Decoding Consumer Grade Video for Filmmaking: Flow-Informed Artifact Removal, Chroma Upsampling, and Bit Depth Interpolation

Frankie Eder
University of California, Berkeley
https://github.com/frankieeder/DCGVF

Abstract

For independent filmmakers, achieving flexibility in post-production color grading pipelines is essential to achieving a professional grade final image, and is dependent on having high quality, high bit depth, and minimally chroma subsampled ungraded footage. In this paper, we collect a database of lossless professional grade video and train a Task Oriented Flow model that simultaneously addresses deblocking, chroma upsampling, and bit-depth interpolation. Our method takes in an H264 compressed, 8-bit, 4:2:0 chroma subsampled video, and approximates a lossless, 10-bit, 4:2:2 chroma subsampled video, "decoding" low quality footage into footage of higher intrinsic data quality. Our model outperforms all tasks of the original Task-Oriented Flow paper (Xue et al.) in both PSNR and SSIM on our novel dataset of professional-grade footage.

1. Introduction

While recent decades have seen incredible advancements in the quality and accessibility of filmmaking equipment and technology, there still exists a noticeable gap in the quality and accessibility between consumer and professional videography systems for filmmaking. One significant symptom of this discrepancy lies in the process of color correcting video footage, commonly referred to as "color grading". Namely, the image compression required to minimize the costs of in-camera image processing hardware and external storage media introduce artifacts like blocking and banding due to the necessary compression codecs,

chroma downsampling, and bit-depth quantization. The process of artistic color manipulation readily exacerbates these artifacts.

Most consumer and "prosumer" grade cameras encode video footage with 8-bit depth and 4:2:0 Chroma Subsampling, and some variant of H.264 encoding. While these standards are based on a reasonable perceptual limit of the human eye, any manipulation of the color from this standard (the manipulation often being an intrinsic part of the artistic process of filmmaking) reveals aliasing artifacts such as banding, especially in low-frequency regions such as skies, human skin, and portions of the image that are intentionally defocused due to stylistic shallow depth of field. The artifacts caused by "stretching" the color signal through color grading are especially noticeable when recording video in a (recently-popularized) "LOG"-style color space that utilizes low contrast to encode as much of the sensor's dynamic range as possible. When reconstructing the normal-contrast image, these artifacts appear immediately.

In order to allow independent filmmakers to better approximate the data quality of higher-end video systems, we propose a deep-learning approach to reconstructing high-quality video data. To utilize the spatial and temporal information in video, we utilize a flow-based approach. Our main contributions are as follows:

1. Code to produce manageable datasets from high-quality video footage for task-oriented flow training, with attention to bit depth.
2. A promising model to chroma upsample, bit depth interpolate, and deblock low-quality video footage collected in the filmmaking setting.

3. Method

3.1 Problem Statement

We can frame our problem as an image reconstruction problem. Given an input sequence of frames, encoded with 8 bits of depth, 4:2:0 Chroma Subsampling, and a lossy compression codec, we would like to construct a 10 bit, 4:2:2, lossless-codec-like video. In essence, we would like to reconstruct the "ProRes422" video file from an "MP4" (or similar) video file.

3.1 Database

To create our database, we generate a streamlined subset of a larger database of high-quality video data.

To create the dataset we use for training, we sample from a dataset of approximately 2TB cinematically captured video sequences collected with a RED Dragon 6K camera encoded in R3DRAW 3:1, a relatively lossless codec encoded with 14 bits of depth and no chroma subsampling. However, for the purposes of generalization, we transcode to Apple Prores 422, a commonly used codec in a variety of video settings where quality. The resulting quality of the video is relatively lossless, with 10 bits of depth and 4:2:2 chroma subsampling[2].

The nature of our dataset sampling is to yield a stored cache of septuplet samples of the same structure used in the denoising/deblocking tasks of Xue et al[2]. To do this, we sample 511 groups of 7 consecutive frames uniformly from our entire high quality set of videos, importing our data with 16 bits of precision. We then transcode each septuplet into our degraded image quality of 8-Bit, 4:2:0, H.264 footage, with a uniformly and randomly sampled bitrate of between .0417 and .2504 effective bits per pixel. For each of these ground truth septuplets and their corresponding degraded septuplet (each of dimension 7 x 3160 x 3792 x 3), we spatially subsample 50 samples utilizing random cropping to the size necessary to input to our architecture (yielding septuplets of size 7 x 256 x 448 x 3) . For our 6K 6:5 input footage of resolution 3160 x 3792, this samples approximately half of the pixel area of the full-resolution septuplet. From each of these random crops, we isolate the 3rd frame from our high quality frame as our target frame, and use the corresponding 7 degraded frames as the input to our network. This process generated a total of 25,550 septuplet samples that we subsequently use for training.

This process is easily adaptable to generate a dataset of arbitrary size from a set of high quality ProRes videos (of arbitrary resolution), and our code easily allows for this flexibility. In our case, this allows our model to learn imagery and optical effects common in a filmmaking setting.

It is worth noting that we attempted training by performing this transcoding on the fly in parallel, but this yielded very slow results. Because of this, we decided to opt for precomputing this dataset. A dataset of 100,000 samples serialized and compressed using NumPy only takes around 80GB of space, which was a small price to pay in comparison to the remainder of our dataset and sped up training dramatically.

3.2 Architecture

Our architecture is identical to the denoising/deblocking architecture of Xue et al[1]. We leverage a publicly available PyTorch implementation of Xue et al.[3] for ease of integrating with the rest of our contributions.

3.3 Training

We start by utilizing weights pretrained on the denoising task. Using our dataset, we train for an additional 5 epochs using an ADAM optimizer[4] with a weight decay and learning rate of $1e^{-4}$. We utilize 60% of our training set for training, 20% for validation, and 20% for testing.

4. Results

4.1 Artifact Removal

Figure 1 shows the qualitative results of our artifact removal. We can see similar deblocking performance in high frequency regions as that in Xue et al. However, as noted in the failure cases in Figure 2, some blocking artifacts remain in low frequency regions of the image. As opposed to high-frequency regions, it may be that the cinematic setting does not encourage enough flow between frames to deblock these regions.

4.2 Bit Depth Interpolation

Analysing our histograms in Figure 1, we can see that our model encourages a more even distribution of image values across our image.

Bit Depth	% Correct Pixels	% Added Precision
10	10.48	70.85
11	5.30	85.06
12	2.63	92.40

Table 1

Metrics concerning bit depth. The middle column represents the percentage of pixels in the results of our model calculated within the precision of the corresponding bit depth. The Right column represents the percentage of pixels that were calculated outside of their original 8-bit value at the corresponding bit depth, that is, the number of pixels that achieved increased precision of some sort (to the significance of the corresponding bit depth).

Table 1 also includes some additional metrics regarding the precision and accuracy of our model. We can see that a significant portion of our pixels are predicted accurate to 10 bits of significance or higher, and almost all pixels encouraged some additional precision.

4.4 Overall

Overall, as we can see in Figure 1, the reconstruction results are qualitatively comparable to Xue et al in high frequency regions. However, among our entire dataset, our numeric results proved better than Xue's results on most tasks, achieving an SSIM of .9726 and a PSNR of 39.67 on our test set.

5. Conclusion

In conclusion, Task-Oriented Flow is a promising direction for artifact removal and especially bit-depth interpolation in filmmaking tasks. Numerically, we outperform most original Task-Oriented Flow applications, even when using roughly a third of the training data compared to Xue et al. To address any underperforming qualitative results, there are a few promising directions for future work.

5.1 Future Work

Other possible directions for future work include a resolution-invariant model that can process our entire input frames at once instead of needing to apply patch-wise. Another option would be to explore a more efficient patch-wise model.

We also may want to consider training on a high-quality edited cut of a film instead of an entire dataset of unedited footage, since unedited footage in the filmmaking domain has little to none of the motion necessary to establish defined optical flow. An already edited film would not only result in more motion necessary for our model, but also encourage our model to learn the exact motions common in the filmmaking setting. Another option would be to encourage a more robust method that utilizes a different method of reconstruction in low-frequency or low-motion regions of the image, to specifically address common artifact regions such as skies and skin tones.

References

[1] T. Xue, B. Chen, J. Wu, D. Wei, W. Freeman. Video Enhancement with Task-Oriented Flow.

[2] Apple, Inc. Apple ProRes White Paper, January 2020.

[3] https://github.com/Coldog2333/pytoflow

[4] D. P. Kingma, J. Ba. Adam: A Method for Stochastic Optimization,

Appendix

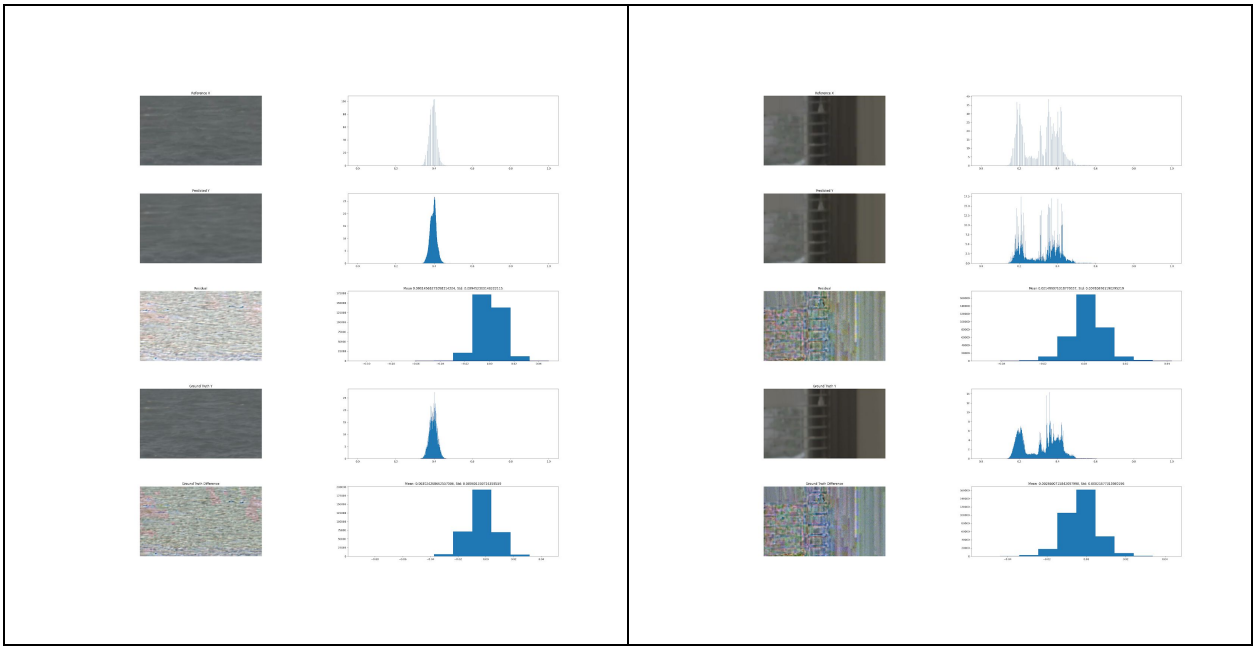

Figure 1

Success Cases for our model. In the left column, from top to bottom: input reference image (the middle frame of our septuplet), our predicted output, our residual image, our ground truth reference image, and our ground truth difference between input and output. In the right column are corresponding histograms: our images include a 10 bit histogram, while our residual and ground truth difference images have a more coarse histogram. We can see that our model encourages sparsity in our predicted output, and in places of high motion, such as the first one of the sea, we get very good qualitative and quantitative results.

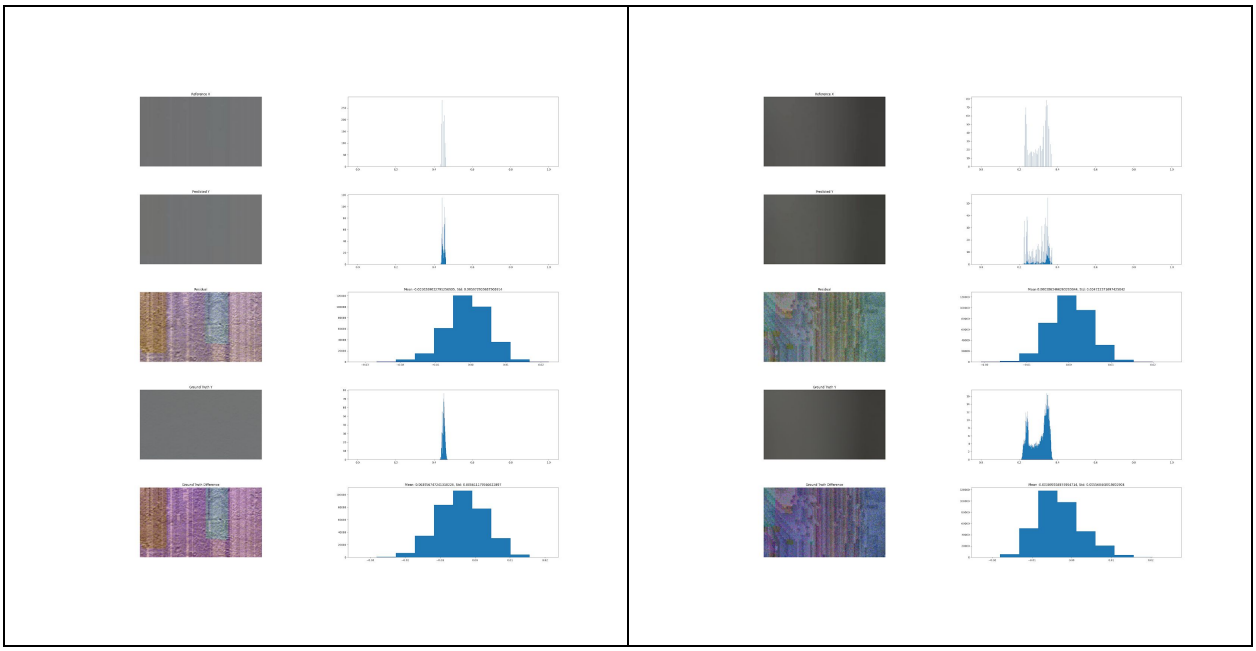

Figure 2

Failure cases for our model. In the left column, from top to bottom: input reference image (the middle frame of our septuplet), our predicted output, our residual image, our ground truth reference image, and our ground truth difference between input and output. In the right column are corresponding histograms: our images include a 10 bit histogram, while our residual and ground truth difference images have a more coarse histogram. We can see that our model still encourages more even distribution of pixel values, suggesting the success of our bit-depth interpolation task. However, in these low frequency, low motion regions, we achieve little artifact removal.