3D Colour Histograms

Callum McGregor

# Abstract

Abstract body.

I certify that all material in this dissertation that is not my own work has been identified:

April 2016

Table of Contents

Abstract 1

1 Introduction 4

1.1 Project Statement and Motivation 4

2 Summary of Literature Review 4

2.1 Colour Spaces 4

2.2 Colour Histograms 5

2.3 Technologies 6

2.4 Colour Quantisation 6

2.5 Colour Adjustments 6

3 Summary of Specification and Design 7

3.1 Project Introduction 7

3.2 System Architecture 8

3.3 Component Design 8

3.4 UI Design 11

3.5 Testing and Evaluation 12

3.6 Time Plan 12

4 Development and Testing 13

5 Testing 17

6 Description of the Final Product 17

6.1 System Architecture 17

6.2 Component Design 17

6.2.1 View 18

6.2.2 Model 18

6.2.3 Controller 20

6.3 User Interface 20

6.4 Product Behaviour and Performance 21

7 Evaluation of the Final Product 22

7.1 Introduction 22

7.2 Requirements 22

7.3 System Architecture 22

7.4 Component Design 23

7.5 User Interface 25

7.6 Product Behaviour and Performance 25

7.7 User Testing Error! Bookmark not defined.

8 Critical Assessment of the Project as a Whole 28

8.1 Pre-Development 28

8.2 Development Cycle 29

8.3 Time Plan 30

8.4 Testing 29

8.5 Conclusion 30

9 Future Work 30

10 Conclusion 31

Bibliography 32

Appendices 35

# Introduction

# Summary of Literature Review

The literature review analysed and drew conclusions from sources gathered during research for the project. Five key areas emerged: colour spaces, colour histograms, technologies, colour quantisation, and colour alterations. This section summarises the literature review, introducing each topic and stating the conclusions drawn with the aim of providing the necessary knowledge for this report.

## Colour Spaces

Colour spaces are a method of representing colour in a coordinate system. Research into this area was important as the colour spaces in which to display the histogram had to be chosen and as each has it’s own benefits and drawbacks. It was also necessary to transform between colour spaces, so research into colour space transforms was done.

From Andrew Steer’s introduction to colour science (2008) and Phil Cruse’s overview of the CIE-L\*a\*b\* space (2015) it was concluded that sRGB and CIE-L\*a\*b\* would be suitable colour spaces in which to display the histogram: sRGB because it is a universal standard and easily understand, and CIE-L\*a\*b\* because it better preserves perceived uniformity of colour difference. Hoffmann’s papers on the CIE-XYZ and L\*a\*b\* colour spaces (CIE Color Space, 2015) (CIELab Color Space, 2015) also helped draw this conclusion but her papers also provided information on colour space transformations; alongside Ryan Juckett’s (2010) and Irotek’s (2014) websites, Hoffmann’s papers provided the understanding and formulae necessary to transform colours between the sRGB, CIE-XYZ, CIE-L\*a\*b\*, and HSL colour spaces (CIE-XYZ is used as an intermediate colour space when transforming between sRGB and CIE-L\*a\*b\*).

## Colour Histograms

The 2D colour histogram of an image plots light levels against the number of pixels with that light level. Luminosity histograms plot the absolute lightness of each pixel, while colour histograms plot each of the RGB colour channels. 3D histograms plot the distribution of colours in 3 dimensions: axes represent primary colours in the chosen colour space and frequency is represented by the size of plots.

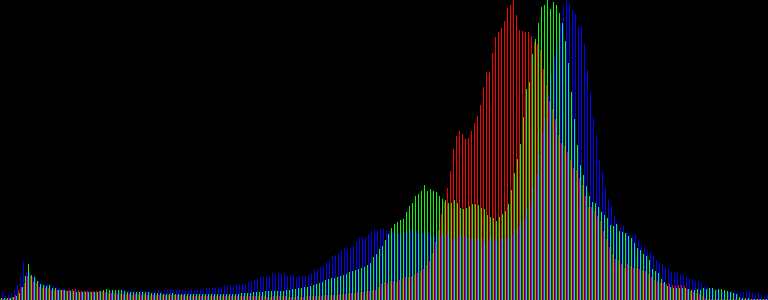


Figure : 2D Luminosity and Colour Histogram

(https://upload.wikimedia.org/wikipedia/commons/f/f5/Odd-eyed\_cat\_histogram.png)

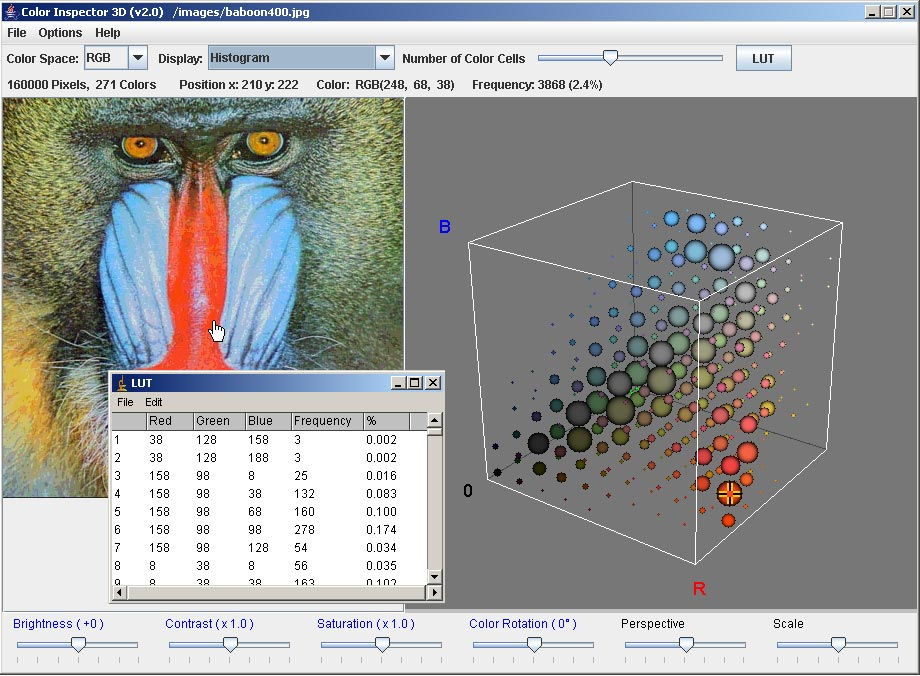


Figure : 3D Colour Histogram

(http://rsb.info.nih.gov/ij/plugins/images/3d-inspector.jpg)

Most sources were on 2D histograms and their applications in photography, such as Practical Photography Tips (PracticalPhotographyTips) and the Cambridge in Colour webpages (McHugh, 2015). These sources helped build an understanding of histograms but were of limited value. Sources on 3D histograms were other implementations, including Villarroal (2013) and Third Avenue Design’s (Third Ave Design) web pages, Caballero & Belmonte’s real-time video histogram (Caballero & Belmonte), and Barthel’s Java application (2007). Through inspection of these sources the best features were compiled and incorporated into this project.

## Technologies

Prospective technologies for the project were also researched. Previous experience gave a bias towards WebGL, a JavaScript rendering library based on OpenGL, augmented with another library, Three.js. Research into these frameworks was done to confirm that they were appropriate.

From a comparison of the Khoronos Group’s and Parisi’s WebGL tutorials (Khoronos Group, 2015) (Parisi, 2010) and Three.js tutorials (mrdoob, 2015), the conclusion was made that while WebGL is well established the code is verbose compare to Three.js, which wraps up WebGL’s functionality into a more concise form. This would hopefully result in better code and faster progress. Research also uncovered GLSL, a scripting language run on the client’s browser for optimum performance, which could be incorporated into Three.js using a the Shader Material (mrdoob, ShaderMaterial).

## Colour Quantisation

If the exact colours extracted from an image were plotted on the histogram their frequencies would be too low. Colour quantisation is a method of compressing an image by reducing the number of colours while loosing as little visual information as possible. It was found that colour quantisation could be applied to allocate colours into ranges that would be plotted onto the histogram.

Khouloud Meskaldji et. al.’s paper on the effects of colour quantisation on histograms (2009) was surprisingly irrelevant and quickly discarded. Segenchuk’s webpage (1997) covers the details of various quantisation techniques; from this it was concluded that the simpler techniques were suitable for my project because other methods wouldn’t sample the image in a uniform grid in order to give more resolution to areas of higher density. Eric Liao’s GitHub page (2013) provided a pseudo-code example of an appropriate quantisation algorithm that would be useful in the development of the project.

## Colour Adjustments

The colour adjustments made available to the user had to be chosen carefully as only a few could be implemented in the limited development time and those chosen should have the best chance of having a useful effect on the histogram.

Apple’s iPhoto (2015) provides various colour controls and a 2D colour histogram; experiments were conducted into which colour controls had the greatest effect on the histogram. Exposure (brightness), contrast, and saturation, and lift/gamma/gain were identified as the two sets of controls with the most potential. Hurkman’s web page (2013) provided formulae for applying lift/gamma/gain and ASC-CDL, summarised by Nikolai (2013), was also identified as a method of applying colour alterations.

# Summary of Specification and Design

The specification and design document was written in preparation for the development of the project. This section of the report summarises the design for the sake of completeness and so that a description of the final product’s design (in section §5) may be evaluated against it (in section §6).

## Project Introduction

During my summer internship I discovered Javier Villarroal’s 3D colour histogram (H3Stogram - 3D Interactive Color Histogram, 2013), which motivated me to investigate the effects of colour alterations on colour histograms. The experience gained from my internship suggested that histograms could be a useful visualisation in image editing software. The aim of this project is to develop a program to investigate whether colour alterations have a significant effect on 3D colour histograms and whether they would be a useful visualation and provide a unique perspective on colour spaces.

The target audience of the project is users of image editing software who would benefit from a visualisation of colour spaces and the effects of colour adjustments on the distribution of colours within them. The audience require a basic knowledge of colour spaces and colour manipulation in order to be able to infer useful information from the histogram.

The basic requirements of the project are that it will allow the user to: upload an image from their local file space; view and control the viewpoint of the image’s 3D colour histogram in the sRGB and CIE-L\*a\*b\* spaces; alter the image’s brightness, saturation, and contrast levels; and view the effects of these adjustments on both the image it’s histogram.

## System Architecture

The project’s architecture is designed with the Model View Controller (MVC) design pattern. MVC separates the parts of the program responsible for processing and displaying data: the model processes data and contains the program’s business logic; the view handles presentation of data, in this case the histogram; and the controller acts as a liaison between the model and view, prompting the model to update the data when the view receives user interaction and instructing the view to refresh it’s view of the model’s data when it changes. The basic UML diagram of an MVC application can be seen in Figure 3:1. MVC was chosen because the separation of presentation and business logic allows flexibility and builds in a tried and tested design from the beginning.

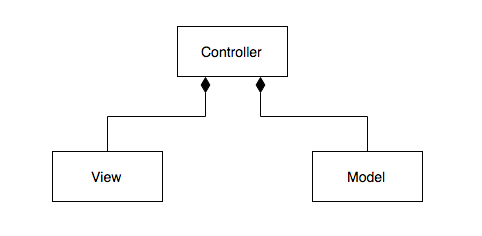


Figure : Model View Controller UML Diagram

## Component Design

The project is broken down into concise component tasks. The data flow diagram in Figure 3:2 illustrates how data is passed between tasks and fit into the MVC design.

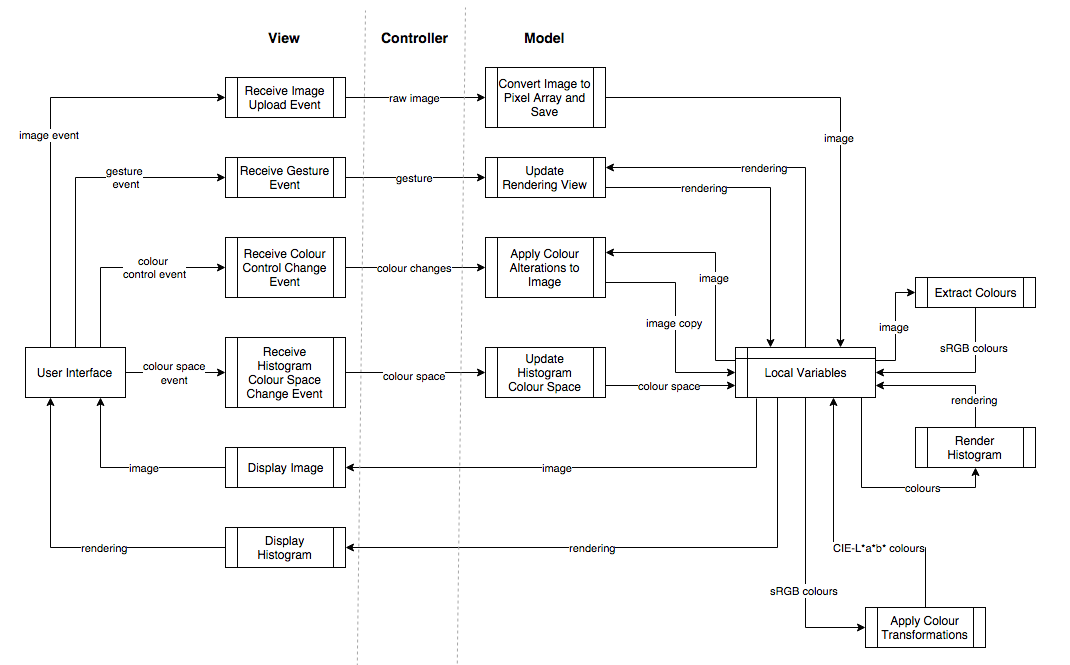


Figure : Data Flow Diagram

### Model

Tasks in the model include: extracting colours from an image, transforming between colour spaces, applying colour alterations, and generating the histogram rendering. State need not persist beyond the life of the program so data is stored in fields instead of databases.

Transforming between the sRGB and CIE-L\*a\*b\* spaces is achieved with an intermediate space, CIE-XYZ, as explained in section §2.1. Converting sRGB to XYZ is done by linearisation followed by a transformation matrix: Figures 3:3 and 3:4 show formulae and equations for these two operations. XYZ is transformed into L\*a\*b\* with another formula, seen in Figure 3:5.

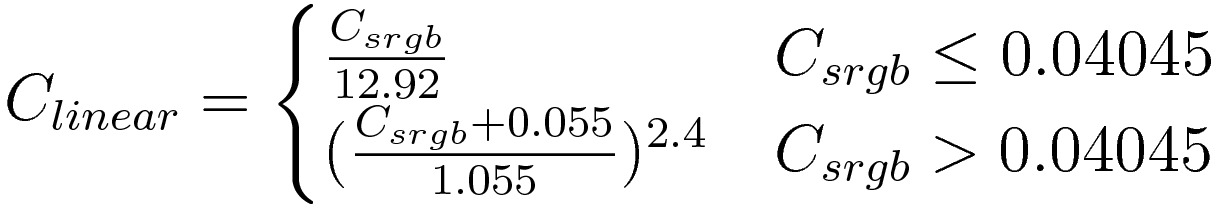


Figure : sRGB Linearisation Formula (Juckett, 2010)

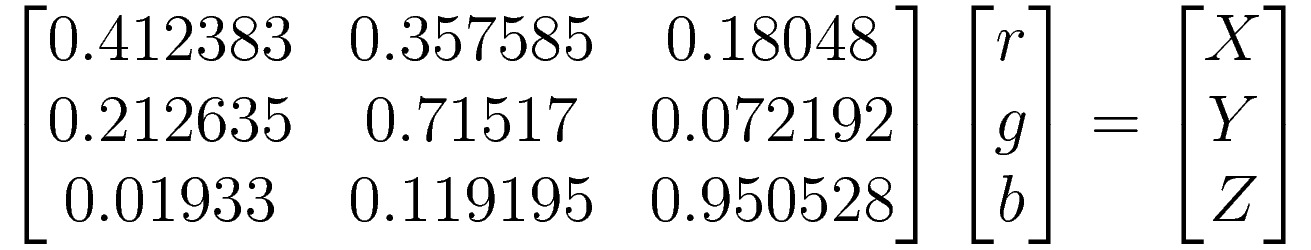


Figure : sRGB to CIE-XYZ Transformation Matrix (Juckett, 2010)

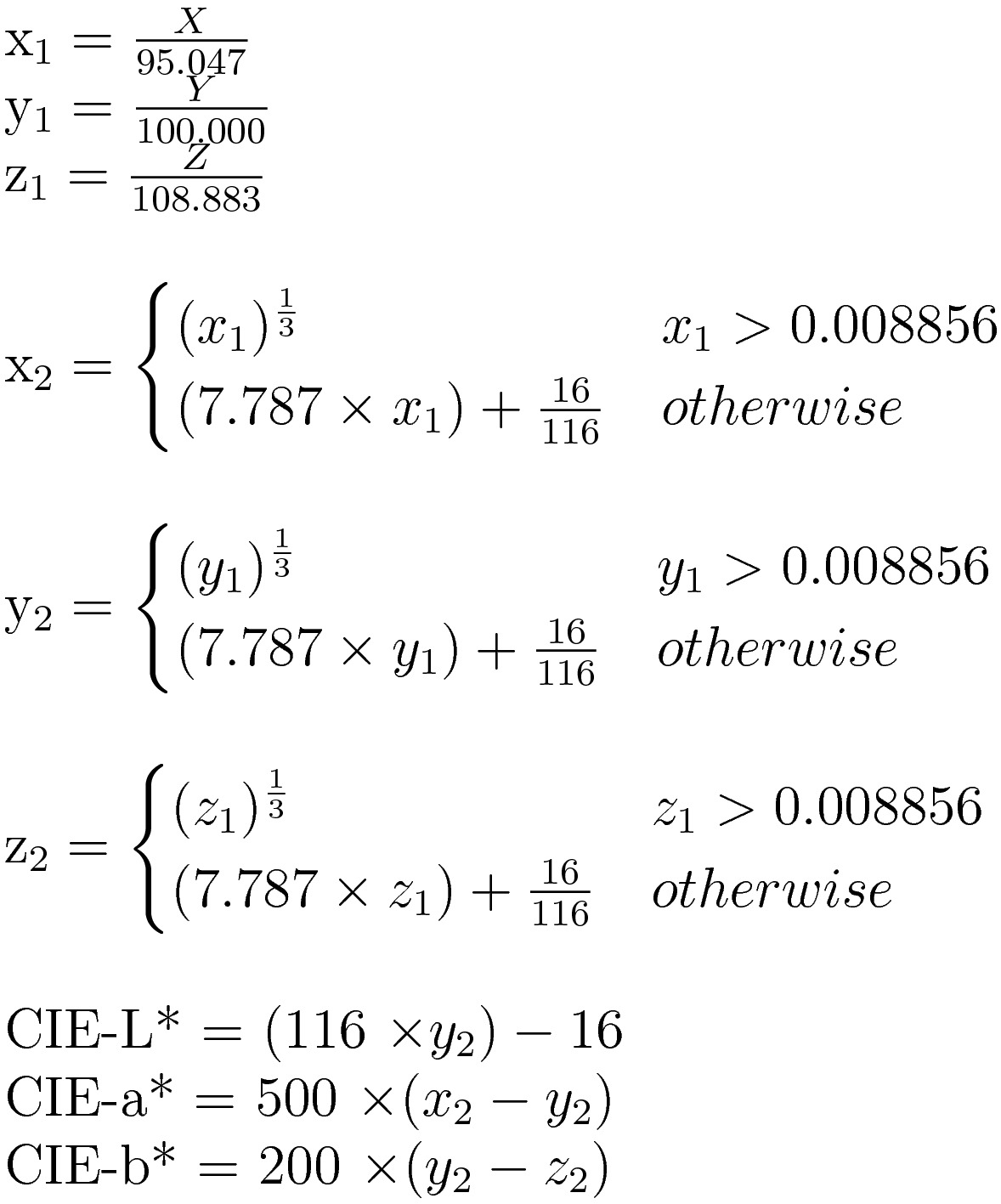


Figure : CIE-XYZ to CIE-L\*a\*b\* formula (Irotek Group Ltd., 2014)

All colour adjustments are applied to colours in the sRGB space. Brightness is adjusted by adding or subtracting a constant value to or from each colour channel. Contrast is adjusted using the formula in Figure 3:6, which calculates a contrast factor before applying it to each colour channel, and saturation is adjusted transforming into the HSL space and adding or subtracting to or from the S value. The formulae to transform between sRGB and HSL can be seen in Figures 3:7 and 3:8.

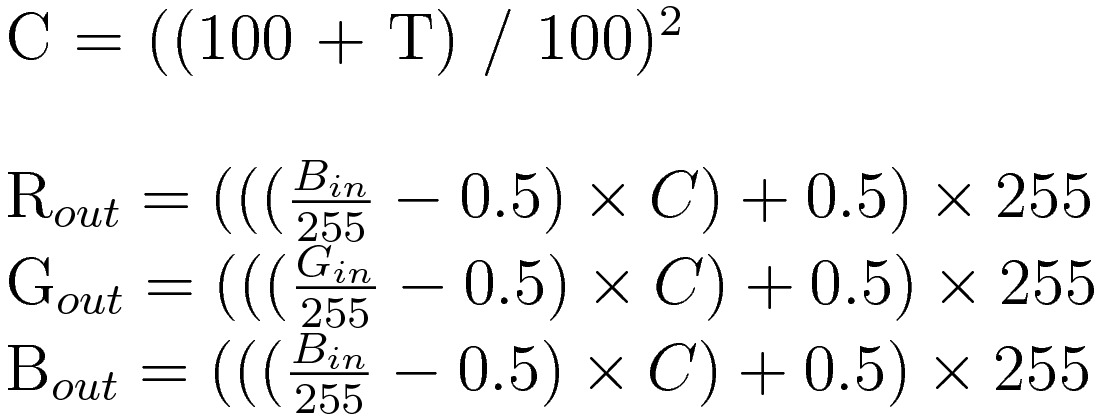


Figure : Contrast Formula (Esterhuizen, 2013)

|  |  |
| --- | --- |
| Figure : sRGB to HSL (RapidTables, 2015) | Figure : HSL to sRGB (RapidTables, 2015) |

### View

Events are fired by user interface in the event of the user uploading an image, performing a gesture on the rendering canvas, changing the histogram colour space, or adjusting the colour controls: in each case the event is caught and relevant information is sent to the controller. The other task of the view is to update it’s representation of the image and it’s histogram.

### Controller

The DFD does not define tasks for the controller and instead implied tasks lie between the view and model where data is passed between the two. These tasks either receive events from the view and prompt the model to update it’s state or fetch data from the model when it’s state changes and passes it to the view.

## UI Design

The program’s user interface will feature the histogram in the centre, surrounded by the image in the top left, above an image upload button and three colour control sliders, and colour space controls in the top right. Uploading an image, adjusting colour controls, or changing the colour space will fire events, handled by the view. Additionally, events are fired when the user left or right click drags or scrolls over the rendering, providing rotation, panning, and zooming controls on the rendering.



Figure : User Interface Sketch

## Testing and Evaluation

A suite of unit tests written with the test driven development paradigm will be used to test the program. Tests should be written to cover a broad range of scenarios, focusing on edge cases, which are the most likely to cause erroneous behaviour. Intentionally erroneous data should also be tested to ensure that the program handles it correctly. It is not possible to formally test the rendering but instead manual tests will be conducted, in which the output will be compared to expected results.

The project was founded upon the idea that colour alterations might have a significant effect on histograms in a way that is useful and unseen in 2D visualisations: while not a formal hypothesis, these ideas will be tested in the evaluation of the project, along with comparing the final product against the project requirements outlined in section §3.1. Some impartial people should also be used to evaluate the project to avoid obvious bias.

## Time Plan

The time plan (Figure 3:3) defines tasks down the left-most column and highlights the weeks in which they should be completed in the subsequent columns. Development is broken down into the first 13 tasks and the final 3 tasks are writing this report and preparing for the poster session and demonstration. Formal deadlines are explicitly marked and informal deadlines are implied at the end of each task’s allotted time.

Dividing development into short tasks, each no longer than 3 weeks, allows an iterative development cycle to be adopted: each cycle consists of implementing a feature, testing it, and a brief reflection and evaluation. Short cycles make development flexible as frequent adjustments to the time plan or design can be made.



Figure : Time Plan Gantt Chart

# Development and Testing

The development and testing methodology is defined in §3.6 and this section of the report describes the development and testing process that actually took place, provides illustrations, and describes points of interest during development. A development diary containing a detailed record of progress was kept and it is used to provide evidence: a complete copy of the diary can be found in the appendix.

## Development and Testing Methodology

Short and iterative development cycles consisting of implementation, testing, and reflection were kept throughout the project. An example of how short cycles were used can be seen in the diary entry for week 6 of term 2:

Week 6 (15/2/2016 – 21/2/2016)

* Used simplified colour transformations (found at <http://www.easyrgb.com/index.php?X=MATH&H=07#text7>) to convert from RGB to Lab
* Refactored ColorRGB class to incorporate both RGB and Lab values into a single object
* New Color class also includes the code to convert between colours spaces

This diary entry illustrates how the task of implementing the sRGB to CIE-L\*a\*b\* transformation was completed during the scheduled week.

While the time plan was never altered, there were times when it was not strictly adhered to and the project fell behind schedule: the diary entry for week 9 of term 2 shows that the goal of implementing colour adjustment formulae was not met that week:

Week 9 (7/3/2016 – 13/3/2016)

* Little progress made …
* Secondly, adjusting brightness results in strange artefacts (extreme colours) to appear
* Are these a result of a poor brightness algorithm? Find out by changing the colour of the image

Despite this small setback, the next diary entry shows how development was back on track by the following week:

Week 10 (14/3/2016 – 20/3/2016)

* The image now changes colour with the histogram …
* Saturation adjustment now also works, using HSL in the same way to brightness …
* Contrast adjustments added using the algorithm found here: <http://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-5-contrast-adjustment/>
* All colour controls now added: only question now is how to handle clipping

Alongside implementing a feature, unit tests would be written in what is the second part of the development cycle. Test driven development was initially adhered to, shown by the diary entry for the 3rd week of term 2:

Week 3 (25/1/2016 – 31/1/2016) …

* Began looking into testing, allowing me to start TDD and unit tests for existing code
* Configured WebStorm to use Karma to run QUnit tests (JetBrains, 2015), this took a day and was a lot of faff but nothing important to conclude ...
* Colour quantisation function in the model is the first to be written using TDD

Unfortunately, testing became less and less frequent as the project progressed. While the features whose correctness was vital were tested, some remained untested and fewer still were tested with test driven development, but rather tests were written at the end of the cycle.

While it was possible to write formal tests for most tasks in the model, tasks handling images were untestable because files uploaded through the HTML5 File element could not be replicated in testing for security reasons. The histogram rendering could not be formally tested either due to it’s nature, along with the GLSL scripts which performed some colour transformations. Manual inspections of the rendering had to be made to ensure its correctness. The sRGB space is easy to inspect as one simply looks to see if the vertices are the correct colours and colours in between are interpolated correctly; it is unlikely that this will be rendered wrong as there are no colour transformations and the framework renderings in sRGB. The CIE-L\*a\*b\* space is harder to test because the framework does not interpolate colours correctly so they must be defined pixel by pixel. By rendering a 2D plane at L=50, the result can be compared to Figure 4:1, which the correct representation of the L\*a\*b\* space. GLSL is required to render in the CIE-L\*a\*b\* space without severe performance issues and while moving calculations to the client’s GPU would improve the program’s performance it would make more code untestable, so it is currently a balance between testable and quick code.

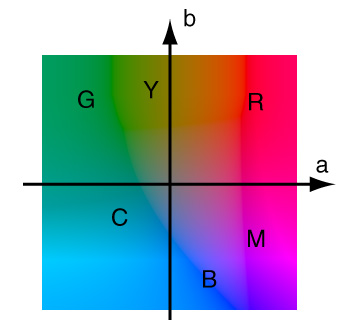


Figure : CIE-L\*a\*b\* at L=50

(http://www.arch.virginia.edu/~km6e/arch551/content/lectures/lec-05/page-2.html)

## Points of Note

The design specifies that the program be structured with the MVC design pattern. During development the design was adjusted while ensuring it was still MVC. The first week’s task was to restructure code written in previous months into the MVC design. Alex Netkachov’s web page (Model-View-Controller (MVC) with JavaScript, 2015) provided me with a JavaScript template of the MVC design pattern and was used to restructure the project. At this point it was also noticed that the original design was not truly MVC and that tasks responsible for rendering should be moved from the model to the view because they were concerned with representing data and not processing it. The diary entry for week’s 1 and 2 of term 2 shows evidence of this:

Weeks 1 & 2 (11/1/2016 – 24/1/2016)

* Prototype code was refactored into MVC using Alex Netkachov’s (2015) template and old (non-MVC) code achieved into “prototype” directory …
* Original DFD diagram had to be changed:
  + All rendering is now done in the view, as it regards the presentation of data and should have no effect on the representation of data in the model
  + Following on from above, a custom colour class was created, allowing me to define colours how I want, independently of the rendering framework, which should be isolated to the view

At the end of development the program was due to be debugged and optimised. Because testing had been conducted already no debugging was required and attention was focused on optimisation, a process that differed from the previous development cycle. Using Google Chrome’s in-browser profiler, it was observed that it would take an unreasonable amount of time to display on the web page images: it was taking 24 to 25 seconds to extract the colours from a 305 by 235 pixel image. Sampling a smaller number of pixels was tried but this resulted in blocky images and the histogram contained notably fewer colours. To overcome this the values of all pixels were collected at once instead of querying the DOM for each pixel individually. This reduced the computation time from 25 seconds to 0.6 seconds. After this the next longest task was instantiating colour objects. Currently a new object is created for each pixel, which is wasteful. Alternative solutions create one object for each unique colour or represent an image’s colours a 1D array of primitives. There was no time to explore these avenues but they are discussed further in section §8.

# Description of the Final Product

In this section the design and function of the final product will be described. To avoid repetition, the original specification and design in section §3 will be referenced when appropriate.

## System Architecture

The product’s architecture implements the Model View Controller (MVC) design pattern, a UML diagram of which is seen in Figure 5:1. The controller holds an instance of the model and view, allowing events fired by the view to listened for and methods called on the model to update it’s state accordingly. The view contains an instance of the model so that it can listen for events indicating the state has changed and the updated fetch data to display.

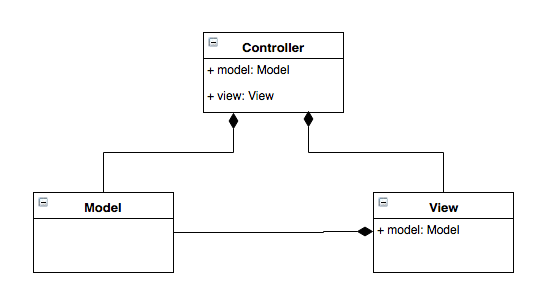


Figure : MVC UML Diagram

## Component Design

The program is divided into component tasks and grouped into the MVC design; the data flow diagram in Figure 5:2 illustrates how data is passed between these tasks.

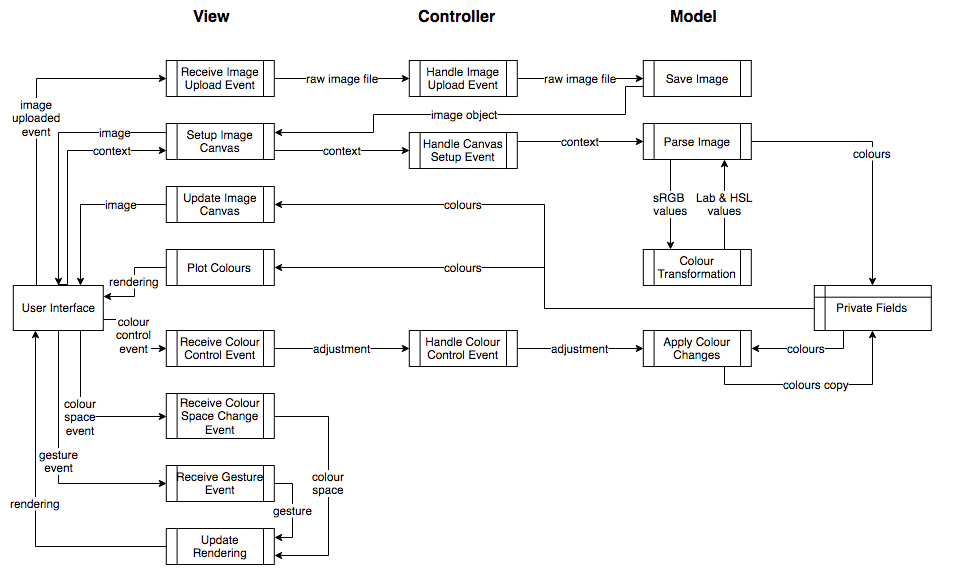


Figure : Data Flow Diagram

### Model

As well as holding the state of the colours the model is responsible for: extracting colours from an image, converting sRGB colours to CIE-L\*a\*b\* and HSL spaces, and apply colour alterations. Whenever the model’s state changes an event is fired and is caught by the view, prompting it to use the model’s getter methods to fetch the updated data.

While the view converts an image into a collection of pixel values (covered in 5.2.2), the model extracts the colours from the image. When the image has been placed in a canvas a context of that canvas is sent back to the model: this context can be inspected to get the colour value at each pixel and the Parse Image task uses it to extract a list of colours in the image.

For each pixel in the image a colour object is created to represent the colour of that pixel. On instantiation the object converts the inputted sRGB colour into the CIE-L\*a\*b\* and HSL space with the Colour Transformation task. The list of colour objects is then stored as a private field of the model, which is accessible to the view through a getter method.

The colour object represents a single colour. The object contains a number of fields which holds the colour’s values in the sRGB, CIE-L\*a\*b\*, and HSL spaces. The values of these fields are calculated by methods on the object, which implement various formulae for converting between spaces. The formulae used for converting between sRGB and HSL are the same as those in Figures 3:7 and 3:8 in the original design. The conversion from sRGB to CIE-L\*a\*b\* uses CIE-XYZ as an intermediate space. Formula for transforming from CIE-XYZ to CIE-L\*a\*b\* are also as defined in Figure 3:5 of section §3, but the linearisation function and transformation matrix in Figures 3:3 and 3:4 respectively have been replaced with the formula seen in Figure 6:3, which is functionally identical but easier to implement in a program.

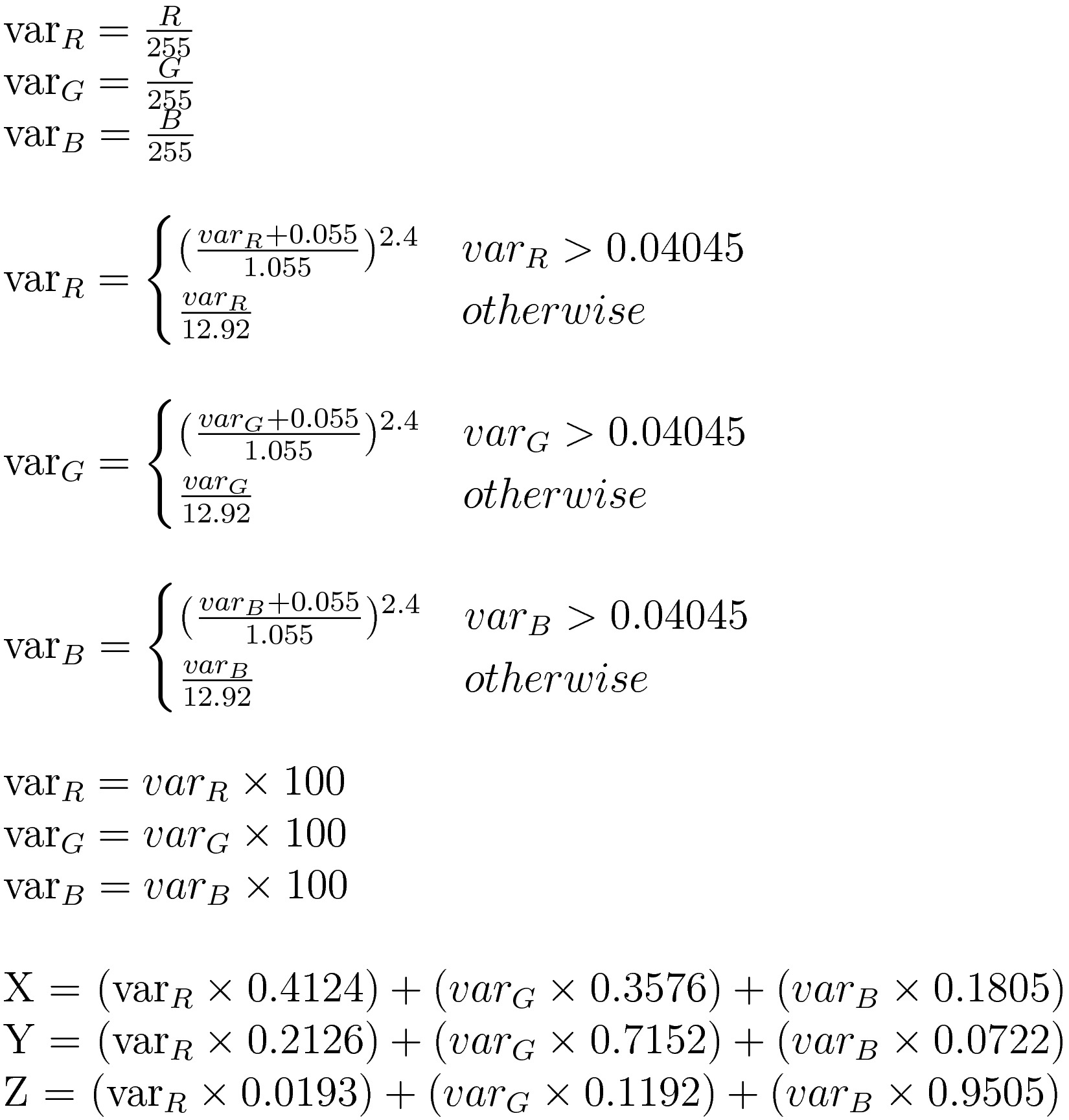


Figure :

The other operation that the model carries out is applying colour alterations in the Apply Colour Changes task. Section §3.3 explains how brightness is applied by adding a constant value to each colour channel, specifies the formula for applying contrast adjustments in Figure 3:6, and how the HSL space is used to adjust the saturation value. An extra step is taken after these formulae are applied to ensure that if a colour channels falls outside the valid ranges then the channel is clipped to the closest limit. When the altered colours are saved to private fields in the model, they do not overwrite the original colours, because colour adjustments are not reversible if colour channels are clipped. Therefore all colour alterations are applied to the original list of unaltered colours. It is the altered list of colours that is accessed by the view.

### View

As in the original design, tasks in the view either handle events from the user interface or update the representation of the data, which in this case is the histogram and a copy of the image.

Most events that receive events from the UI extract the relevant information and send another event, which is caught by the controller so that the model’s state can be updated accordingly. An exception to this is the Receive Colour Space Change Event and Receive Gesture Event tasks, which are handled within the view as they do not affect the model’s state but instead are used to update the user’s view of the histogram. Other tasks that are responsible for updating the user’s view of the histogram and the image it represents are Update Image Canvas and Plot Colours, which use the view’s instance of the model to fetch data when the model fires an event saying that it has been updated.

The task that sits on the line between handling events and updating the data presentation is Setup Image Canvas. This tasks receives an Image object from the model, formed with the raw image file passed to it from the view, and puts that object into an HTML5 Canvas object. This allows the user to view the image they uploaded but it also allows the model to inspect the value at each pixel using the canvas’ context, which is passed back to the mdoel by the Setup Image Canvas task.

### Controller

The controller contains an instance of both the model and the view. Tasks defined in the DFD diagram show how it listens to events from the view and passes them onto the model, which has no knowledge of the view’s existence. The controller does not intercept data flowing from the model to the view because the view listens directly to the model’s events.

## User Interface

The user interface of my program is a web page. The focus of the web page is the rendering of the histogram, which is placed in the centre of the window, and to the top left of the rendering is a copy of the image uploaded by the user. Running along the top of the page is a section that contains controls: a button allowing the user to upload an image from their local file system; a radio button giving the user the choice of viewing the histogram in the sRGB or CIE-L\*a\*b\* spaces; and a series of three sliders, representing the brightness, contrast, and saturation controls.

Interacting the user interface fires events that are handled by the view. The button below the image canvas prompts a window to appear which allows the user to navigate their local file system and select an image file; an event is fired when they confirm their selection containing a copy of the image file. As explained previously, the image will then appear in the canvas above the button. Changing the colour space radio button selection will also fire an event, this one causing the rendering to change to the selected colour space. Changing the position of one of the three sliders triggers an event containing the new position of the slider. Shortly after the image and the histogram will be updated with the newly altered colours. The final user interaction with the UI to cause an event is gestures on the rendering: dragging with the left or right mouse button held and scrolling with allow the user to rotate, pan, or zoom the rendering. Each gesture triggers and event containing the buttons pressed and location of the cursor on the screen, which is interpreted by the view to move the rendering accordingly.

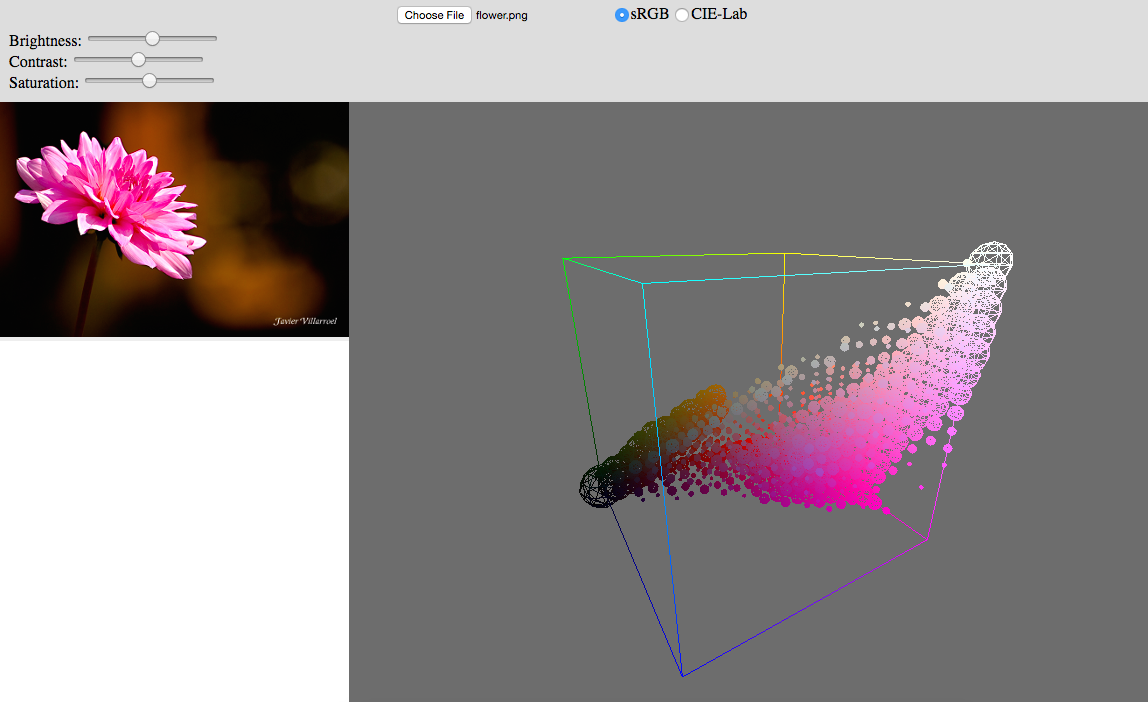


Figure : User Interface

## Product Behaviour and Performance

The original design specifies the product’s basic requirements are to allow the user to: upload an image from their local storage; view the image’s 3D colour histogram in the sRGB or CIE-L\*a\*b\* space as in interactive rendering; change an image’s brightness, saturation, and contrast levels; and view the effects of these colour adjustments on both the image and histogram. The final product provides all of these features.

References will be made to the product’s user interface, which can be seen in Figure 6:3. The user uploads an image by clicking the button labelled “Choose File” at the top of the window; this prompts a second window to open, allowing the user to navigate their file system and select an image file. Once confirmed, the uploaded image will be processed and will appear in the canvas on the left while the histogram of the image appears in the canvas on the right; this process will take a few seconds.

Initially presented in the sRGB space, the histogram can also be viewed in the CIE-L\*a\*b\* space through the use of the radio button at the top of the window. This will redraw the histogram in the new colour space. This should not take more than a second because the colours are transformed into the CIE-L\*a\*b\* space when the image is uploaded.

The rendering of the histogram can be manipulated by the user through gestures made on the rendering canvas. A left-click and drag will rotate the rendering around it’s centre; a right-click and drag pans the rendering’s centre; and scrolling will zoom in and out on the rendering centre.

Finally, the brightness, contrast, and saturation levels of the uploaded image can be adjusted through the use of the three sliders in the top left of the window. Initially in the centre position, moving them will cause the image and it’s histogram to be redrawn with the appropriate colour alterations applied. This process will also take a few seconds while the colours are manipulated.

While it provides all of the features defined in the design, it does not perform these operations in real-time as hoped.

# Evaluation of the Final Product

## Introduction

My project can be evaluated by comparing it to the specification and design document that I wrote before starting development: if my final product meets the requirements and matches the design in this document it will be considered successful. There were points in development where the design had to be modified in light of new information but as long as these changes can be justified and still meet the initial requirements they will be deemed acceptable. It can also be evaluated through user testing and experimentation to see if it allows me to answer the questions that I set out to answer: can 3D histograms be used as a visualisation tool in image editing in such a way that they provide a different perspective to common tools provided in image editing software.

## Requirements

The program that I have developed meets all of the basic requirements defined in design. It allows the user to: upload an image; view it’s histogram in the sRGB and CIE-L\*a\*b\* spaces, adjust an image’s brightness, contrast, and saturation levels; and view the effects of these controls on the image and it’s histogram. I therefore conclude that my project has successfully met the requirements set in the original design. One requirement that I failed to meet was that the program should run in as close to real-time as possible. As it stands, displaying an image’s histogram and apply colour alterations takes a few seconds of computation. I have previously discussed in §4 Development how I have successfully improved the speed of my program, but it is still a way away from real-time, but there are further improvements that could be made, which are discussed later. I also mentioned that if I had time left at the end of development I would begin work on transforming the histogram from a visualisation to a tool: I did not start development on this but it is discussed later in this document.

## System Architecture

I adhered to plan to use the Model View Controller design pattern set out in section §3.2, as shown in the description of my final product’s system architecture in section §6.1. As I developed my program I learnt more about to apply the MVC design pattern to my project and consequently the UML diagram in Figure 3:1 changed into that seen in Figure 6:1. The two diagrams are similar: the controller still contains an instance of the view and model, but the view was modified to contain an instance of the model so that could listen to events tired by the model and access data directly. This change was due to an error in the original design and a misunderstanding of how to apply the MVC design pattern in this case. However the current design is still not strictly correct: the implementation of the MVC design pattern does vary by interpretation, but after reviewing my code I am under the impression that user interaction should go directly to the controller, bypassing the view. This is a minor structure change that has no immediate effect on the program, but if the project were to progress it is an issue that should be resolved to prevent problems later on during development.

## Component Design

The final product contains all of the features outlined in the original design’s data flow diagram, but some of the features have been moved and the flow of data changed. The most notable change is that tasks responsible for creating the rendering of the histogram have been moved from the model to the view: I made this decision early during development after coming to the conclusion that, while central to the core concept of my program, the histogram was entirely a representation of the data in the model and did not carry out any data processing itself. Two additional effects of the rendering being moved to the view is that colours must be fetched from the model by the view and that the current colour space of the histogram is stored as state in the view, not the model. As these changes does not affect how the program behaves and it is more in keeping with the MVC design than my original design, I call this change an improvement.

Another alteration to the structure of the program occurred when I discovered that the HTML5 canvas object could be used to convert most common image file formats into an array of pixels. My original data flow diagram had a task in the model that took the raw image file from the view and converted it into an array of pixels that could be queried and saved it to the model, before extracting the colours form that image. With the introduction of the HTML5 canvas element the first tasks was split into two and incorporated the second. The model would first receive the raw image file, convert it into an Image file, which did not allow pixels to be queried but was passed back to the view where it was placed in the canvas, and the context sent back to the model. The context is then used to extract the colours from the image. The structured defined in the design is correct and this change only came about when a new and easier method of converting an image file was discovered. The structure was updated accordingly and therefore is an acceptable change.

The draw-back to this approach is that the view processes some data, in that it transforms the image into a form that is useful to the model. This does not strictly follow the MVC design, but performing this conversion myself in the model would have been too much work to achieve and my results would not have been as good as those obtained using the HTML canvas. It is also questionable whether the single task that converts a raw image file into an Image object should remain in the model or whether it is merely a artefact of my previous design and should be moved into the view. This does however still raise the question of whether images should be processed in the view.

As well as the newly added tasks to handle the rendering, the view contains four tasks which receive events from the DOM. Events fired when an image is uploaded or a colour control changed follow the design’s specification and are passed to the controller, but the two tasks that receive colour space change and gesture events no longer pass the event on to the controller because now that rendering is done in the view, they do not affect the state of the model, and are handled within the view.

Another change made within the model is when colour transformations are applied. The task that transforms a colour from one colour space to another has been moved to work with the task that extracts colours. This was done because during development I decided to perform all colour transformations when a colour object is created. I made this decision because I thought it would be better to sacrifice some time at the start of the program so that it would run quicker with minimised loadings later on.

The original specification and design document specifies a formula for the CIE-XYZ to CIE-L\*a\*b\* transformation, which is used in the final product: however the initial method of transforming from sRGB to CIE-XYZ using a linearisation function and transformation is replaced by a second formula. These two can be shown to be equivalent in Figure 7:1 where the example sRGB colour at the top is transformed first using a linearisation function and transformation, followed by the formula. One can see that both results are identical, showing that the two processes are also identical.



Figure :

When colour alteration formulae are applied, colour channels are clipped to the nearest limit if they fall outside the valid range. This naïve approach can create unexpected colours to appear and prompts consideration of how colours should be clipped. One other alternative is to scale each channel proportionally towards the centre of the colour space until the invalid channel is within bounds. I did not have time to experiment with alternative clipping methods during development but it is a consideration that I would like to explore in the future to see if it improves results of my colour adjustments.

## User Interface

The user interface of the final product contains all of the features outlined in the design and it is a web page. My intent was to have the rendering taking up the entire web page with the image and controls overlaid on top inside containing boxes. The final product’s user interface is more basic: the rendering and image are in two adjacent divisions, with a third above them containing all necessary controls. Additionally the divisions do not resize to fill the web page, but the image’s division does resize to fit the image. The reason for my user interface not meeting the requirements set in the design was that during time constraints I prioritised the program rather than it’s user interface. I do not regret this decision as I was able to produce a program that contained all the necessary features and met requirements, but improving the interface would be one of the first things I would do, given more time.

## Product Behaviour and Evaluation

As shown in §6.4, the final product contains all the features defined by the original design, but it must also be decided whether the product allow me to answer my initial questions set in §3.1: do colour alterations have a significant effect on an image’s 3D colour histogram and does the histogram provide a unique point of view, suitable for a visualisation to accompany in image editing tools?

The evaluation of my product, specified in section §3, would ideally be conducted both by me and by a number of impartial people who belong to my target audience. Unfortunately because my target audience is a relatively small niche of people, I was unable to find anyone impartial who had enough knowledge on colour to be able to properly evaluate my project and as a consequence I have had to do evaluation myself. I am not impartial but I will strive to be as objective as possible in my evaluation.

To evaluate my program, I will attempt to answer the two questions stated above. Applying colour alterations to an image and deciding whether these changes have had a significant effect on the histogram will answer the first question. The second question can be answered in two parts: firstly I will investigate whether the affects on the histogram are useful and secondly I will investigate whether the 3D colour histogram shows any advantage over it’s 2D counterpart in this task. During development I used the same image of a flower so decided that using an unseen image for the evaluation would ensure that the program is not tailored towards a one particular case. The image I chose for my evaluation was the photograph of Lena Söderberg, a famous test image in image editing. The unedited image and it’s histogram is shown in Figure 7:2. Figures 7:3 through 7:8 show the effects that increasing and decreasing brightness, contrast, and saturation levels have on the histogram. I have only used the sRGB space in these examples as it can be compared to 2D histograms.

Looking at Figures 7:3 to 7:8 and comparing them to Figure 7:2, I think that it is clear that the histogram is significantly affected by the colour alterations and that this answers the first of my two questions.

Examining the histograms in more detail, it can be seen that adjusting brightness moves colours towards the white and black vertices of the colour space. Some unexpected colours appear at the edge of the space but I attribute these to the method of clipping when applying colour adjustments. Varying contrast levels pushes colours towards the extremes of the colour space and pulls them into the centre. At first glance this is a similar effect to saturation, but on further inspection contrast repels and attracts colours to and from the centre point of the colour space while saturation attracts and repels around the white-black diagonal. These observations were predictable, are logical when compared to the operation causing them, and start one thinking why this might be the case: because these effects are clearly correlated to the colour adjustments, I conclude that the histogram provides a meaningful perspective to these operations.

Finally, my 3D histogram can be compared against a 2D histogram to determine whether the third dimension provides a unique perspective. I am comparing my program against Apple’s iPhoto (Apple Inc., 2015), which provides exposure (brightness), contrast, and saturation controls and a 2D colour histogram. On the 2D histogram, adjusting brightness shifts the three colour channels towards the ends of the scale; varying contrast spreads and contracts colour channels along the x-axis, while maintaining the basic shape of each plots; and saturation spreads or contracts the colours out along the x-axis, separating or coalescing the three colour plots without maintaining the plot’s shape. By comparing these observations to those on my 3D histogram, I conclude that the 2D histogram is good at showing how the colour channels change in relation to each other and querying the exact level of a colour. While precise values of the 3D histogram’s levels are not clearly visible, it offers the ability to view individual colours and the distribution of the colours. It is not possible to represent sRGB well in 2 dimensions because sRGB uses each axis as a primary colour. CIE-L\*a\*b\* space can be visualised in 2 dimensions by fixing the lightness value (as seen in Figure 4:1) but while this representation be accurate, it is limited as the third axis is lost. CIE-L\*a\*b\* is also a minority and most colour spaces can not be accurately represented in 2D, but with a 3D histogram any colour space can be represented perfectly.

|  |  |
| --- | --- |
| Macintosh HD:Users:callum:Desktop:Lenna_0.png  Figure : Lenna Uneditted | |
| Macintosh HD:Users:callum:Desktop:Lenna bright max.png  Figure : Increased Brightness | Macintosh HD:Users:callum:Desktop:Lenna bright down max.png  Figure : Decreased Brightness |
| Macintosh HD:Users:callum:Desktop:Lenna cont up.png  Figure : Increased Contrast | Macintosh HD:Users:callum:Desktop:Lenna contr down.png  Figure : Decreased Contrast |
| Macintosh HD:Users:callum:Desktop:Lenna sat up.png  Figure : Increased Saturation | Macintosh HD:Users:callum:Desktop:Lenna sat down.png  Figure : Decreased Saturation |

The optimistic goal of plotting an image’s colours on a histogram and performing colour alterations in real time was not reached. I prioritised creating a working product before optimising it and I deem this to be a wise decision. I made efforts to improve the performance of the product at the end of development and in the case of extracting colours from the image, was able to reduce the process down from 25 seconds to 0.6 seconds with a minor change. I have had more ideas as to how the program can be further optimised; unfortunately I did not have time to explore these options during development but they are discussed §9 along with other future work for the project such as the development of the project as a tool.

# Critical Assessment of the Project as a Whole

## Pre-Development

During the first term my focus was on the literature review and specification and design document. The time plan that I produced as part of the project statement and plan was helpful in ensuring I kept on track and met deadlines. Thanks to my previous experience in the area I was able to start my literature review with a number of sources already lined up. It would have been easy to become complacent with the knowledge and experience that I already had, but during my research I pushed myself to explore new areas and better understand those that I was already familiar with. At the end of my research, I had compiled and analysed a considerable number of sources for a variety of topics relevant to my project. A regret that I do have about by research was the lack of literature sources, as all of my sources were either web pages or online journal entries. The specification and design document was the second piece of work I would submit before I began development. It’s my opinion that this design document reflected both the results from my literature review and targets sets out in my initial project statement and plan. The level of detail was appropriate for a design at that stage in the project, allowing the reader to interpret the design himself or herself and implement how they choose, while still fulfilling the requirements of the project. Both the information I gathered in my research and the project design put me in good stead for starting development.

During this time I also started writing code, the aim of which was twofold: it would support my literature review and design, allowing me to come to more informed conclusions; and secondly it would give me a head start in the development stage of my project. Because code that I wrote was largely based on that which I had already written over the previous summer and there was little original or new material, progress was quick: however the code was written in an ad-hoc manner and not tested. As a result, the program was poorly structured and fragile, containing a number of bugs. The poor structure also meant that the code was difficult to build upon. Intervention had to be taken at the beginning of development to get the code to an acceptable standard; luckily, due to a realistic and well thought out development plan I had ample time to fix the code.

## Development Cycle

The start of the second term also marked the start of development. The specification and design document’s time plan spanned the final stages of my project but in particular it focused on development and how it was broken into short and iterating development cycles: a summary of this can be found in §3.6 and evidence of how it was carried out in §4. I found that short development cycles made the project more manageable. Each week I had a single goal to accomplish, which helped to keep my mind focused and prevented my from going down a tangent. Testing at each cycle gave me confidence that the program was correct throughout the project and removed the need for a large amount of time set aside for debugging and testing at the end of development, as this was all done alongside development. The review at the end of each cycle also allowed me to assess how much progress had been made and adjust my plans accordingly, or reassess the design of my project.

In hindsight I did not put enough emphasis on the review: I should have re-examined the project plan at each cycle. An example of where this would have been beneficial is in my data flow diagram. As discussed in section §6, the structure of my final program differed from the original design; this itself is not a problem as long as the project still meets the requirements and the design is sane, but I did not give enough thought to how the project was changing as I went along. Consequently, at the end of development there were aspects of the project’s structure that were not in accordance with the MVC design. I was however able to reflect at the end of each cycle in my diary. The diary was intended for retrospective analysis to help in the writing of this report, but it also allowed me to pause and reflect during development. While not intentional, this diary benefitted my project and it is a piece of initiative that I am proud of.

## Testing

The design specified that my code would be tested with unit tests and that the test-driven development paradigm would be followed. This was a good decision because it incorporates testing into the development cycle and results in better quality and correct code. While initially followed, I often got distracted from test driven development and forget to test until I completed a cycle. The tests that I wrote covered a wide range of scenarios, focusing on edge cases, and including testing with invalid inputs. While I wrote good tests, if I had kept to test driven development for more of development, better code would have been produced.

I also wish that I had done more research into testing frameworks before deciding upon QUnit. QUnit was useful but it lacked some functionality that I required, leaving holes in my tests: for example in order to test whether a value is true within a tolerance I had to define my own test method. Some parts of my program were inevitably going to be hard to test, such as the rendering, and I was not able to write any formal tests for these features: however these tests may have been possible with a different testing framework. In summary, QUnit turned out to be a limited testing framework that only offered me the basics in testing ability and using it for my application was a poor choice.

The parts of my project which were untestable by formal tests, such as the rendering, had to be tested manually. The process of convincing myself that the program was rendering the correct colours is discussed in section §4, but that’s all it was, convincing myself that it was correct, it would have been very difficult to convince a scrutinising reader that my renderings are correct, unlike my formal tests. Sadly, due to the nature of the rendering framework I used, I was not able to write formal tests to ensure that the colours were correct; even some of the colour transformations could not be tested as they are run on the GPU with the rendering. While testing the rendering may not be a realistic goal, doing as little as possible in the rendering so that more code can be tested would have been a better decision.

## Time Plan

I set myself a goal to complete development by Easter so that the remaining time could be allotted to this report, the poster, and preparation for the demonstration. I knew from experience that estimating the time needed to complete software development tasks was difficult so I was generous with my timings and prepared to alter the schedule if required, as long as development was still forecast to finish by the start of Easter. However it was not necessary to change the development schedule because I met each deadline, often with time to spare. While this success was party due to a good work ethic, in hindsight the time allocated for certain tasks, such as colour adjustment algorithms and colour space transformations, was overly generous would have been better spent improving the efficiency of my program or exploring future development as a tool at the end of development.

## Results and Evaluation

I am happy with the results that my project has yielded: in my evaluation (section §7.6) I was able to confidently say that the histogram provides a unique and useful perspective on colour distributions and the effects of colour manipulation. However only I was able to make these optimistic and convenient conclusions. The target audience, while broad enough that I am not the only one who would find my findings interesting, is restricted enough that I was not able to gather anyone impartial who could critically evaluate my project. I would not change my project requirements or target to accommodate as I still believe this project’s subject to be interesting, but if I were to start again I would consider my method of evaluation more seriously from the beginning and this would include gathering a small group of people who could help me to evaluate my project and make their own conclusions on whether the histogram is a useful tool.

# Future Work

While I have successfully implemented all the features I set out to in the design there are still improvements that can be made. The project also has potential beyond it’s current form because, as stated at the start of this document, I see the potential for 3D colour histograms to be used as a tool of image editing as well as a visualisation.

In it’s current state, I see two key areas of the project that could be improved: the user interface, and the program’s performance. Due to it being a necessity of my project and not the focus, the user interface is poorly designed and implemented. If my project was to become something that I could make available online, the user interface would need a nearly complete overhaul to get it to the necessary standards. If the program was to be released it’s performance would also need to be improved, otherwise the computation times render it almost unusable. I have already begun to optimise the program but there is a lot more that could be done, and must be done if I wanted to reach my goal of making it real-time.

Currently, each time a colour object is created it is converted into the CIE-L\*a\*b\* and HSL colour spaces to save time later one: however it may be preferable to use lazy instantiation and only perform these transformations when necessary so to speed up the web page initially. Another consideration that I could make would be to only have one colour object for each unique colour, referenced by each instance of that colour in an image, as I am currently performing the same transformations many times for duplicate colours. An extreme solution would be to represent an image’s colours an a one dimensional array of colours and apply all transformations to that array. This approach would reduce the number of objects but duplicated colours would be transformed multiple times.

In section §6 and at other points in this document I have mentioned how the naïve clipping used in the colour alterations sometimes causes unexpected colours to appear. I have learnt during this project that there are many ways to implement even the simplest alterations such as brightness, contrast, and saturation. The aim of my project was not to implement the best colour adjustment controls as I could so I am not overly concerned about this, but if I wanted to provide the most useful tool possible, more research into the best way to implement these controls and how to better perform clipping would be useful.

Looking beyond my current project, I see the possibility for 3D colour histograms to be used as tools, not just visualisations. My project was a prototype that aimed to answer the question of whether histograms offered a unique perspective on colours, unavailable with conventional and current tools. The results of my project are reassuring and suggest that histograms are useful as visualisations, but could this be carried on to develop them into tools as well?

* How these tools would work?

# Conclusion

# Bibliography

Apple Inc. (2015, March 19). iPhoto. *9.6.1*. Apple Inc.

Barthel, K. U. (2007, September 16). *3D Color Inspecter/Color Histogram.* Retrieved November 12, 2015 from Research Services Branch: http://rsb.info.nih.gov/ij/plugins/color-inspector.html

Caballero, M. L., & Belmonte, N. G. (n.d.). *Real time color 3D histogram analysis*. Retrieved October 3, 2015 from Senchalabs: http://www.senchalabs.org/philogl/PhiloGL/examples/histogram/

Cruse, P. (2015, June 12). *Introduction to Colour Spaces.* Retrieved October 2, 2015 from Colour Phil: http://www.colourphil.co.uk/lab\_lch\_colour\_space.shtml

Hoffmann, G. (2015, September 24). *CIE Color Space.* Retrieved November 5, 2015 from docs-hoffman: http://docs-hoffmann.de/ciexyz29082000.pdf

Hoffmann, G. (2015, September 24). *CIELab Color Space.* Retrieved November 5, 2015 from docs-hoffmann: http://docs-hoffmann.de/cielab03022003.pdf

Hurkman, A. H. (2013, May 20). *What's In a Name?* Retrieved November 9, 2015 from Pro Video Coalition: http://www.provideocoalition.com/whats-in-a-name

Irotek Group Ltd. (2014). *Color conversion math and formulas*. Retrieved October 2, 2015 from EasyRGB: http://www.easyrgb.com/index.php?X=MATH&H=07#text7

Javi Fontan, D. G. (2015, April 10). *GLSL Sandbox*. Retrieved October 2, 2015 from GLSL Sandbox: http://glslsandbox.com/

JetBrains. (2015, December 23). *Creating JavaScript Unit Tests*. Retrieved January 20, 2016 from JetBrains: https://www.jetbrains.com/webstorm/help/creating-javascript-unit-tests.html

Juckett, R. (2010, May 16). *RGB Color Space Conversion*. From Ryan Juckett: http://www.ryanjuckett.com/programming/rgb-color-space-conversion/

Khoronos Group. (2015, April 26). *Main Page*. Retrieved November 5, 2015 from WebGL Wiki: https://www.khronos.org/webgl/wiki/Main\_Page

Liao, E. (2013, April 18). *Uniform Color Quantization*. Retrieved November 8, 2015 from Github: https://github.com/rcliao/MultimediaSystem/wiki/Uniform-Color-Quantization

Loch, F. G. (2008). *Image Processing Algorithms Part 5: Contrast Adjustment*. Retrieved March 14, 2016 from Dreamland Fantasy Studios: http://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-5-contrast-adjustment/

McHugh, S. (2015). *Camera Histograms: Luminosity & Color*. Retrieved October 3, 2015 from Cambridge in Colour: http://www.cambridgeincolour.com/tutorials/histograms2.htm

Meskaldji, K., Boucherkha, S., & Chikhi, S. (2009). Color Quantization and its Impact on Color Histogram Based Image Retrieval. In O. A. Abdelmalek Amine (Ed.), *Conférence Internationale sur l'informatique et ses Applications.* *547.* Algeria: http://ceur-ws.org.

mrdoob. (n.d.). *ShaderMaterial*. Retrieved October 2, 2015 from three.js: http://threejs.org/docs/#Reference/Materials/ShaderMaterial

mrdoob. (2015, September 22). *three.js*. Retrieved September 27, 2015 from three.js: http://threejs.org/

Netkachov, A. (2015). *Model-View-Controller (MVC) with JavaScript*. Retrieved January 11, 2016 from Alex@Net: https://alexatnet.com/articles/model-view-controller-mvc-javascript

Nikolai. (2013, May 9). *Color Decision List - introduction*. Retrieved October 2, 2015 from Waldman Media: http://www.niwa.nu/2013/05/color-decision-list-introduction/

Parisi, T. (2010, December 6). *The Lessons*. (T. Parisi, Editor) Retrieved October 2, 2015 from Learning WebGL: http://learningwebgl.com/blog/?page\_id=1217

PracticalPhotographyTips. (n.d.). *Color Histograms*. Retrieved October 31, 2015 from Practical Photography Tips: http://practicalphotographytips.com/Digital-Camera-Basics/color-histograms.html

Rockwell, K. (2006). *How To Use Color Histograms*. Retrieved October 3, 2015 from Ken Rockwell: http://www.kenrockwell.com/tech/yrgb.htm

Segenchuk, S. (1997, May 5). *An Overview of Colour Quantization Techniques*. Retrieved November 7, 2015 from Worcester Polytechnic Institute: http://web.cs.wpi.edu/~matt/courses/cs563/talks/color\_quant/CQindex.html

Steer, W. A. (2008, March 15). *Introduction to Colour Science*. Retrieved November 5, 2015 from Techmind: http://www.techmind.org/colour/

Tarlier, F. (2009, November 5). *lift/gamma/gain VS. ASC CDL*. Retrieved October 2, 2015 from Francois Tarlier's blog: http://www.francois-tarlier.com/blog/liftgammagain-vs-asc-cdl/

Third Ave Design, T. S. (n.d.). *3D Histogram*. Retrieved October 3, 2015 from 3D Histogram: http://3dhistogram.com/?#contact

Villarroal, J. (n.d.). *H3Stogram - 3D Interactive Color Histogram*. Retrieved September 27, 2015 from H3Stogram: http://h3stogram.herokuapp.com/

Villarroal, J. (2013). *H3Stogram - 3D Interactive Color Histogram*. Retrieved September 27, 2015 from H3Stogram: http://h3stogram.herokuapp.com/

Wikipedia. (2015, January 8). *ASC CDL.* Retrieved October 2, 2015 from Wikipedia: https://en.wikipedia.org/wiki/ASC\_CDL

Wikipedia. (2015, September 26). *CIE 1931 color space*. Retrieved October 2, 2015 from Wikipedia: https://en.wikipedia.org/wiki/CIE\_1931\_color\_space#Color\_matching\_functions

Wikipedia. (2015, September 29). *Color histogram*. Retrieved October 2, 2015 from Wikipedia: https://en.wikipedia.org/wiki/Color\_histogram

# Appendices