DSP Final Project Team 8
High ISO Noise Reduction

Members: 徐品傑, 莫子威

I. Abstract

When trying to take pictures in low light conditions, we often increase the ISO value to get better exposed pictures. However, by raising the ISO value we also amplify the noise in the image. And since the SNR is already low in such conditions, the noise effect greatly influences the resulting image. We wish to eliminate the noise amplified by high ISO during post processing of the image and still get well exposed pictures.

We will denoise the image using both spatial and temporal methods, by taking burst shots of the same scene. We will keep the camera as steady as possible, make sure objects are still between each frame, and keep our camera settings consistent throughout the shoot. This way we can be certain the different pixel values between frames are caused mainly by random noise, which we will try to eliminate.

II. Analysis

When taking the images, the values of each pixel are corrupted by noise, and since it is almost impossible to completely separate the original signal from the noise, we can only first guess the noise distribution and then calculate the signal from the assumed noise. We can express the value of each pixel with the following equation:

$$g(x,\ y) = f(x,\ y) + n(x,\ y)\,,\quad (1)$$

where $g(x,\ y)$ is the measured pixel value from the camera at the point (x, y), f is the real pixel value, and n is noise.

According to the images we took, the distribution is assumed to be gaussian distribution. For simplicity, we suppose that the mean of noise is zero. Under this assumption, we then need to find a way to merge and optimize all the burst shots into our final denoised image.

III. Proposed Method

Under the assumption that the noise is Gaussian distributed, we will apply LMMSE(Linear Minimum Mean Square Error) across burst shots to denoise the image. We chose LMMSE because according to previous works (e.g., [Zhang and Wu 2005]), to obtain the optimal estimate of the final image value $\widehat{g}$ from g and n, the optimal minimum mean square-error estimation (MMSE) of $\widehat{g}$ is

$$\widehat{g} = E[f|g] = \int f \cdot p(f|g)df, \quad (2)$$

However, the MMSE estimation is very difficult, if possible at all, since p(f|g) is seldom known. Instead, we use LMMSE since it is a good approximation especially when f and n are Gaussian processes. So we will use the following equation instead to approximate the MMSE of our images:

$$\widehat{g} = u + \frac{\sigma_c^2}{\sigma_c^2+\sigma^2}(g_t - u), \quad (3)$$

where u is the mean of all consistent pixels (same position) across all N frames (N is the number of input images). The variance σ_c^2 of true pixels is approximated by max(0, $\sigma_t^2 - \sigma^2$), where σ_t^2 is the variance of each consistent pixel across frames and σ^2 is the variance of the noise, and g_t is pixel color from the t-th frame, which will be our reference image to denoise.

We also found that by first denoising each frame spatially, then applying LMMSE we can obtain even better results, which we will show in our supplementary pdf. The filter we used is the anisotropic filter, which is implemented using the imdiffusefilt() function in MATLAB's library. This filter preserves the sharpness of edges better than Gaussian blurring. This is possible because a threshold function is used to prevent diffusion to happen across edges and therefore preserves edges in the image. However, some blurring will still occur as you can see in our supplement.

In order to reduce the sensitivity to reference frame selection, we will apply the LMMSE method to several frames (N_{ref} pieces) and merge them by weighted average. Every weight is estimated by the reliability of data and calculated for every pixel respectively. According to previous work (e.g., [Liu and Lu 2014]), we use the following equation to calculate every weight:

$$w(x,y) = \sqrt{\frac{m(x,y)}{N}},$$

where N is the total frame number, and m(x, y) is the number of inliers (g_t is an inlier when $|g_t - \widehat{g}| < \sigma_t$, where σ_t is the standard deviation of $\{g_t\}$) at the point (x, y).

Establishing the weighted matrix, we implement the weighted average in the following expression $N_{ref} - 1$ times:

$$\widehat{R_s} = R_s \otimes W + R_{s-1} \otimes (1 - W)$$

where R_s and R_{s-1} are different LMMSE result and use them to update the result R_s. After merging all N_{ref} reference frames, we obtain our final denoised image.

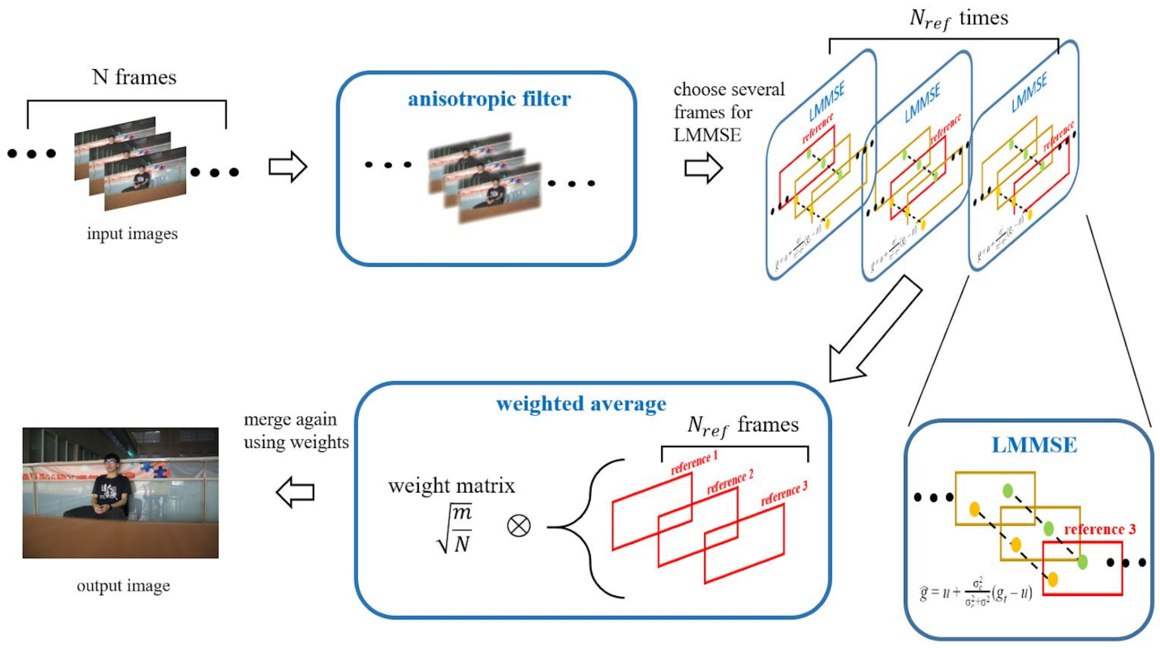
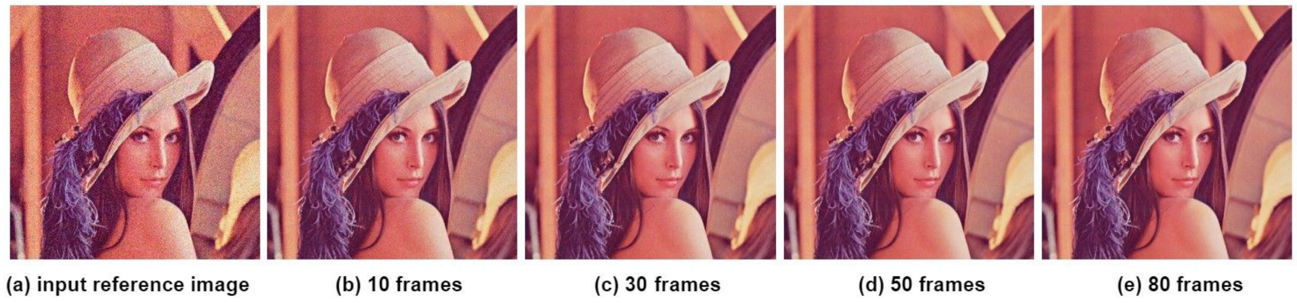
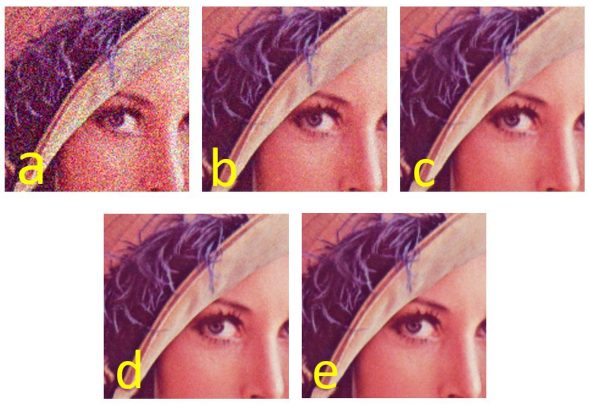


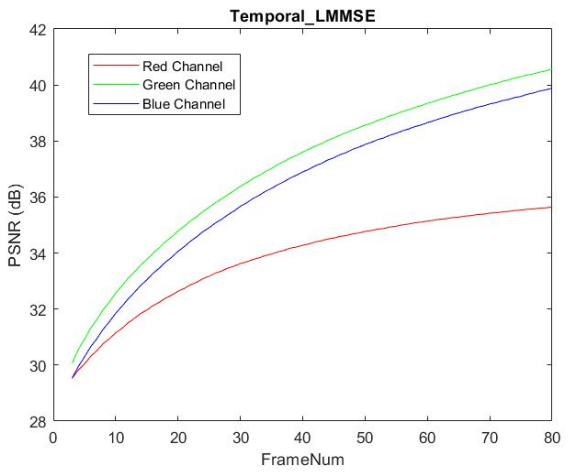
figure 1, *shows the complete process of our method*

IV. Analysis

A. First we added Gaussian noise with normalized variance 0.02 to Lena (512x512), here we try to see how the total frame numbers affects the PSNR after denoising.



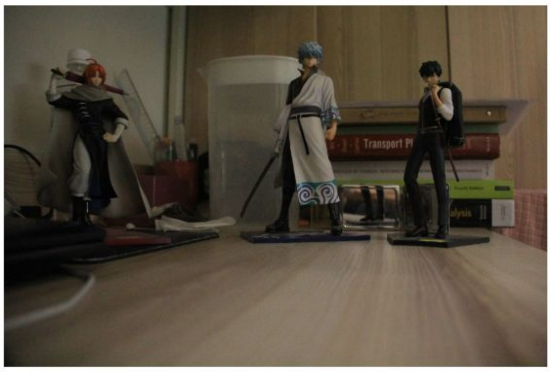


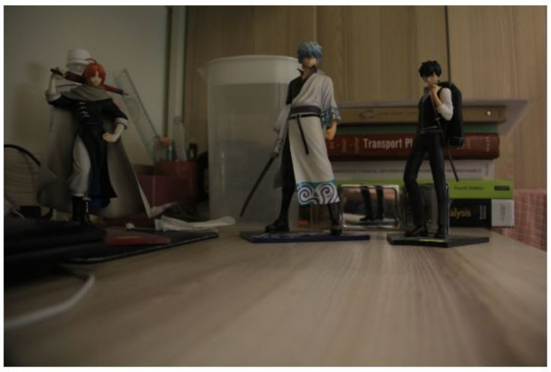


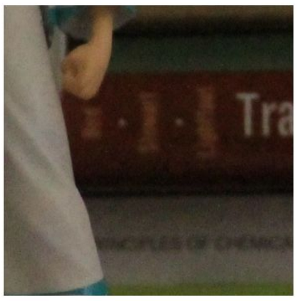
Here the PSNR is done by taking the original Lena image without Gaussian noise and the final denoised image. It is obvious as the total frames increase, PSNR also increase, however the value starts to saturate around 50 frames, and the denoising effect is barely noticeable when we directly view the images.

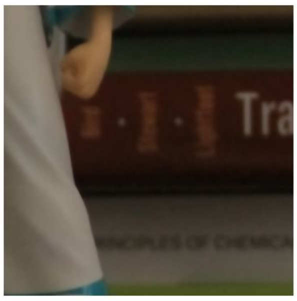
B. Now we apply our algorithm to pictures we have taken:

I. Total number of frames:12, ISO:6400

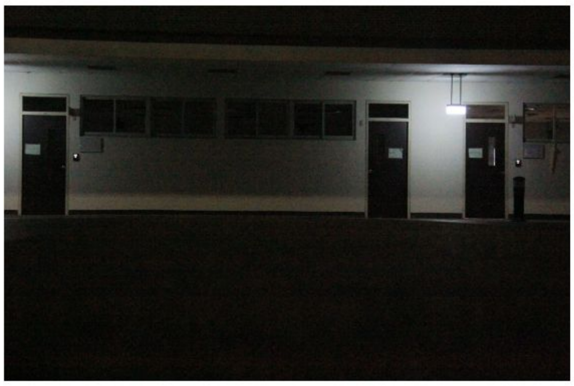

(a) input reference image

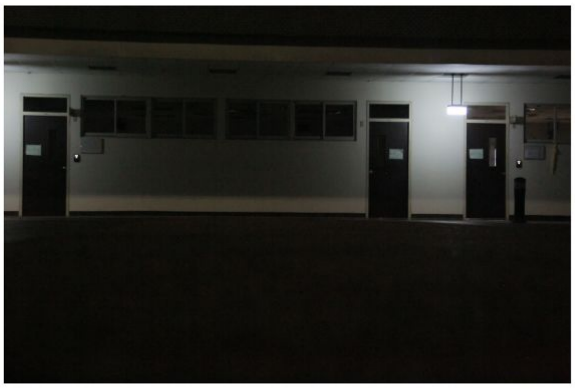

(b) image after denoise


(c) close up of (a)

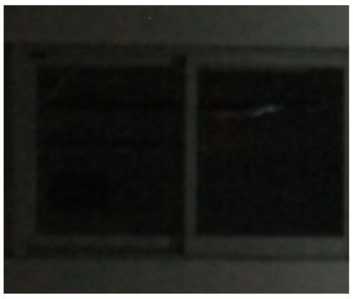

(d) close up of (b)

II. Total number of frames: 21, ISO:12800


(a) input reference image


(b) image after denoise

(c) close up of (a)


(d) close up of (b)

V. Conclusion

From the results above we can see that the images are much cleaner after our denoising method. With temporal processing, the final image retains its sharpness, which is different from only using spatial filters that filter the high frequency components (details) out. However, to increase PSNR, we also used anisotropic filters on the inputted images, since it improves the quality of each frame while retaining most of the details (comparisons between adding the diffusion filter and not adding the filter will be provided in the supplementary.pdf, where a tradeoff between detail and denoising can be seen).

VI. Contributions

A. Method Contributions

When we started researching this project, we found Lin Dada[1]'s work. But since we are taking burst shots, we should have enough information to denoise in a pixel-wise manner. Therefore, we assumed that the same pixels in different frames are a set of random variables. We then found LMMSE is an excellent way to approach the best result based on [2] and [3]. However, we thought that different references would vary the result, so we utilized the average weighting method in [1], but calculated the adaptive weight for every pixel using equations from [2] to avoid the poor result due to outliers in the input images. We also decided to add the diffusion filter in the beginning, even though it isn't part of the temporal process. We just thought filtering the images beforehand would yield better results, and indeed the PSNR values were higher, but some details were blurred, which was unideal.

B. Implementation Contributions

We wrote the code on Matlab. Except PSNR and imdiffusefilt() which are built in Matlab functions, we implemented all the referenced equations by ourselves and wrote our own code, including LMMSE and weighting average.

VII. Member Roles and Responsibilities

We both did equal amount of work. We took pictures, wrote all the codes, sorted the data, and wrote the report together.

VIII. Reference

[1] Lin Dada, "Temporal Noise Reduction (3DNR)" [Online]. Available: http://aboutdada.com/?p=1294 [Accessed Jun. 21, 2019]

[2] Lei Zhang and Xiaolin Wu, “Color Demosaicking Via Directional Linear Minimum Mean Square-Error Estimation” [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1532316 [Accessed Jun. 21, 2019]

[3] Ziwei Liu, “Fast Burst Images Denoising” [Online]. Available: https://liuziwei7.github.io/papers/burstdenoising.pdf?fbclid=IwAR25OwmF-6Un6OyawcE7vulNK-G3cd7LtkkuDcBckykZvsTos5qkcNR5IBk [Accessed Jun. 21, 2019]