Preservation of Color Metadata Across Transformations of Pixel Data

Coleman Earlywine and Henry Dietz
Department of Electrical and Computer Engineering, University of Kentucky; Lexington, Kentucky

Abstract

Image data is now commonly represented in any of a myriad of file formats, from proprietary raw formats through standards like JPEG or PNG. In addition to recording pixel values, most of these formats contain metadata that defines how pixel values translate into colors and brightnesses. Unfortunately, when non-trivial computations are used to transform pixel values, it is often necessary that both the metadata and file format used be changed. For example, using a program called `parsek`*, this tool aligns images and produces a "raw" super resolution result which has problematic shifts in color and tonality. The goal of the current work has been to understand what is causing these shifts and how they can be avoided. Toward that goal, the color and tonal metadata information representations in a variety of file formats is reviewed. The effectiveness of preserving appearance when file format is changed is then reviewed. A reference color and tonal rendering is obtained by examining specific metadata and preview renderings embedded in several file formats.*

Introduction

The primary motivation for the current work was an issue exposed in an EI2024 introducing `parsek`, the Probabilistic Alignment Raw Stitcher Experiment from Kentucky[3]. This software can take multiple input files in any format, but preferably accepts raw formats, aligns the images, and then uses a new statistical method to compute a super-resolution image. If the input is raw pixel data, it is maintained in the raw colorspace and linear gamma throughout `parsek`'s computations to maximize accuracy. Thus, although the output of `parsek` has values for all three color channels at each pixel location whereas most raw format images only have one channel per pixel following a CFA pattern, the super-resolution integration of multiple raw images is a raw image. However, none of the raw image formats in common use have a simple process for writing new pixel data while preserving other metadata, especially color and tonal information. Thus, `parsek` essentially discards the color and tonal information from the raw input files, producing raw output as a 16-bit PNG file which contains no useful metadata. The disturbing result is that default rendering of the `parsek` output looks wildly different from that of one of the original raw images, as seen in Figure 1. Of course, the raw pixel data is there to allow high-quality correction of the color and tonality, but that is awkward at best.

This problem is not unique to `parsek`. For example, `kremy`, KentuckY Raw Error Modeler[4][5], inputs a single raw image and outputs a revised raw image which has lower noise and slightly enhanced sharpness. The catch is that most raw file formats are not editable. However, `kremy` solves the problem by leveraging the fact that it is possible to in-place edit uncompressed raw data in a DNG file[6]. Thus, `kremy` invokes Adobe Digital Negative Converter (hence referred to as **adc**)[7] to convert whatever raw file is supplied into an *uncompressed* DNG. The data within the uncompressed DNG file is essentially an array, which `kremy` edits in-place without altering any metadata in the DNG file. The in-place editing uses logic from `dcraw`[8] to read the metadata and locate the pixel array to edit, and is constrained to make edits only changing the values of pixels, not image dimensions, number of color channels, nor even pixel value bit depth.

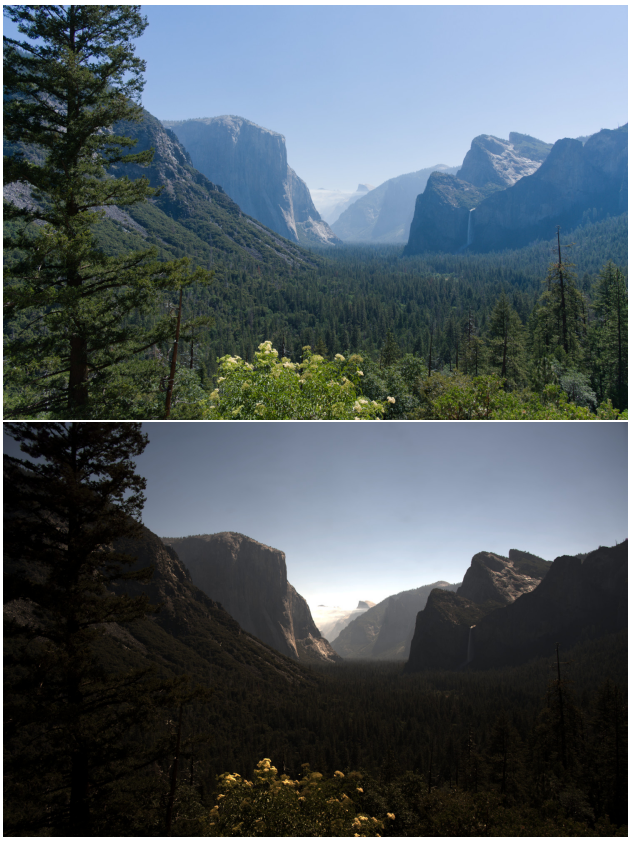

Figure 1. *Default rendering of one Sony ARW file and* `parsek` *output*

Unfortunately, each new camera makes a few, typically unimportant, changes to the raw file metadata it generates and **adc** rejects any raw from a camera it does not fully recognize. However, there are various other tools freely available that can be used to convert a raw into an uncompressed DNG. One such tool is OnlineConverter[9], which operates via a web interface. Better for integration with software like `kremy` is an open-source tool:

raw2dng[10], which is built on top of LibRaw[11]. The question thus becomes: do DNGs converted from various other raw formats preserve the image data and metadata properties?

The following section briefly discusses the key issues with various image file formats. Given that DNG seems the obvious answer for a flexible, portable, editable, raw format, we attempt to empirically answer the fundamental question: does raw conversion to DNG lose information? The current work concludes with a brief summary of how raw-to-raw transformations might be handled using DNGs.

Proprietary raw Formats and DNG

Most manufacturers of high-end digital cameras define their own file formats for encoding "uncooked" – raw – sensor data. More precisely, all the raw formats used by major camera manufacturers are internally structured as TIFF files, providing a standard way to access fields. However, the field names and encodings used vary significantly across manufacturers.

The raw format Sony developed for their Alpha cameras, .ARW, optionally uses a particularly unusual type of compression to achieve a constant compression factor. This allows in-place editing of pixel values within the compressed representation without needing to decompress the complete image[12]. The main motivation for this compression, which is enabled by default, appears to be that it can be applied on the sensor chip, thus allowing Sony cameras to achieve significantly higher bandwidth in transferring pixel data to the camera's memory. The compression therefore enables higher capture framerates. The metadata encoding other attributes of the capture, from shutter speed to black point and color-correction data, are also somewhat specific to Sony. They are understood by the software that Sony packages with their cameras, but precisely how these fields should be interpreted is not openly documented, so the processing must be at least partly reverse-engineered to be supported by other software.

Similarly, Nikon cameras use Nikon Electronic Format (.NEF) encoding within a TIFF file with their own choices of field names and encoding details. The format is fully understood by Nikon Capture NX-D, but again the details of how the encoded data should be interpreted are undocumented. Canon also has their own raw formats, such as Canon Raw version 2 (.CR2), again with encoding and decoding details not publically documented. The encoding used by Canon is also a little unusual, in their case because it appears to leverage the JPEG compression hardware in their cameras[8]. The developer of dcraw[8], Dave Coffin, was originally motivated to create his software by the fact that he otherwise could not read the raw files produced by his Canon camera; soon, he was supporting a multitude of different raw formats. While dcraw is a stand-alone C program, code and algorithms from it have been applied in creating most software that can read multiple raw file formats, including those using the LibRaw[11] library – and Adobe's Digital Negative Converter[7].

Adobe introduced the Digital Negative (DNG) format in 2004 as a potentially open standard to address the interoperability challenges posed by proprietary raw formats[6]. DNG offers several distinct advantages over proprietary raw formats used by cameras. DNG standardizes how raw image data can be encoded and defines a set of metadata field names that can encode the other properties. DNG supports both uncompressed storage of pixel values and a lossless compression scheme that can reduce file size without sacrificing image quality. Additionally, DNG supports encoding of metadata compliant with any of several standards (discussed later in this paper). This ensures that critical information about each capture and the processing history can be preserved and accessible. Adobe argues that use of DNG is future-proofing your captured images so that future software will still be able to process your images even without manufacturer support. It is also possible for the final image quality to be improved by re-processing DNGs using new post-processing algorithms that did not exist when the images were captured.

It is worth noting that not all processing that logically produces a raw image necessarily begins with raw images. For example, parsek can be given JPEG or PNG files as input. Even with such output-oriented input file formats, it can take into account the original CFA pattern, so that it only trusts the pixel color channel data where the corresponding sensel sampled that color channel. It is easy to imagine similar methods generating raw image content from a video sequence. Generating DNG image data from such inputs is reasonably straightforward, but determining how to preserve metadata may be even more ponderous than with raw conversion to DNG.

Is raw Conversion to DNG Lossless?

While the need for a standard like DNG is clear and compelling, the practical question remains: is raw conversion to DNG lossless? For the conversion to be lossless, two properties must hold:

1. Conversion from a raw format into DNG must preserve image data
2. Conversion from a raw format into DNG must preserve metadata properties

The first question is not asking if DNGs use a *lossy encoding* to achieve compression. Lossy compression is an option that can be disabled in most DNG converters. Rather, it is asking if the numeric values of pixels are identical in the original raw and DNG.

To determine if pixel values and metadata are preserved by conversion into a DNG, raw images were converted using a variety of software tools and option settings. In the current work, we will focus primarily on the conversion of a single camera raw file produced by a Sony ILCE-7RM5 – Sony's currently highest-resolution, and highest image quality, camera. The conversion cases to be compared are:

- **arw**: This gives properties of the original Sony .ARW raw capture, and is thus the ground truth for comparisons. Note that Sony actually has several different formats that can be used for a .ARW file, and the one used here is the version that records uncompressed 14-bit sensel readings. The particular file used here consists of 132,108,288 bytes of image data and metadata – the same file used to produce the JPEG top image of Figure 1.
- **adc**: Properties of the compressed DNG produced by Adobe Digital Negative Converter are given. To be precise, Version 17.2.0.2155 of Adobe DNG Converter was used on a Windows system – which was awkward in that all other work reported here is done using systems running Linux. Only Windows and macOS are directly supported by Adobe. The

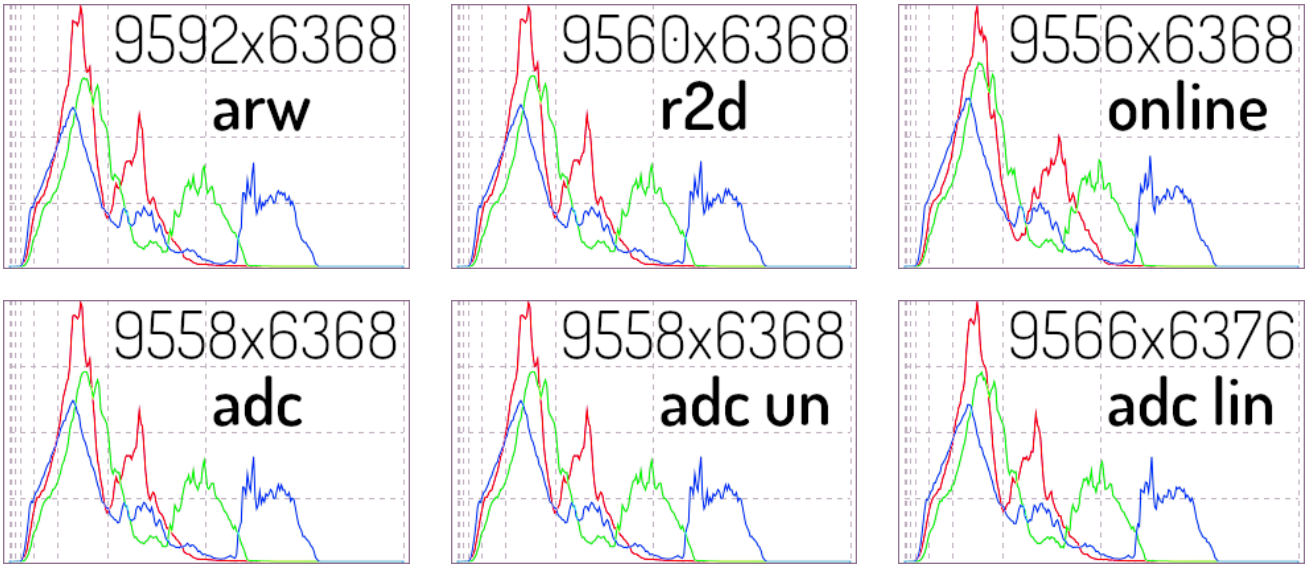

Figure 2. *Linear raw histograms of the ARW file and DNG versions*

compression used in the generated DNG is stated to be lossless. The compression is effective, with this file containing just 60,670,166 bytes, which is less than one byte per pixel despite encoding 14-bit sensel readings.

- **adc un**: Adobe Digital Negative Converter is also capable of producing images encoded in an uncompressed DNG format. Although at this writing it is rarely done, this format is by far the most amenable to in-place editing of the image data as is done by `kremy`. The file size is 122,886,428 bytes, which is smaller than the original uncompressed ARW file.
- **adc lin**: Although most cameras use CFA (color filter arrays) that mean each pixel location samples only one color channel, Adobe Digital Negative Converter offers the option of producing a raw file in which colors have been interpolated to provide values for Red, Green, and Blue at each pixel location. The demosaicing interpolation is intended to simplify the task for programs reading DNG files at the expense of a larger file size, and this file is 366,917,306 bytes.
- **r2d**: The open-source `raw2dng` program gives a way to embed conversion to DNG in user-written program without using a helper application. Although perhaps not as finely tuned as Adobe's converter, it produces a slightly smaller file for this example at 60,075,074 bytes.
- **online**: Online Converter is a web-interfaced free file conversion service handling a multitude of file format conversions, not just conversions to DNG. Using it on this image produced the smallest file in this case, containing just 60,074,878 bytes.

The following two subsections explore the extent to which image pixel data and metadata are preserved in conversion to DNG.

Is Image Data Preserved?

To determine if image data is preserved, the five DNG conversions described above were performed on the sample ARW image. Figure 2 summarizes what happened to the pixel data.

The image data captured was 9600×6376 pixels, but as is typical for digital cameras, some pixels are deliberately shaded from exposure to serve as a black reference. Thus, the actual image size delivered is generally somewhat smaller than the sensel count. It can be lossless to record fewer pixels in the file if only black reference border pixels are removed. For example, the **arw** rendering is 9592×6368, which suggests a border region of no more than 8 sensels. However, **adc**'s image data contained just 9566×6376 pixels of which it would only render 9558×6368. Examining the images, the **arw** view is slightly wider on the right side. Clearly, *image data is not preserved because even the pixel counts differ in all the DNGs*. It is particularly strange that **adc** and **adc lin** do not use the same dimensions. Because JPEG DCT compression works on 8×8 blocks of pixels, it is fairly common that image file conversions will set pixel counts to multiples of 8 by either ignoring or adding rows/columns; in the dimensions seen here, only 9556 and 9558 are not multiples of 8.

Where pixels were not omitted, the pixel values generally look very similar. To confirm this, the raw linear histograms were examined using `RawTherapee`[13]. As the graphs in Figure 2 show, there are at least minor differences between the reference `arw` histogram and all others. With the exception of **online**, these differences are not particularly concerning. The Red bias in **online** is as obvious in the rendered image as it is in this graph. Perhaps that converter was attempting to compensate for the relative lack of reds in the scene photographed? Some small changes in pixel values may be the result of attempting to convert between different color standards, such as sRGB vs. Adobe RGB colorspaces or even slightly different approximations to sRGB (the `.ARW` and DNG files here all identify themselves as sRGB).

Is Metadata Preserved?

Metadata, which includes information such as camera settings, geolocation, copyright details, and timestamps, plays a vital role in photography. It enhances the value of an image by providing contextual and technical details essential for editing, cataloging, and intellectual property management. Most significantly for applications like `parsek`, it is the metadata that allows

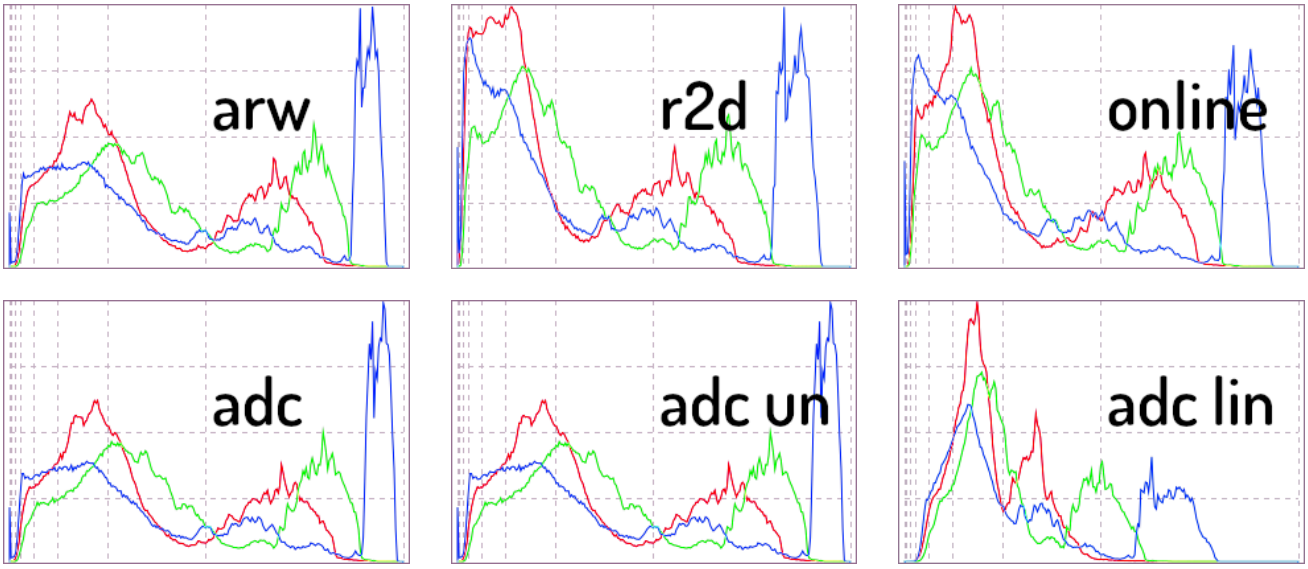

Figure 3. *Histograms of Auto-Matched Tone Curves*

accurate rendering of color and tonality.

However, during the conversion process between file types, metadata is often lost or improperly transferred. This loss can occur due to differences in how file formats handle metadata, software limitations, or user error. For example, converting a raw file to JPEG might strip nuanced metadata details, while some software tools fail to recognize or export specific metadata fields. Such issues not only disrupt workflows but can also compromise the integrity and usability of photographic assets. As photographers and industries increasingly depend on digital images, addressing these challenges becomes imperative to ensure the accurate preservation and transfer of metadata across formats. There are three main sections of current metadata in images, each serving a distinct purpose in cataloging and describing digital photographs. The first one is EXIF, which stands for Exchangeable Image File Format. This section defines all of the information about the camera at the time when the photo was taken, including crucial technical details such as aperture settings, shutter speed, ISO sensitivity, focal length, and even GPS coordinates if the camera has this capability.

The International Press Telecommunications Council, IPTC, provides yet another set of rules for encoding metadata. This standard focuses on providing descriptive information of the image, which includes titles, captions, and copyright information. IPTC metadata is particularly valuable for professional photographers, journalists, and media organizations as it helps track and manage image rights and usage. It also can include keywords, contact information for the photographer, and detailed descriptions of the image content.

Lastly, there is XMP, developed by Adobe, which stands for Extensible Metadata Platform. This standard allows for the addition of metadata tags beyond the implementation of EXIF and IPTC tags. XMP is particularly flexible and can store information about edits made to the image, color profiles, and custom metadata fields. The beauty of XMP lies in its extensibility, allowing photographers and organizations to create custom metadata schemas that suit their specific needs.

There is a lot of metadata in most image files and the test case here is not exceptional. Using `ExifTool`[14] to examine the metadata, there are 318 metadata fields recorded in the **arw**. Of those, roughly 1/3 were maintained intact in each of the DNG versions. The complication is that the multiple standards, and options within them, allow the same information to be conveyed in a multitude of ways. Thus, it is very difficult to say how much metadata is lost as opposed to transformed.

One way to somewhat quantitatively evaluate how well metadata regarding color correction and tonal properties is preserved is to allow a raw developer to apply correction based on the metadata, which may include a corrected thumbnail image usable as a correction reference. This was done using `RawTherapee` and the histograms of the images as rendered are shown in Figure 3. These histograms show much more significant differences than were seen in the raw linear histograms of Figure 2, which suggests that the color correction and tonal metadata is corrupted significantly more than the raw pixel data is. It is significant that **adc lin** was essentially left uncorrected by this process; the histogram is identical to the raw linear one. Only the differences between the **arw** and **adc** or **adc un** histograms are small enough for the rendering to be considered successful.

Another way to view these differences is to look at some metadata fields that play key roles in the rendering. Table 1 summarizes the “Color Matrix” and “Black Level” field values in these images. Although the **arw** “Color Matrix” has dramatically different values from the “Color Matrix 1” and “Color Matrix 2” fields used by **adc** and **adc un**, and the “Black Level” settings are wildly different, the renderings differ less.

Toward a Solution

Eventually, conversion to DNG may become a viable way to losslessly move both pixel data and metadata into a standardized form. There is good reason to believe that the complexity and lossiness of conversion are rooted in the difficulty of reverse-engineering the interpretation of random raw formats. If so, as reverse-engineered solutions become more polished and manu-

facturers slowly converge on using more standardized encodings, programs that process raw images may be able to use DNG for both inputs and outputs without encountering unfortunate surprises – like the `parsek` rendering shown in Figure 1.

However, there is a problem that does not address: the lack of tools for editing one or more DNGs to produce a new raw image in DNG format.

Editing DNG Data

It should be relatively simple for tools like `karwy` and `parsek` to operate on raw (DNG) data and generate a DNG that preserves the unchanged properties of the initial raw data and metadata. Since many metadata properties are simply inherited from the original raw format, specifying which ones to copy is fundamentally infeasible; the specification should be of which ones *cannot* be copied. Ideally, this process should also add XMP metadata documenting how the DNG was generated.

As discussed earlier, `karwy` does edit DNGs efficiently by in-place overwriting of the pixel data within an uncompressed DNG. However, that editing is too constrained. Things that should be editable, but cannot be edited in-place as `karwy` does, include:

- **Image dimensions**: this is the fundamental necessity for `parsek` and other image super-resolution tools
- **Number of color channels at each pixel site**: again, super-resolution tools like `parsek` will generally produce fully-populated color channels at each pixel location; this also happens with many AI enhancements
- **Bit depth per pixel color channel**: in any scheme where averaging is applied, there is naturally an increase in dynamic range that may require additional bits per value

In theory, there is nothing preventing creation of a software infrastructure allowing easier and more flexible creation of new raws by editing of existing raws. Perhaps the most productive route would involve using the editing abilities of `ExifTool`, which are freely available for use via a C++ library interface.

Work-Arounds

The transformations required to preserve correctness of metadata involving color and tonality are the most critical for most applications, but are among the most difficult to implement[1]. That is largely because they are expressed in a very compact way, essentially as parameters to a mapping formula.

`RawTherapee`'s auto-matched tone curve is a very pragmatic way to approximate getting color and tonality correct. Rather than just using the relevant numeric metadata, this approach makes use of an embedded JPEG preview image in the raw or DNG file. Based on the assumption that the JPEG preview has the desired color and tonality, `RawTherapee` attempts to adjust the brightness, contrast, and midtone levels to approximate the preview.

Unfortunately, it is not necessarily true that the preview image has the correct color and tonality because it is generally reflecting the camera JPEG preferences. Thus, for example, a JPEG setting selecting vibrant colors will result in a preview image with vibrant colors rather than the ideal mapping of the raw pixel data to "pure" sRGB. This trick is also limited by the fact that JPEG preview images suffer various significant artifacts from mapping into a non-linear-gamma YUV colorspace and suffering lossy compression. It is also common that the preview will have various corrections applied, for example, for lens distortion, causing misalignments with the uncorrected raw data. The act of converting to DNG may also cause JPEG previews to be stripped, re-sized, re-compressed, etc., causing further inaccuracies.

However, the basic concept of basing correction on an image that was created from the original mapped into the correct colors and tonality is fundamentally sound. Even if such an image is not embedded in the DNG, but provided separately, it could be used to compute a CLUT (Color Look-Up Table) that could be used to correct the color and tonality of the edited raw. Alternatively, a computed CLUT could be embedded in the DNG metadata or applied separately.

Thus far, the most effective work-around we have implemented involves using `ExifTool` to replace the preview image in a DNG with one that is less compressed and more accurately represents the desired appearance obtained from the original. For example, using `RawTherapee`'s auto-matched tone curve, this approach is even reasonably effective using the rendering of one of the original images as the preview of the super-resolution image created by `parsek` wrapped as a DNG. Of course, that only is effective because images created by `parsek` are intended to have very similar color and tone to any of the individual original raws that were combined by `parsek`. For example, it would be much more difficult to make this approach work for a tool producing raw panoramas by stitching multiple captures. Any raw editing that significantly alters the image content is problematic: consider color and tonality when an AI tool performs sky replacement.

Table 1: Color Matrix and Black Level Coding

File	Color Matrix Fields	Black Level
arw	Color Matrix: 1191 -194 27 145 1038 -159 55 -54 1023	Black Level: 512 512 512 512
adc, **adc un**	Color Matrix 1: 0.9185 -0.4857 0.0505 -0.3651 1.1061 0.2982 -0.0161 0.0698 0.6769 Color Matrix 2: 0.82 -0.2976 -0.0719 -0.4296 1.2053 0.2532 -0.0429 0.1282 0.5774	Black Level: 0 206 4041
adc lin	Color Matrix 1: 0.9185 -0.4857 0.0505 -0.3651 1.1061 0.2982 -0.0161 0.0698 0.6769 Color Matrix 2: 0.82 -0.2976 -0.0719 -0.4296 1.2053 0.2532 -0.0429 0.1282 0.5774	Black Level: 0 247 3907
r2d	Color Matrix 1: 0.4913 -0.0541 -0.0202 -0.613 1.3513 0.2906 -0.1564 0.2151 0.7183	Black Level: 512 512 512 512
online	Color Matrix 1: 0.82 -0.2976 -0.0719 -0.4296 1.2053 0.2532 -0.0429 0.1282 0.5774	Black Level: 512 512 512 512

References

[1] D. Andrew Rowlands, "Color conversion matrices in digital cameras: a tutorial," Optical Engineering, Volume 59, Issue 11, 110801, November 17, 2020, doi: 10.1117/1.OE.59.11.110801

[2] https://www.onlineconverter.com, accessed 2/28/2025

[3] Henry Dietz, "Leveraging Pixel Value Certainty in Pixel-Shift and Other Multi-Shot Super-Resolution Processing," in Electronic Imaging, 2024, pp 142-1 - 142-7, doi: 10.2352/EI.2024.36.15.COIMG-142

[4] Henry Gordon Dietz, "Refining raw pixel values using a value error model to drive texture synthesis" in Proc. IS&T Int'l. Symp. on Electronic Imaging: Image Processing: Algorithms and Systems XV, 2017, pp 56 - 66, doi: 10.2352/ISSN.2470-1173.2017.13.IPAS-084

[5] Henry Dietz, "An improved raw image enhancement algorithm using a statistical model for pixel value error," in Proc. IS&T Int'l. Symp. on Electronic Imaging: Computational Imaging, 2022, pp 151-1 - 151-6, doi: 10.2352/EI.2022.34.14.COIMG-151

[6] Adobe, "The public archival format for digital camera raw data," https://helpx.adobe.com/camera-raw/digital-negative.html, accessed 3/2/2025

[7] Adobe, "Adobe Digital Negative Converter," https://helpx.adobe.com/camera-raw/using/adobe-dng-converter.html, accessed 3/2/2025

[8] Dave Coffin, "Decoding raw digital photos in Linux," https://www.dechifro.org/dcraw/, accessed 3/2/2025

[9] Online Converter, https://www.onlineconverter.com/raw-to-dng, accessed 3/2/2025

[10] Fimagena, "Linux utility for converting raw photo files into DNG, TIFF or JPEG format," https://github.com/Fimagena/raw2dng, accessed 3/2/2025

[11] Iliah Borg and Alex Tutubalin, "About LibRaw," LibRaw LLC, https://www.libraw.org/about, accessed 3/2/2025

[12] Henry Dietz, "Sony ARW2 Compression: Artifacts And Credible Repair," in Electronic Imaging, 2016, doi: 10.2352/ISSN.2470-1173.2016.2.VIPC-227

[13] RawTherapee, https://www.rawtherapee.com/, accessed 3/2/2025

[14] Phil Harvey, "ExifTool," https://exiftool.org/, accessed 3/2/2025