

DNG Processor

Bring out the best, then do it again

2017 - 2019

Amir Zaidi

Background

Ever since 2017 I have fantasized about making my own camera application. I wanted to give people better images with the phone they had, and fix the problems that OEMs introduce. When the OnePlus 5 released with dots in its images and a watercolour effect, I became eager to show the world what I could do. Back then I started work on an app called "Night Camera", which made me learn about the camera API on Android. There were, however, problems in this API that could not be overcome. To fix them I would need access to secret OEM code. Some might remember when I released some half-broken builds of Night Camera to test the behaviour of the API on various phones. It looked like a dead end, but I explored the remaining possibilities.

One day I stumbled upon old, forgotten code from AOSP Compatibility Test Suite, where they compared a device's RAW with the corresponding JPEG image to check for their similarity. I implemented this code into Night Camera in Spring 2018, and this was the start of real progress followed by a lot of testing, where I ran into two hard limits. One, this RAW processing code written in Renderscript by Google was so slow that it would be unusable to run during the use of a camera app. Two, third party camera apps still have a lot of bugs because of the API problems, even when just requesting the RAW data from the device. At that point I gave up and dropped the project of making a camera app altogether.

However, a few months later I got a new idea. I could use the RAW images saved by the OEM camera app, and process those in the background, albeit slowly. This was how "DNG Processor" was born (DNG is the file extension used by RAW images). The first challenge was to learn to parse a DNG file and extract the RAW information from it. You would expect that Android has built-in support for this, but I had to write the code that manually reads it byte-by-byte. There were also nearly no sources online that showed how it worked, thus many hours were wasted on testing my own implementation from scratch. Once this was done, I attached it to the old RAW processing code used in Night Camera. It took 20 seconds of intensive CPU work to process a single image and the result looked terribly noisy, but I was getting somewhere.

Then I spent time learning about denoising, histogram equalization, and other important techniques that make an image look "good". Lots of trial and error with various algorithms, a few from literature and a few that I came up with myself. One thing was still not solved, and that was the atrocious runtime. When running as a background application it could take nearly a minute to process one single image, during which your phone would heat up like an oven. I couldn't find a way around that and once again put the project on hold.

Now, it is back, and better than ever. Over the past half year I learned to use OpenGL and I had a University course on image processing, both of which came handy now that I had time off. I rewrote the entire Renderscript processing pipeline in GLSL, and that brought the runtime down to an incredible 1.5s on my OnePlus 3T. On a more modern phone like the OnePlus 6, I've heard reports that it takes less than half a second. Finally, a lot of tuning was done using samples from the OnePlus 3, 5 and 6. Artifacts were removed, noise reduction was improved, curves that make the result look crunchy and colourful were tuned for every situation.

How does this app work?

It waits for new RAW images captured using a supported camera app, and then processes them in the background. It does this by using the power of your phone's graphics unit for up to a few seconds. The results have more details, higher dynamic range, and can improve over time when the application gets updated.

Only a few devices are currently supported, and everything else is assumed to be broken, either due to a lack of memory or non-functional OpenGL drivers. If your phone does miraculously work with DNG Processor, please report it so I can add it to the list of supported devices. To make sure background processing works, whitelist the app in your battery optimization list. This is especially important on OnePlus devices, as I know that not doing this will break the automatic processing on OnePlus 5 and 5T.

Why not Google Camera?

There are two reasons why using this app might be preferable over using Google Camera. The first reason is that the ports all have their own bugs and kinks, while the camera app included on the device usually works well in any situation, albeit with lesser quality. If something does go wrong in DNG Processor, the RAW file remains, so your shot is never lost. The second reason is that this app is fully open source and will get better over time through a community effort. People with image processing experience can take a look at the code and contribute, since there's no obfuscation.

Tuning

I focused on having natural colour tones and retaining all the details possible, especially in the foliage. It can be a bit noisy, but there is no watercolour effect in my images. Some might find the results a bit washed out, and I agree the saturation can be a bit low in some scenes. It's hard to balance these things, but I tried to balance it as well as I could.

Techniques

Gradient-based demosaicing

Many people do not know that the amount of megapixels a sensor reports is not equal to the amount of data the final image contains. A 16 MP sensor effectively only has 4 MP of RGB pixels, since 4 values need to be combined for one RGB pixel. However, there are smart tricks to interpolate the 16 MP data to get a 16 MP output image that looks very close to what it would have looked like if the sensor had captured 16 MP of full RGB pixels.

<https://hal.archives-ouvertes.fr/hal-00683233/document>



Left - Bilinear interpolation

Right - Gradient-based interpolation

Luminosity extrapolation

Green sensor data is important because it determines the perceived brightness of a pixel. In the highlights, when green sensor data clips the real value is extrapolated based on the whitepoint and the red and blue sensor data. In this way, green values above 1 can be detected and are valid. Later in the pipeline this is accounted for, and they are mapped into a visible range.



OxygenOS Camera - **Limited range**



DNG Processor - **Extended range**

Chroma noise reduction

A makeshift algorithm searches for the best path to reduce noise by finding similar colours and averaging over them. In low light the strength is increased to remove coloured speckles from the image as much as possible. On very noisy patches, desaturation is applied as a final step in the noise reduction script.



DNG Processor - Noise reduction disabled



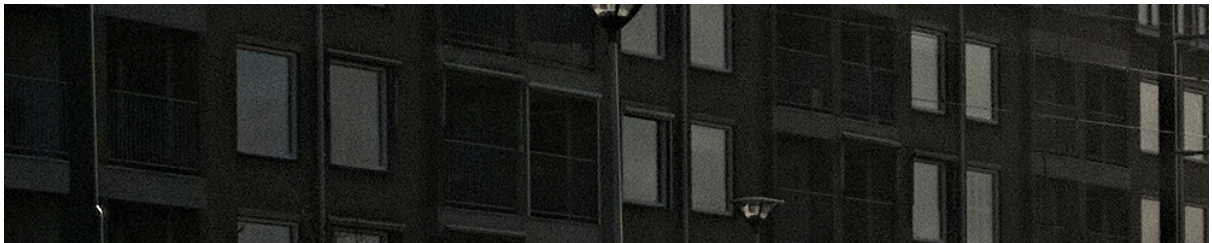
DNG Processor - Noise reduction enabled

Sharpening

This is a simple technique that brings out texture details and highlights edges. Sharpening is reduced when the noise level of the image is high, to prevent noise from taking over the image.



DNG Processor - Unsharpened



DNG Processor - Sharpened

Histogram equalization

Histogram equalization is done by analyzing samples from the intermediate to build an intensity histogram. Consecutive values are added to build the cumulative histogram, which is supplied to the processing pipeline as a mapping texture. With this technique, the full range of intensity values is used, by distributing the intensities evenly over $[0, 1]$.

https://www.tutorialspoint.com/dip/histogram_equalization.htm



DNG Processor - Post processing disabled, no histogram equalization



DNG processor - Post processing natural, 25% strength histogram equalization

Local contrast enhancement

Adaptive local contrast enhancement is a popular technique to give the image more depth by adding stronger contrast in local structures while preserving edges. Implementing this efficiently is done with a two-pass algorithm: both steps blur the previous image but one does it in the horizontal direction and the other in the vertical direction. The blurred image can be used to determine how much any intensity in the final image should be changed.

<http://www.dl-c.com/Temp/downloads/Whitepapers/Local%20Contrast%20Enhancement.pdf>



DNG Processor - Local contrast enhancement disabled



DNG Processor - Local contrast enhancement enabled

Sigmoid mapping

Because luminosity after green extrapolation, sharpening, local contrast enhancement and histogram equalization can be far above the maximum visible 1.0 value, a way to map everything back into a visible range is needed. A sigmoid function is the perfect tool for this job. It can transform an input from $[0, \infty]$ to $[0, 1]$ while preserving the lower values.

<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

Hue-saturation map

For a more punchy and intense photo, the saturation of pleasant colors (green and blue) is increased, while the saturation of unpleasant colors (yellow and purple) is kept low. Some saturation values might go above 1.0, so another sigmoid function is used to get the saturation back in range.



DNG Processor - **No saturation changes**



DNG Processor - **Natural saturation**

Tone mapping

Devices only have a limited range of brightness values, and the peak is determined by the device's brightness setting. Tone mapping helps in giving viewers the illusion of a larger dynamic range by smartly mapping intensity values, while keeping the hue the same.

<http://filmicworlds.com/blog/filmic-tonemapping-operators/>

https://www.csie.ntu.edu.tw/~cyy/courses/vfx/13spring/lectures/handouts/lec04_tonemapping_4up.pdf



DNG Processor - No tone map



DNG Processor - Google's post-gamma ACR3 tone map

Device fixes

Some devices need additional fixes for the RAW processing to give good results. OnePlus 5 has the dot pattern, so some pixels have to be disabled and interpolated using the surrounding pixels. Its forward and colour matrices are also incorrect, so those have been hard-coded using values from Savitar.

Samples

[Google Drive - DNG Processor 3/25](#)

Device list

Working

OnePlus 6, 6T, 5, 5T, 3, 3T
Xiaomi Poco F1
Samsung Galaxy S10, S7 (Exynos)
Samsung Galaxy Note 9 (Exynos)
HTC U11

Needs root to enable HAL3

Moto G6, G5+ G4+
Xiaomi Redmi Note 5, 4
Xiaomi Mi A1
Lenovo ZUK Z2, Z2 Plus
ASUS Zenfone Max Pro M1
Wiko U Feel Prime

Broken

Moto G1

Contact me on Telegram if you want to add a phone to these lists: @mirai