Reconfiguring the Imaging Pipeline for Computer Vision: Supplemental Material

Mark Buckler
Cornell University

Suren Jayasuriya
Carnegie Mellon University

Adrian Sampson
Cornell University

This supplemental material contains additional background about the traditional image signal processing pipeline, additional empirical results, and more quantitative detail behind the design decisions that inform our proposed "vision mode."

Contents

1. Background: The Imaging Pipeline

For readers unfamiliar with the ISP pipeline, we describe the standard pipeline found in any modern camera, from DSLRs to smartphones. This expands on Section 3.1 in the main paper.

We consider a complete system including a computer vision algorithm that processes images and produces vision results. Figure 1a in the main paper depicts the traditional pipeline. The main components are an image sensor, which reacts to light and produces a RAW image signal; an image signal processor (ISP) unit, which transforms, enhances, and compresses the signal to produce a complete image, usually in JPEG format; and the vision application itself.

1.1. Camera Sensor

The first step in statically capturing a scene is to convert light into an electronic form. Both CCD and CMOS image sensors use solid state devices which take advantage of the photoelectric effect to convert light into voltage. Most modern devices use CMOS sensors, which use active arrays of photodiodes to convert light to charge, and then to convert charge to voltage. These pixels are typically the size of a few microns, with modern mobile image sensors reaching sizes of 1.1 µm, and are configured in arrays consisting of several megapixels.

CMOS photodiodes have a broadband spectral response in visible light, so they can only capture monochrome intensity data by themselves. To capture color, sensors add photodiode-sized filters that allow specific wavelengths of light to pass through. Each photodiode is therefore statically allocated to sense a specific color: typically red, green, or blue. The layout of these filters is called the *mosaic*. The most common mosaic is the Bayer filter [8], which is a 2×2 pattern consisting of two green pixels, one red pixel, and one blue pixel. The emphasis on green emulates the human visual system, which is more sensitive to green wavelengths.

During capture, the camera reads out a row of the image sensor where each pixel voltage is amplified at the column level and then quantized with an ADC. A frame rate determines the time it takes to read and quantize a complete image. The camera emits a digital signal referred to as a RAW image, and sends it to the ISP for processing.

1.2. Image Signal Processor

Modern mobile devices couple the image sensor with a specialized image signal processor (ISP) chip, which is responsible for transforming the RAW data to a final, compressed image—typically, a JPEG file. ISPs consist of a series of signal processing stages that are designed to make the images more palatable for human vision. While the precise makeup of an ISP pipeline varies, we describe a typical set of stages found in most designs here.

Denoising. RAW images suffer from three sources of noise: *shot noise*, due to the physics of light detection; *thermal noise* in the pixels, and *read noise* from the

readout circuitry. The ISP uses a denoising algorithm such as BM3D [2] or NLM [1] to improve the image's SNR without blurring important image features such as edges and textures. Denoising algorithms are typically expensive because they utilize spatial context, and it is particularly difficult in low-light scenarios.

Demosaicing. The next stage compensates for the image sensor's color filter mosaic. In the Bayer layout, each pixel in the RAW image contains either red, green, or blue data; in the output image, each pixel must contain all three channels. The *demosaicing* algorithm fills in the missing color channels for each pixel by interpolating values from neighboring pixels. Simple interpolation algorithms such as nearest-neighbor or averaging lead to blurry edges and other artifacts, so more advanced demosaicing algorithms use gradient-based information at each pixel to help preserve sharp edge details.

Color transformations and gamut mapping. A series of color transformation stages translate the image into a color space that is visually pleasing. These color transformations are local, per-pixel operations given by a 3×3 matrix multiplication. For a given pixel $p \in \mathbb{R}^3$, a linear color transformation is a matrix multiplication:

$$p' = Mp \tag{1}$$

where $M \in \mathbb{R}^{3\times3}$.

The first transformations are *color mapping* and *white balancing*. Color mapping reduces the intensity of the green channel to match that of blue and red and includes modifications for artistic effect. The white balancing transformation converts the image's color temperature to match that of the lighting in the scene. The matrix values for these transformations are typically chosen specifically by each camera manufacturer for aesthetic effect.

The next stage is *gamut mapping*, which converts color values captured outside of a display's acceptable color range (but still perceivable to human vision) into acceptable color values. Gamut mapping, unlike the prior stages, is nonlinear (but still per-pixel). ISPs may also transform the image into a non-RGB color space, such as YUV or HSV [8].

Tone mapping. The next stage, *tone mapping*, is a nonlinear, per-pixel function with multiple responsibilities. It compresses the image's dynamic range and applies additional aesthetic effects. Typically, this process results in aesthetically pleasing visual contrast for an image, making the dark areas brighter while not overexposing or saturating the bright areas. One type of global tone mapping called **gamma compression** transforms the luminance of a pixel p (in YUV space):

$$p' = Ap^{\gamma} \tag{2}$$

where $A > 0$ and $0 < \gamma < 1$. However, most modern ISPs use more computationally expensive, local tone mapping based on contrast or gradient domain methods to enhance image quality, specifically for high dynamic range scenes such as outdoors and bright lighting.

Compression. In addition to reducing storage requirements, compression helps reduce the amount of data transmitted between chips. In many systems, all three components—the image sensor, ISP, and application logic—are on physically separate integrated circuits, so communication requires costly off-chip transmission.

The most common image compression standard is JPEG, which uses the discrete cosine transform quantization to exploit signal sparsity in the high-frequency space. Other algorithms, such as JPEG 2000, use the wavelet transform, but the idea is the same: allocate more stage to low-frequency information and omit high-frequency information to sacrifice detail for space efficiency. This JPEG algorithm is typically physically instantiated as a codec that forms a dedicated block of logic on the ISP.

2. Proposed Pipelines

The main paper describes two potential simplified ISP pipelines including only the stages that are essential for all algorithms we studied: demosaicing, gamma compression, and denoising. Normalized data was shown in the main paper to make more efficient use of space, but here in Figure 1 we show the absolute error for each benchmark. As depicted in the main paper, the pipeline with just demosaicing and gamma compression performs close to the baseline for most benchmarks; the outlier is SGBM, where denoising has a significant effect. OpenFace, also as discussed in the main paper, is alone in performing better on the converted images than on the original dataset.

3. Approximate Demosaicing

We find that the demosaicing ISP stage is useful for the vision applications we examine. In the main paper, we describe *subsampling* as a circuit-level replacement for "true" demosaicing on the ISP.

Here, we also consider two other lower-quality demosaicing techniques that use simpler signal processing. The two techniques are bilinear interpolation and a nearest-neighbor algorithm. Figure 2 visualizes these

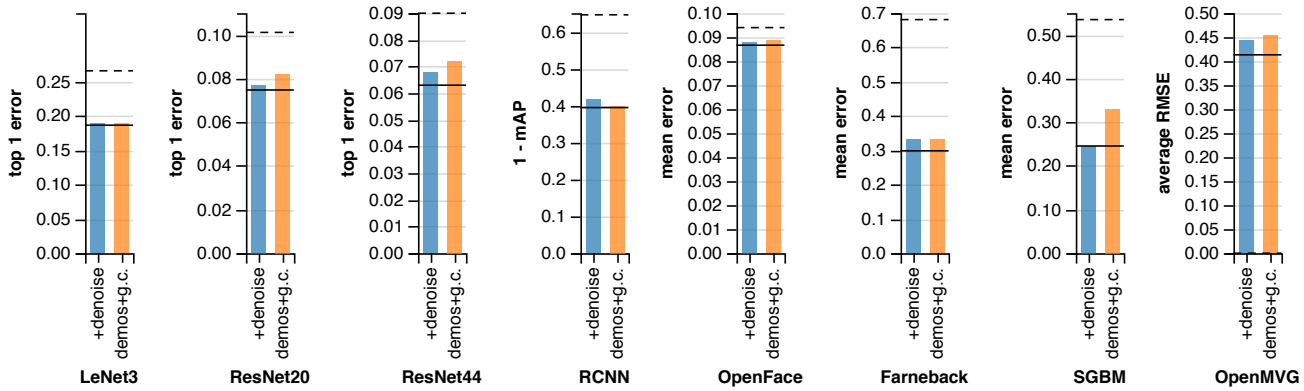
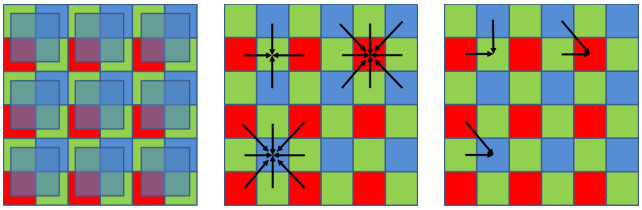


Figure 1: Vision accuracy for two proposed pipelines.



(a) Subsample (b) Bilinear (c) NN

Figure 2: Visualizations for the approximate forms of demosaicing: subsampling, bilinear interpolation, and a nearest-neighbor algorithm.

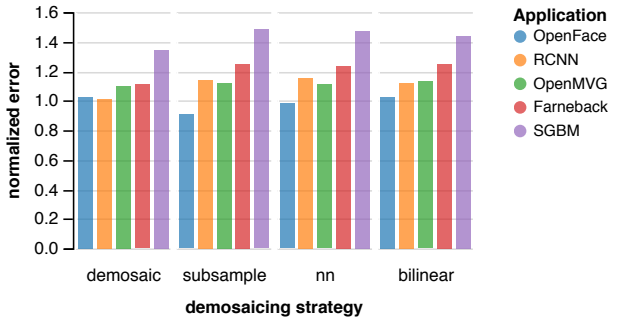


Figure 3: Normalized task error for four demosaicing strategies. Each cluster shows a configuration simulating a pipeline with only gamma compression enabled. The *demosaic* cluster shows the original demosaiced data (i.e., all stages were reversed *except* for demosaicing). The others show images with simulated demosaicing using subsampling (the strategy described in the main paper), nearest-neighbor demosaicing, and bilinear interpolation.

techniques and Figure 3 shows their results. Our subsample demosaicing fills in missing channel values from their corresponding channels in the Bayer pattern. Bilinear interpolation fills channel values by averaging the corresponding local pixels, and the nearest-neighbor technique simply copies the channel value from a nearby pixel.

Figure 3 compares the vision task performance for these techniques. All three mechanisms lead to similar vision error. For this reason, we chose the cheapest technique, which eliminates the need for any signal processing: subsampling.

4. Resolution

While the resolution of a standard mobile system's image sensor can be on the order of a megapixel, the input resolution to a state-of-the-art convolutional neural network is often no more than 300×300. For this reason, images are typically scaled down to fit the input dimensions of the neural network. While the algorithms used to scale down these images are typically edge aware, it is also possible to output a reduced resolution from the image sensor. One method of doing this is pixel-binning which connects multiple photodiodes together, collectively increasing the charge and thereby reducing the error associated with signal amplification [9].

Figure 4 shows the results of our resolution experiments we conducted with the high resolution version of CIFAR-10 dataset that we describe in the main paper. Our testing was conducted by averaging pixels in the region of the sensor which would be binned, thereby reducing resolution. Any further reduction in accuracy was performed with OpenCV's edge aware image scaling algorithm [4]. As can be seen in Figure 4 the increase in error when using pixel binning isn't remarkably large, but we find generally capturing with a higher resultion is always better if possible. This presents a tradeoff between energy used to capture the image and the error for vision tasks.

	Demosaic	NL-Means Denoise	Color Transforms	Gamut Map	Tone Map	JPEG Compress
Instructions	3.45×10^8	4.84×10^{11}	2.40×10^8	5.38×10^8	4.63×10^8	6.74×10^8
Cycles	3.62×10^8	3.06×10^{11}	2.26×10^8	8.09×10^8	4.84×10^8	2.94×10^8
Cache Refs	4.17×10^6	1.60×10^8	1.80×10^6	4.11×10^6	2.63×10^6	6.96×10^5
FP Ops	1.95×10^5	6.77×10^8	1.45×10^5	2.43×10^5	1.52×10^5	9.40×10^3

Table 1: Profiling statistics for software implementations of each ISP pipeline stage.

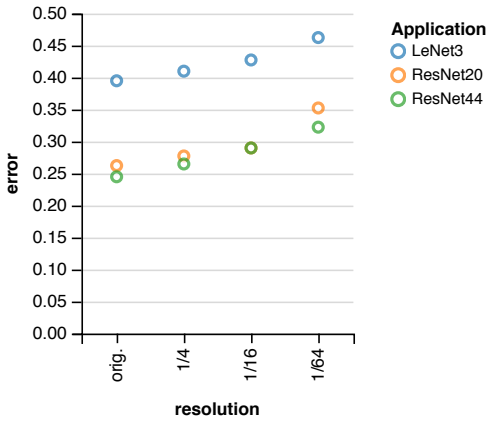


Figure 4: Impact of resolution on three CNNs for object recognition. Using a custom data set consisting of higher-resolution images from ImageNet matching the CIFAR-10 categories, we simulate pixel binning in the sensor, which produces downsampled images. The y-axis shows the top-1 error for each network.

5. Quantization

In the main paper, we present logarithmic quantization as a way to replace digital gamma compression in the ISP. As we discussed, the benefit that gamma compression provides is largely to enable a more compressed encoding to represent the intensity values by converting the data distribution from log-normal to normal. However, information theory tells us that we can achieve the minimum quantization error (and maximum entropy) when the encoded distribution is uniform [5]. So, in this section we go further by exploring the possibility of tuning quantization specifically to the statistical properties of natural scenes.

To compute the optimal quantization levels for our data, we first fit a log-normal curve to the histogram of natural images. For our experiments we used a subset of CIFAR-10 [7] which had been converted to its raw form using the CRIP tool. This log-normal curve served as our probability density function (PDF), which we then integrated to compute our cumulative density function (CDF). We then inverted the CDF to determine the

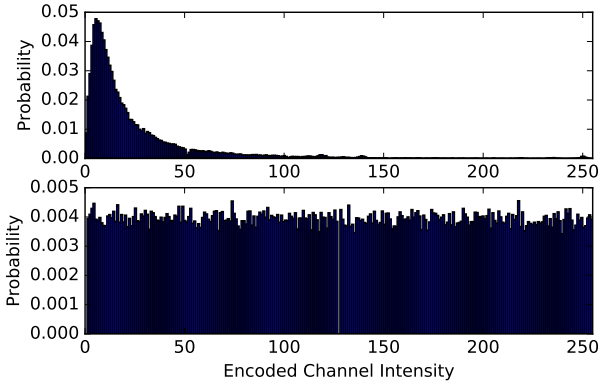
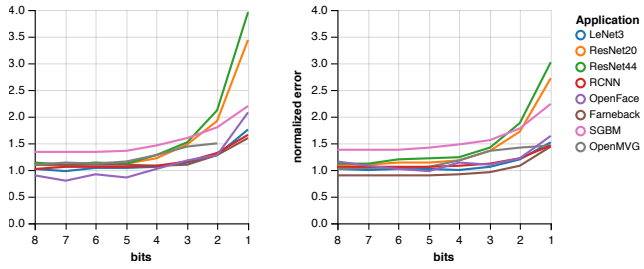


Figure 5: Histograms of the light intensity distribution for CRIP-converted raw CIFAR-10 data (top) and CDF quantized CIFAR-10 data (bottom).



(a) Logarithmic. (b) CDF-based.

Figure 6: Effect of the image quantization strategy on vision accuracy in a pipeline with only demosaicing enabled. This figure shows logarithmic quantization and a second strategy based on measuring the cumulative distribution function (CDF) of the input data. For traditional linear quantization, see the main paper.

distribution of quantization levels. Using uniformly distributed values across the CDF results in uniformly distributed encoded (digital) values. Figure 5 shows both the input and quantized distributions.

This CDF-based technique approximates the minimum-entropy quantization distribution. An even more precise distribution of levels may be derived using the Lloyd-Max algorithm [3].

Figure 6 compares the vision task performance using

this CDF technique with the simpler, data-agnostic logarithmic quantization described in the main paper. The error across all applications tends to be lower. While the logarithmic quantization strategy is less sensitive to bit-width reduction than linear quantization, the CDF technique is even less sensitive. Where 5 bits suffice for most benchmarks under logarithmic quantization, 4 bits generally suffice with CDF-based quantization.

Our main proposal focuses on logarithmic quantization, however, because of hardware feasibility: logarithmic ADCs are known in the literature and can be implemented with a piecewise-linear approximation scheme [6]. Using the CDF quantization scheme would require an ADC with *arbitrary* quantization levels; the hardware complexity for such an ADC design is not clear in the literature.

6. ISP Profiling

While total energy numbers for ISPs have been published, we are unaware of an energy breakdown per ISP stage. A promising area of future work is to simulate each ISP stage at the hardware level, but as a simpler examination of ISP stage costs, we present measurements based on software profiling. For the description of the experimental setup, see the main paper.

In Table 1, we show the number of dynamic instructions, cycles, L1 cache references, and floating-point operations needed to perform each of the ISP stages. In the table, *instructions* indicates the number of dynamic assembly instructions that the CPU executed, and *cycles* shows the number of CPU cycles elapsed during execution. When the cycle count is smaller than the number of instructions, the CPU has successfully extracted instruction-level parallelism (ILP); otherwise, performance is worsened by lower ILP or frequent memory accesses. The *cache refs* row quantifies the rate of memory accesses: it shows the number of times the L1 cache was accessed, which is an upper bound on the number of access to off-chip main memory. While most operations in a CPU use simple fixed-point and integer arithmetic, complex scientific computation uses floating-point operations, which are more expensive. The *FP ops* row shows the number of floating-point instructions executed in each stage.

With this level of detail, we see that denoising is a significantly more complex stage than the others. With this one exception, all stages require similar numbers of instructions, cycles, and cache references. The floating-point operation frequency is similar as well, with the exception of the JPEG compression stage: the JPEG codec is optimized for fixed-point implementation. We plan to explore these costs in more detail with a hardware implementation in future work, but these software-based measurements demonstrate that the implementation of the denoising stage will be of particular importance.

References

[1] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Model Simulation*, 4(2):490–530, 2005.

[2] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, Aug. 2007.

[3] M. Garey, D. Johnson, and H. Witsenhausen. The complexity of the generalized Lloyd–Max problem. *IEEE Transactions on Information Theory*, 28(2):255–256, Mar. 1982.

[4] Itseez. OpenCV. `http://opencv.org`.

[5] H. K. J.N. Kapur. *Entropy Optimization Principles and Their Applications*, volume 9. Water Science and Technology Library, 1992.

[6] M. Judy, A. M. Sodagar, R. Lotfi, and M. Sawan. Nonlinear signal-specific adc for efficient neural recording in brain-machine interfaces. *IEEE Transactions on Biomedical Circuits and Systems*, 8(3):371–381, June 2014.

[7] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[8] R. E. W. Rafael C. Gonzalez. *Digital Image Processing (4th Edition)*. Pearson, 2017.

[9] Z. Zhou, B. Pain, and E. R. Fossum. Frame-transfer cmos active pixel sensor with pixel binning. *IEEE Transactions on Electron Devices*, 44(10):1764–1768, Oct. 1997.