RAWFlash: A cloud based RAW image editor

Student Name: Ryan Collins
Supervisor Name: Dr Tom Friedetzky

Submitted as part of the degree of [MEng Computer Science] to the
Board of Examiners in the Department of Computer Sciences, Durham University
April 10, 2018

Abstract —

Context/Background RAW images are used by photographers to improve the quality of their images, specifically when editing. With JPEG files, the loss of data from compression leads to poorer quality edits, while using RAW results in a higher dynamic range. The editing of these files is traditionally done on a high-spec machine, capable of processing images with large resolutions. In the field, this can be difficult. When used for websites, the standard process involves editing a RAW file locally and exporting it, uploading the exported image to a Content Management System (CMS). Here, the CMS will then edit the file again, resizing the image to ensure it displays properly on the browser.

Aims The main aim of this project is to test the feasibility of a Cloud-based RAW image editor, allowing image adjustments such as exposure adjustment, colour adjustment, and overall improvements to the quality of the image, all within a cloud application rather than on the desktop.

Method A render server back-end will first be implemented as an API, taking in an input as a JSON object, and then processing the image, and then a JavaScript client shall be created to interface with this API.

Results

Keywords — RAW image editing, dcraw, cloud image editing, docker, image processing

I INTRODUCTION

The project itself concerns itself with editing RAW files through a system deployable to the cloud so that it can both be used by people editing photos, as well as by extension by other systems. While RAW editing isn't new, the creation of a system to edit RAW files would give users more control in content management, and produce better quality images.

A RAW Files Explained

RAW isn't a file format in itself, but rather an umbrella term for a set of many different formats that store the raw camera sensor data. There are a set of different proprietary file formats, that all store similar information, but in different ways.

When an image is captured, the camera sensor uses charge buildup to represent the amount of light that is incident on the sensor (using a phenomenon known as the photoelectric effect). Colour data itself isn't stored at all in the RAW image. Colour is achieved by putting a filter over the sensor, such that each individual part of the sensor captures only red, green or blue light, which then allows one to build up a full colour image. The process of creating a full colour image from the camera sensor is called demosaicing.

There are several different steps to decoding RAW images, in addition to demosaicing which was explained above. These are:

White Balance In order

Gamma Correction Cameras typically represent colour changes linearly, with a gamma of 1.0. **TODO FIND SOURCE OTHER THAN CURRENt**. However, this isn't necessarily pleasing to the eye, so this can be changed to change how tones are reflected. For example, this can

Sharpening/Noise Reduction Sometimes, noise can creep into the image, particularly if the image itself is shot with large ISO levels (ISO is a sensitivity setting that can be set when taking a photo, to allow the camera sensor to be more sensitive to light **TODO FIND SOURCE FOR ISO DEFINITION**). This can be corrected during the editing process.

Also, sometimes edges might need to be enhanced to bring out the detail in the image. This can again be used for stylistic purposes, but this is ultimately the decision of the image editor.

A.1 Comparison to JPEG

JPEG, a common format in the consumer photography market, often has smaller files due to JPEG compression. While this can be good to minimize the amount of memory used, the lossy compression is problematic as adjusting tone and colour information as JPEG heavily compresses colours (so minor adjustments in colour will appear as the same colour in JPEG). When shooting an image with JPEG, the camera uses a built-in RAW processor to automatically process the RAW data, perhaps using some settings that can be adjusted in the camera, and compresses the image down into a JPEG format.

On the other hand, RAW allows one to have far more control over the final image, by adjusting various settings with a far greater freedom and by using the camera's built in processing.

(Fraser 2004)

B Applications of RAW Image Editing

C Problems with Current Implementations

Current Implementations

Most current implementations exist as native applications, designed specifically for one or several different platforms. With this, a specific machine shall have the program installed, and shall contain all the features from managing files, to rendering images, to displaying the output to the user as required.

However, larger images can require a larger amount of resources to render properly, and furthermore if one intends on editing RAW images on the go, transporting a larger machine isn't convenient. Furthermore, the RAW editing software might not necessarily be available on the platform that is nearest to the user.

Instead of this approach, this project is taking an approach of having a Cloud system that can render images, which can scale if necessary by adding more servers, so whatever client device is used, raw images can be edited providing the user has an internet connection (which with the recent presence of 3G and 4G, they are likely to have).

D Project Objectives

II RELATED WORK

A RAW files, and parsing

B Existing Solutions

Adobe Lightroom Adobe Lightroom is a native and proprietary tool, developed by Adobe, for editing images, both RAW images, and otherwise. RAW processing in Lightroom is done by Adobe Camera RAW (as Lightroom builds on this useful tool),

C

III SOLUTION

A Client-Server Communication Protocol

B Rendering Server Structure

C Web Editor Functionality

D Storing User Images

While one can edit individual images by simply supplying them to the editor, they need to be accessed somehow, preferably through supplying some URL, where the RAW image can be downloaded, and then used to edit an image. The system shall be built to be as customizable as possible, by downloading images from a supplied URL in the request, and this RAW image is downloaded (or cached to the render server), wherever the file itself is hosted. This way, we can either supply a system for uploading/storing images, or the user can choose to use an image store elsewhere such as Amazon S3, OpenStack Swift, or other storage backend. Our system simply uses the Django file system, which stores the file in an uploads folder on disk.

E Managing a big system

F Unit Testing Issues with Image Processing

For one to compare the output of the system with other RAW editors is not necessarily as easy as bitwise ANDing the files, and finding the percentage differences. While this might work

if all systems use the same algorithms to do the same tasks, due to the proprietary nature of RAW image editing, different implementations of demosaicing exist, yielding slightly different images, along with potential different ways of adopting equalization, gamma correction and various other colour adjustments (for example, we use RGB gain to adjust the colours, while LightRoom instead uses Chromacity). It'll be difficult to use the same settings for every single system. Instead, it might be better to edit a variety of different images, and compare them by eye, to determine whether the algorithm itself is working correctly, and using base cases to test the output of a system by eye.

Base cases include:

Using a gamma of zero will yield a completely white image. Uisng a gamma of 1 will yield the input image, for gamma correction. Setting red gain to 1, and all others to zero will yield an image with only the red channel. Doing the same for green and blue should reveal the same. This will ensure the algoritm is working correctly.

Exposure zero should yield a completely black image. Using a very high exposure should yield a completely, or nearly white image.

We can test undo and redo trivially.

IV RESULTS

A Comparison With Other Editors

Testing the system against Lightroom, Darktable and RawTherapee, testing the image output compared to these, and also performance based tests compared with traditional native software.

TODO: Mention specs of the machine.

B User Testing

Get User Comments on system

C

V EVALUATION

A Application to Web Content Management Systems

B Benefits of Portable Image Editing

C Performance Issues and Improvements

VI CONCLUSIONS

This section summarises the main points of this paper. Do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions. This section should be no more than 1 page in length.

The page lengths given for each section are indicative and will vary from project to project but should not exceed the upper limit. A summary is shown in Table 1.

Table 1: SUMMARY OF PAGE LENGTHS FOR SECTIONS

	Section	Number of Pages
I.	Introduction	2–3
II.	Related Work	2–3
III.	Solution	4–7
IV.	Results	2–3
V.	Evaluation	1-2
VI.	Conclusions	1

References

Fraser, B. (2004), White paper: Understanding digital raw capture, Technical report, Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA, United States. Accessed 10th April 2018.