David Kennedy

Computational Photo Final Project Report

I implemented the denoising step of the scheme for improving underexposed videos to a watchable state outlined in the paper “Video Enhancement Using Per-Pixel Virtual Exposures” [1].

I first determine a target gain ratio for each pixel. The gain ratio is an estimate of what the original value of that pixel should be multiplied by to get its estimated final value. Gain ratios varied between 1 and 9. This gain ratio is then passed to my ASTA filter. The ASTA filter first runs a temporal filter on the luminance channel, attempting to average together “gain ratio” temporal pixels for each pixel. At any pixel where there was movement and thus you can not average together “gain ratio” temporal pixels, the result from the temporal filter is then passed to a spatial filter. If the temporal filter left us just a little short of the target gain ratio, the spatial filter will run the opencv bilateral filter with a kernel with small diameter that will not average together pixels when they are very different. If we are left very short, we will use a bilateral with a latger diameter and larger range of pixel differences accepted. Finally, I run a spatially uniform tone mapping log function as described in the paper on just the luminance channel.

My program takes about 20 seconds to process one frame. While there are some improvements that could be made to this running time, this is much faster than my original times using sequential rather than parallel array operations. These times were over 3 minutes per frame. This final rate makes my program time consuming but usable.

Results:

At first, I did not see a whole lot of improvement in my videos from just the denoising step. I think this is because I did not have sufficiently noisy input videos. The footage produced for me was made with a very good camera, and while all the shots were sufficiently underexposed (luminosity usually under 50 for every pixel), they were not noisy. Possibly if the ISO was set much higher and video was shot in even lower light conditions, I could have got much more noisy input video. When I cranked up the neighborhood distance metric on the temporal filter so that it accepted very different pixels to average together as expected, I got very intense ghosting artifacts, proving to me that my program was working or almost working.

I tested my results by applying a spatially uniform tone mapping function to the results of the denoising step. This was easily the most impressive part of the project. Details came out of the background and right side of the frame in a video where these regions were underexposed whereas the left foreground was properly exposed. You can see these results below:

My results show that after the tonemapping step, the colors look washed out and a lot less vibrant. The colors looked fine before tonemapping. I am only tonemapping the

luminosity values. Do I need to tonemap the RGB values as well in some way to prevent this undesirable effect?

I want to try running the program without my 500 line denoising program and just my 50 line tonemapping function to be sure it works better with the denoising. I am guessing that any noise in the version before tonemapping will be greatly exaggerated after tonemapping. Thus, the results would probably be much more noisy if I did not do my complicated denoising before my very simple tonemapping.

Future Improvements:

Major improvements could be made to my code by fixing rounding errors. I think there is a bug in my code in that the pixel values produced by the temporal filter part are more often below the original luminosity then they are above the original luminosity. Possibly this is a rounding error. There are other places where I may also be losing precision in my calculations. For example, I am rounding the float luminosity values between the temporal filter step and the spatial filter step. The downside of using the opencv bilateral filter is that I must pass it an integer array of pixel values rather than a float array of pixel values. The results from the temporal filter thus must be rounded to the closest integer before being passed to the bilateral filter.

Further, whereas in the paper their spatial filter is a bilateral median filter, mine is simply a bilateral filter. Thus, I am not removing shot noise in frames where I could get little to no temporal contribution. If the pixel is in a moving region and its center contains a lot of noise, the bilateral filter will get no spatial contribution either. Thus,

the noisy pixel will stay at its same noisy value without adding some sort of median filtering step before I run the bilateral filter.

[1] Eric P. Bennet, Leonard McMillan, Video Enhancement Using Per Pixel Virtual Exposures, ACM Transactions on Graphics (TOG), v. 24, n. 3, July, 2005