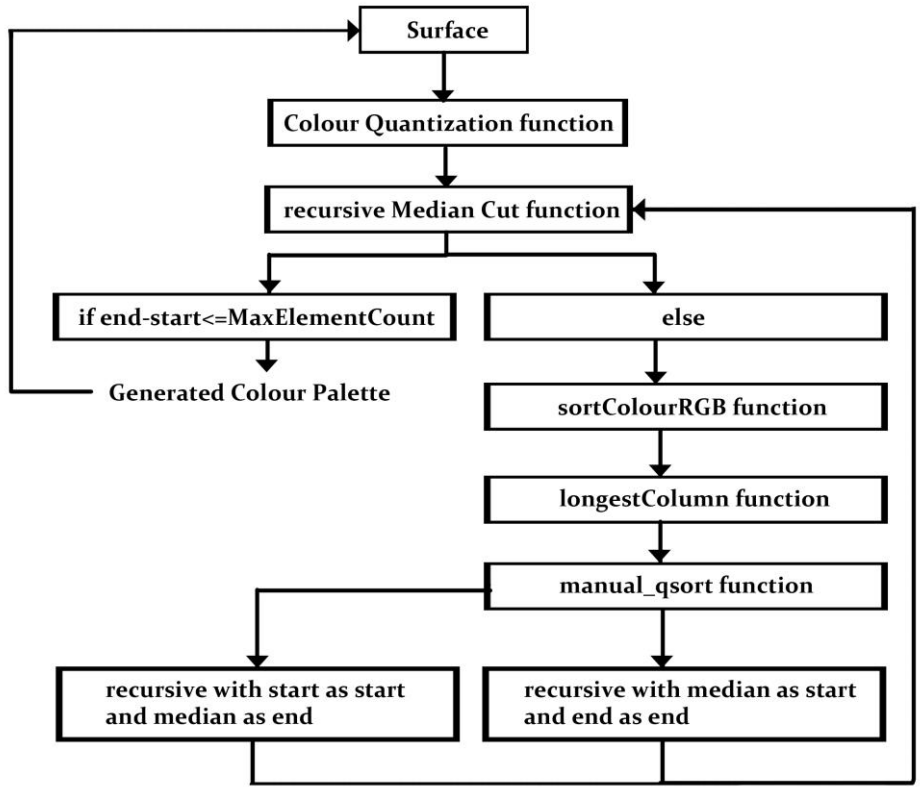


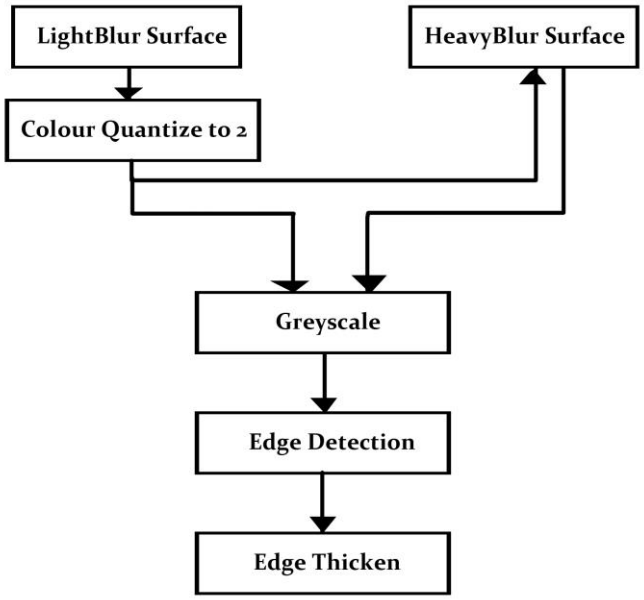
The problem I faced was that the edges that was detected for certain images like the fish picture (418x287 pixels) is completely black inside the fish. However for other images it works fine. If I use the replace method, as the pixels inside the fish is not completely black, the combination would still work. However if I were to use the multiply method, although it gives a better result for most of the pictures, it does not work for some images with a resulting edge detection like the fish's. For the sake of a function able program, I chose to use the replace method.

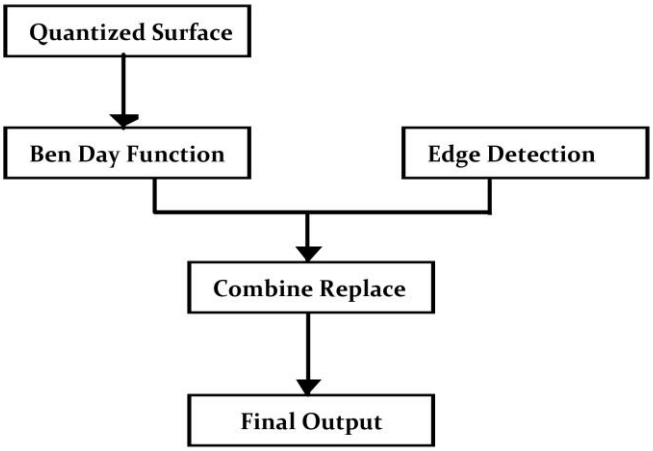
Flow of control and program outline

Throughout my program, I created a couple of surfaces each storing information of the various processes of image manipulation. This in turn can be displayed in my final output and saved through converting the displayed pixels to that of the surface pixels with the manipulated pixel information. The rough layout of my program is as shown below.

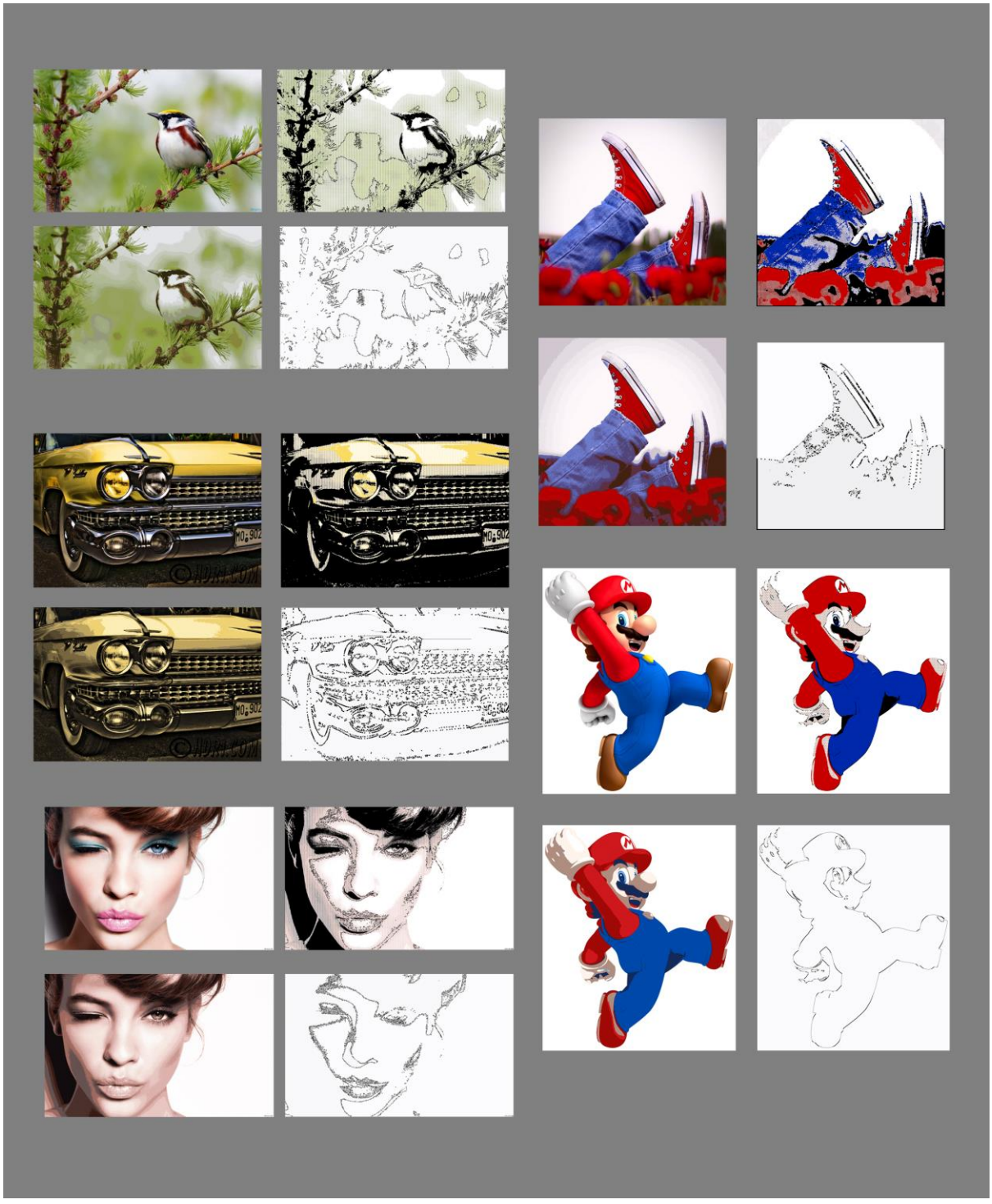



Surface
Colour Quantization function
recursive Median Cut function
if end-start<=MaxElementCount
else
Generated Colour Palette
sortColourRGB function
longestColumn function
manual_qsort function
recursive with start as start and median as end
recursive with median as start and end as end


LightBlur Surface
HeavyBlur Surface
Colour Quantize to 2
Greyscale
Edge Detection
Edge Thicken


Quantized Surface
Ben Day Function
Edge Detection
Combine Replace
Final Output

Results



Bibliography

Anderson, E,F.,2015. *Computing for Graphics: Fundamental Algorithms*[online]. NCCA. Available from: https://mybu.bournemouth.ac.uk/bbcswebdav/pid-1130330-dt-content-rid-2466177_2/courses/061019/CFGlecture18print.pdf [Accessed 04 April 2015].

Anderson, E,F.,2015. *Computing for Graphics: Algorithms cont & Computer Graphics Algorithms*[online]. NCCA. Available from: https://mybu.bournemouth.ac.uk/bbcswebdav/pid-1132052-dt-content-rid-2480584_2/courses/061019/CFGlecture19print.pdf [Accessed 04 April 2015].

Anderson, E,F.,2015. *Computing for Graphics: C/C++ & Software Projects*[online]. NCCA. Available from: https://mybu.bournemouth.ac.uk/bbcswebdav/pid-1116453-dt-content-rid-2070060_2/courses/061019/CFGlecture12print.pdf [Accessed 09 May 2015]

Bloomberg,D,S.,No Date. *Color quantization using modified median cut* [online]. Available from: http://www.leptonica.com/papers/mediancut.pdf [Accessed 03 April 2015]

Clark,D.,No Date. *The Popularity Algorithm* [online.] Available from: https://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCEQFjAA&url=http%3A%2F%2Fdcis.uohyd.ernet.in%2F~mravi%2Fdownloads%2FCIP%2FOLD-CIP%2FThe%2520Popularity%2520Algorithm.doc&ei=RHRNVbyNCMfuUt-WgcgG&usg=AFQjCNHs0jYCkOANTXnLAgAbP31PmPYjUw [Accessed 03 April 2015]

D'Agostino,D.,2014. *SDL2: Converting an Image to Greyscale*[online]. Programmer's Ranch. Available from: http://www.programmersranch.com/2014/02/sdl2-converting-image-to-grayscale.html [Accessed 03 April 2015]

University of Wisconsin, 2002. *cs559-5.* Madison: University of Wisconsin.

Kruger,A., 1994. *Median-Cut Colour Quantization: Fitting true-color images onto VGA displays* [online]. Dr.Dobb's – the world of software development. Available from: http://www.drdobbs.com/database/median-cut-color-quantization/184409309?pgno=1 [Accessed 03 April 2015]

Lazy Foo' Productions, No Date. *Optimized Surface Loading and Soft Stretching* [online]. Lazy Foo' Productions. Available from: http://lazyfoo.net/tutorials/SDL/05_optimized_surface_loading_and_soft_stretching/index.php [Accessed 08 April 2015].

The Biography.com website, 2015. *Roy Lichtenstein* [online]. The Biography.com website. Available from: http://www.biography.com/people/roy-lichtenstein-9381678 [Accessed 01 May 2015].

Thyssen,A.,2006. *ImageMagick v6 Examples – Color Quantization and Dithering* [online]. Imagemagick.org. Available from: http://www.imagemagick.org/Usage/quantize/#posterize [Accessed 03 April 2015]