Video noise reduction

Zhenghao Sun	Shu Yang
szh1007@bu.edu	conlany@bu.edu

1. Task

This project primarily investigates various denoising algorithms in the time domain, spatial domain, and transform domain for their efficacy in standard noise reduction environments, while also assessing the resilience and vulnerabilities of these denoising methods. Furthermore, If time permits, we plan to tailor algorithms specifically to mitigate noise in certain situations, such as denoising video capturing movement of liquid droplets.

2. Related work

Denoising is a fundamental step in image and video processing with an extensive array of methods developed over the years. These time-tested exemplary algorithms provide us with a foundation and inspiration for our research.

One of the simplest and earliest methods for image noise reduction is point-based noise filtering[1]. This technique involves processing each pixel individually, typically applying a straightforward operation like thresholding or averaging. While this method can be effective for certain types of uniform noise, it often lacks the sophistication needed for more complex noise patterns or for preserving detailed features in the image.

However, frame-by-frame low-pass filtering for denoising tends to blur the video spatially. Therefore, performing temporal low-pass filtering rather than spatial[2] will be helpful to solve this problem. One option is to develop a video noise reduction algorithm based on luminance/color filtering along estimated motion trajectories[3].

Another option is to leverage the concept of nonlocal noise reduction[4], which revolves around the idea that an image has repetitive patterns and similarities. Leveraging these similarities can help in efficiently suppressing noise without affecting the key features of the image. The non-local means approach, in particular, uses bilateral filters[5] to consider both spatial proximity and intensity similarity when smoothing an image, thus preserving edges and details.

3. Approach

3.1 2-D non-local means

A common algorithm for noise reduction involves averaging the values of the neighboring pixels for the value of the central pixel. To minimize the blurring effect to the greatest extent, traditional methods use a weighted average, where the weights are determined by the distance or the difference in grayscale values.

$$u(i) = \sum_{j \in S_i} w(i, j) g(j)$$

Using the aforementioned method, the influence of noise on the grayscale values will inevitably affect the calculation of weights. Therefore, as opposed to traditional algorithms that base the weight on the grayscale value of a single point (which is easily disturbed by noise), the weight for a point is determined by the overall similarity of that point and its neighborhood. This approach helps to mitigate the impact of noise on the weight determination by considering a larger context around each pixel.

$$w(i, j) = \frac{1}{Z(j)} \exp(- \frac{\| N(p_i) - N(p_j) \|_{2,a}^{2}}{h})$$

Where

$$Z(i) = \sum_{j \in S_i} \exp(- \frac{\| N(p_i) - N(p_j) \|_{2,a}^{2}}{h})$$

N(*pj*) represents the pixel grayscale values within an image block of size *M*×*M* centered at *j*, and h controls the decay rate of the exponential function.

3.2 PCA-NLM

Similarly, due to the presence of noise within a neighborhood, although the accuracy of weight calculation based on the neighborhood has improved a lot compared to a single point, it is still affected by noise. To further eliminate the influence of noise, since noise information mainly exists in the space composed of non-principal component eigenvectors, using PCA (Principal Component Analysis) to calculate and retain its principal components can preserve the signal to the greatest extent and remove the influence of noise. Operating PCA calculation we have:

$$\boldsymbol{X}_{\text{obs}} = \boldsymbol{U \Lambda U}^{\mathrm{T}}$$

Where:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

By retaining the eigenvectors corresponding to the first m principal components, the weights are determined by calculating the similarity of the projections of other points and their neighborhoods in the search window onto the space formed by these eigenvectors, compared to the projection of the central point.

$$w(i,j) = \frac{1}{Z_{\mathrm{PCA}}(j)} \exp\left(-\frac{\| \mathbf{N}_{\mathrm{PCA}}(p_i) - \mathbf{N}_{\mathrm{PCA}}(p_j) \|^2}{h}\right)$$

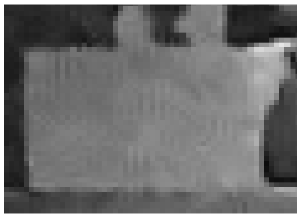
Where:

$$\mathbf{N}_{PCA}(p_i) = \boldsymbol{U}_p \bullet \mathbf{N}_p(p_i)$$

$$\boldsymbol{U}_p = [u_1, u_2, \cdots, u_m]$$

$$Z_{\mathrm{PCA}}(i) = \sum_{j \in Si} \exp\left(-\frac{\| \mathbf{N}_{\mathrm{PCA}}(p_i) - \mathbf{N}_{\mathrm{PCA}}(p_j) \|^2}{h}\right)$$

Through testing, it has been found that the PCA-NLM performs well. However, different parameter selections can have certain impacts on the effectiveness. A 5x5 neighborhood has better results when processing areas with simple textures, while a 3x3 neighborhood performs better in areas with more detail. However, there are better choices than using a fixed neighborhood size for the entire image. Larger neighborhoods can make mistakes in processing detailed parts, while smaller neighborhoods can produce speckles in relatively smooth areas, as shown in the following figure:



Therefore, this study adds an analysis of texture complexity to the PCA-NLM framework and classifies the complexity of the texture of the area to which each point belongs into three levels. Correspondingly, different neighborhood sizes are used, and different parameters h are set for each level.

To decide the complexity of the structure within the neighborhood, the following methods are applied. First, the local structure tensor *J* is computed for each point within a neighborhood of size *k*×*k* in the image. Then, by using Singular Value Decomposition (SVD), we can obtain the magnitude of the gradient values in the two principal directions. If the gradient values in both principal directions are large, it indicates a situation of a complex corner structure. If one principal direction has a large gradient value and the other is small, it indicates the presence of an edge. If both principal directions have relatively small values, it indicates a relatively smooth region with the lowest structural complexity.

$$\boldsymbol{J} = \boldsymbol{G}\boldsymbol{G}^{\mathrm{T}} = \begin{bmatrix} \sum_{i \in \mathbf{N}} g_{ai}^2 & \sum_{i \in \mathbf{N}} g_{ai} g_{bi} \\ \sum_{i \in \mathbf{N}} g_{ai} g_{bi} & \sum_{i \in \mathbf{N}} g_{bi}^2 \end{bmatrix}$$

Where:

$g = \nabla f(a, b)$

By operating the SVD, we have:

$$\boldsymbol{J} = \boldsymbol{W} \begin{bmatrix} \lambda_1' & 0 \\ 0 & \lambda_2' \end{bmatrix} [R_1 \quad R_2]$$

We define the following edge texture feature descriptor function:

$$F(x) = \frac{(\lambda_1' - \lambda_2')^n}{1 + B(\lambda_1' + \lambda_2')} \times \lambda_1'$$

Based on the values of *F*(*x*), by selecting different neighborhood sizes and different values of the parameter *h*, we obtain the following results. The optimized PCA-NLM, referred to as textured PCA-NLM, which utilizes the aforementioned method, shows that the speckle issue caused by too small a neighborhood selection in PCA-NLM has been resolved in smooth areas.

3.4 3-D non-local means

In addition to the two spatial coordinates, a temporal coordinate is also considered. The core mathematical formula of the 3D Non-local means denoising algorithm is based on the weighted average of each pixel or small patch in the image. The weights are calculated based on the similarity of the patch to its neighboring patches within a search window. The mathematical expression is as follows:

Suppose *V(i, j, k)* is the pixel value at location*(i, j, k)* in the 3D image(such as a video sequence), where i, j are spatial coordinates and k is the temporal coordinates. For a given pixel *V(i, j, k)*, its denoised value *DN(i, j, k)* can be calculated using the following formula:

$$DN(i,j,k) = \frac{\sum_{p,q,r\in N(i,j,k)} w(i,j,k,p,q,r)\cdot V(p,q,r)}{\sum_{p,q,r\in N(i,j,k)} w(i,j,k,p,q,r)}$$

Here, *N(i, j, k)* represents a three-dimensional search window centered at *(i, j, k)*, and *w(i, j, k, p, q, r)* is the weight, representing the similarity between *(i, j, k)* and *(p, q, r)*. The weight is usually calculated based on the Euclidean distance between two patches, as shown below:

$$w(i,j,k,p,q,r) = \exp\left(-\frac{\|P(i,j,k)-P(p,q,r)\|^2}{h^2}\right)$$

In this context, *P(i, j, k)* and *P(p, q, r)* are small patches centered at *(i, j, k)* and *(p, q, r)*. Respectively, *h* is a smoothing parameter that controls decay, and ||*P(i, j, k) - P(p, q, r)*|| is the Euclidean distance between two patches.

Through this method, the algorithm effectively utilizes the redundancy of the image in space while considering the temporal dimension, thus achieving the purpose of denoising.

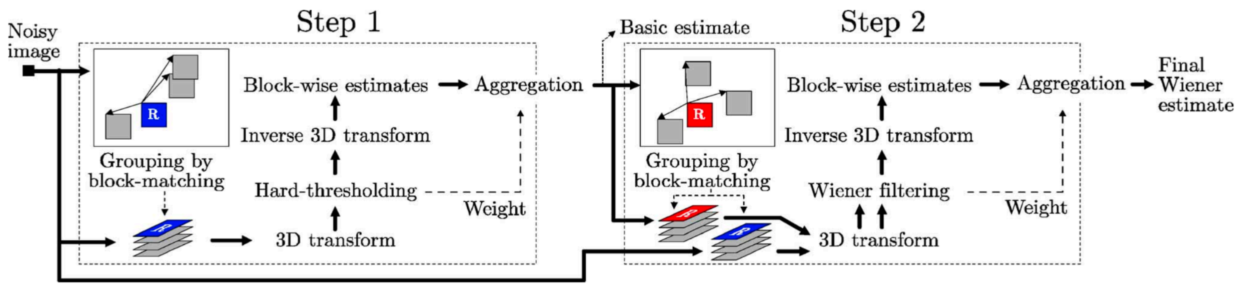
3.5 3-D motion-compensated Kalman filtering

Because Kalman filtering is fundamentally different from mean filtering, Kalman filtering does not use the weighted average of neighboring values for filtering but instead denoises by combining predictive values and observed values. Therefore, Kalman filtering can be used for preprocessing to reduce noise to a certain extent, and then further denoising can be carried out by inputting into a mean filter. Kalman filtering mainly consists of two phases: prediction and update. In this study, block matching is used as the prediction module. First, it is assumed that two adjacent frames have the same luminance, but the objects in them are spatially unique. This assumption makes motion compensation as a prediction module reasonable. It−1 and It is used as inputs to the module, resulting in the matched It~. Then combines with the noisy image(observation) ,we have:

$$\hat{X}_k = K_{.k}.Z_k + (1 - K_k).\hat{X}_{k-1}$$

Where is the current estimation of the signal value, is the previous estimation of signal value, is the measurement value and is the Kalman gain.

3.6 Block-Matching and 3D Filtering[6]



3.6.1

The principle of the algorithm is to form a 3-D group of blocks by identifying blocks with similar content. Since each block has comparable content, performing a Fourier transform on this group of blocks results in a cumulative effect of the relevant information. Consequently, meaningful information can be extracted by filtering out the portions with smaller magnitudes (which typically represent noise).

3.7 Fast Non-local means filtering

After testing, it has been found that filtering a 256*256 size image using the traditional Non-Local Means (NLM) algorithm takes about one minute. The main time consumption lies in the calculation of the similarity between the current patch (centered around the current pixel) and all target patches (centered around every other point in the search window). The similarity is measured using the Euclidean norm (L2 norm).

$$\|N(i)-N(j)\|_{2,\alpha}^2 = \|N(i)\|_{2,\alpha}^2 + \|N(j)\|_{2,\alpha}^2 - 2 \times N(i) \otimes N(j)$$

Where $\|N(i)\|_{2,\alpha}^2$ and $\|N(j)\|_{2,\alpha}^2$ are the the weighted L2 norm of N(i) and N(j) and $2 \times N(i) \otimes N(j)$ is the weighted inner product of N(i) and N(j), which can be calculated using convolution. The inner product have the following characteristic:

$$\mathcal{F}\{h \star I\} = \overline{\mathcal{F}(h)} \cdot \mathcal{F}\{I\}$$

$$\{h \star I\} = \mathcal{F}^{-1}\{\overline{\mathcal{F}(h)} \cdot \mathcal{F}\{I\}\}$$

Wherein, F and F-1 represent the Fourier transform and inverse Fourier transform, respectively. The symbol ' * ' denotes cross-correlation, the symbol '.' denotes pointwise multiplication, and '—' represents the complex conjugate.Thus we have, for each

$$N(i) \otimes N(j)\,,\ i \in \Omega\,,\ j \in \Omega_i$$

We have:

$$\mathcal{F}^{-1}\{\overline{\mathcal{F}\{G \cdot N(i)\}} \cdot \mathcal{F}\{D(i)\}\}$$

Wherein G is the Gaussian weighting function, and $D(i)$ represents the image in the search window.

3.7 Adaptive Non-local means filtering

In the NLM algorithm, the filter parameter $\hbar$ is typically chosen to be the standard deviation of the noise, that is $\hbar$=σ. As shown in Figure, the smaller the value of the filter parameter $\hbar$, the greater the influence of the weighted Euclidean distance d(i,j) between image blocks on the weight w(i,j); conversely, the larger the value of the filter parameter, the lesser the influence of the weighted Euclidean distance between image blocks. Therefore, when the image contains strong noise, that is, when σ is large, the weights tend to be uniform, and the denoising effect of NLM tends towards the effect of mean filtering, resulting in the loss of image structure and detail information, making the image blurry.

To address the above issue, we attempt to learn from the idea of deep neural networks, using multi-layer networks to gradually improve the denoising effect of the generated images, instead of using a single filter with a large $\hbar$ parameter to generate the images directly. Suppose filtering is performed k times, the filtering parameters for the K times are $\hbar$1(i),$\hbar$2(i),...$\hbar$k(i), where hk(i)<σ, for k=1,2,3,...K. In this experiment, we choose K=2, and the filtering parameter for the first filtering is chosen to be $\hbar$=0.5σ.

$$D\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n} a_i^2 D(X_i) + 2 \sum_{1\leq i} \sum_{<j\leq n} a_i a_j \mathrm{Cov}(X_i, X_j)$$

Where D stands for variance and ai, i=1,2,3,...,k are the weights, Cov(Xi,Xj) stands for the covariance . This equation describes the variance of each point in an image with known noise variance, after undergoing a weighted average. In our case, since the noise at each point is uncorrelated with each other, it follows that Cov(xi,xj)=0,when i do not equal j.Thus we have

$$D\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n} a_i^2 D(X_i)$$

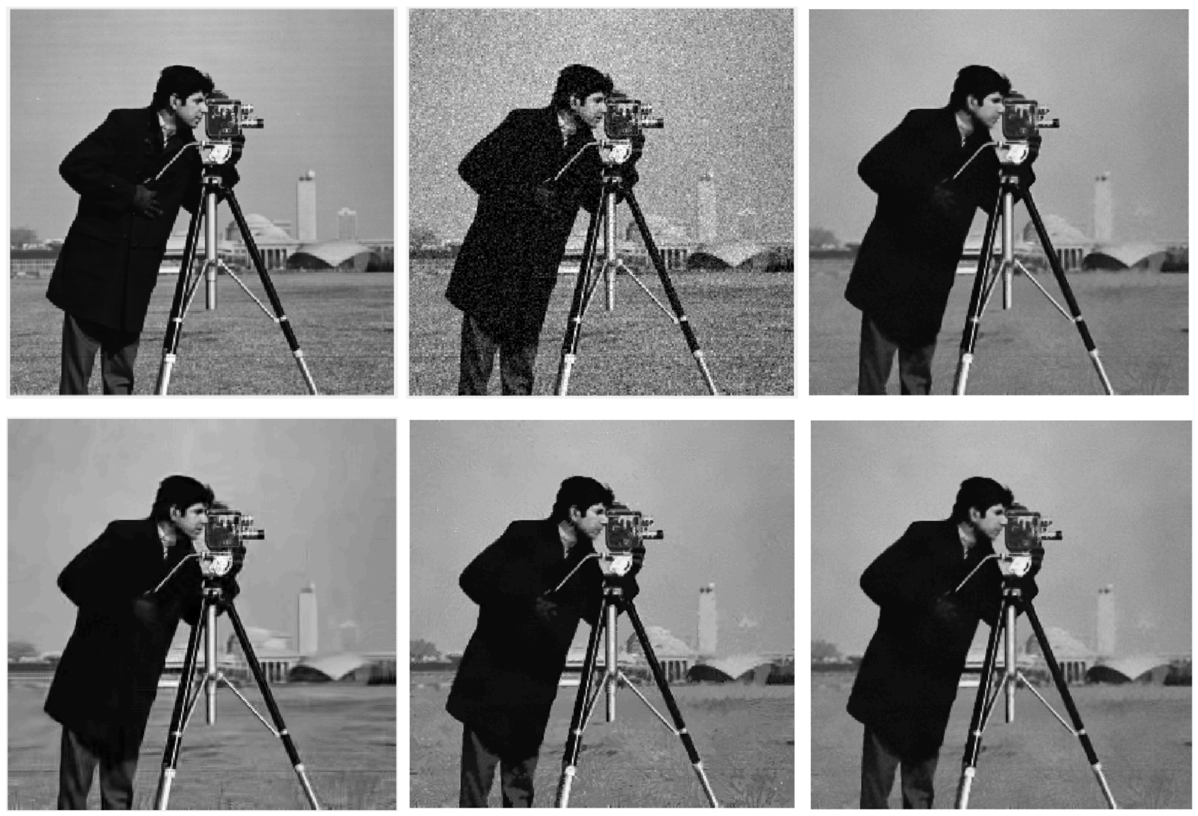
Since the noise is independently and identically distributed, according to the formula above, after the first filtering, the standard deviation of the noise at each pixel point at position i is

$$\sigma_2(i) = \sqrt{\sum_{j\in\Omega_i} w^2(i,j)\sigma^2}, \quad i \in \Omega$$

Thus we set

$$h_2(i) = \sigma_2(i)$$

4. Results



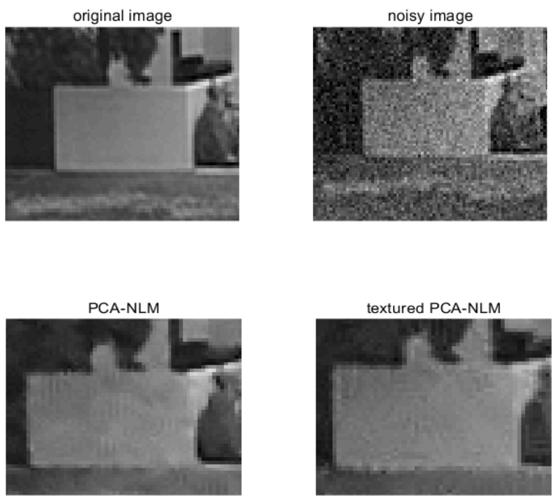
Noise σ	Noisy image	PCA-NLM	Fast-NLM	ANLM	BM3D
10	28.10	32.40	32.56	31.05	34.13
20	22.08	28.47	28.65	29.28	30.46
30	18.56	26.16	26.39	27.32	28.57
40	16.06	24.72	24.53	26.30	27.07
50	14.12	23.76	23.62	23.94	26.08

Noise σ	Noisy image	PCA-NLM	Fast-NLM	ANLM	BM3D
10	0.6304	0.9115	0.9037	0.9121	0.9300
20	0.3961	0.8796	0.8591	0.8516	0.8722
30	0.2839	0.7359	0.7269	0.7421	0.8312
40	0.2178	0.6912	0.6891	0.7436	0.7990
50	0.1738	0.6425	0.6317	0.7104	0.7737

Noise σ	PCA-NLM	Fast-NLM	ANLM	BM3D
10	99.31	2.822	5.524	74.807
20	97.41	2.815	4.475	75.020
30	71.06	2.522	4.991	76.119
40	43.25	1.081	3.629	76.471
50	57.63	1.062	2.253	77.112

Noise σ	Noisy	PCA-NLM	KLM	ANLM	V-BM3D
10	28.13 / 0.7233	32.43 / 0.9115	31.95 / 0.9567	32.94 / 0.9147	36.25 / 0.9638

30	18.59 / 0.3299	27.29 / 0.8128	26.04 / 0.7415	27.86 / 0.7917	30.85 / 0.8895
50	14.15 / 0.1826	22.62 / 0.7418	24.85 / 0.6412	24.03 / 0.6499	27.65 /0.7986



4. Solution selections

4.1 target solution

Implement all 5 algorithms to produce videos with effective noise reduction. Additionally, test across various application scenarios, introducing Gaussian noise, salt-and-pepper noise, and speckle noises. Specific application scenarios include noise reduction for videos with small objects, videos with low luminance values, and videos capturing fast-moving objects.

5. Approximate Timeline

Task	Timeline
Submit Project Proposal	10/19/2023
Read related paper	
Preprocess dataset	
Implement algorithms	

Submit intermediate report	11/09/2023
Refine algorithms	
Compare experiments' results	
Generalize codes and results	
Write final report	
Prepare presentation	
Presentation	12/07/2023
Submit the final code, report, and slide	12/12/2023

6. Division of labor

Shu Yang:

- Implement algorithms including:
  - 2-D low-pass filtering (Wiener),
  - 2-D non-local means,
  - 3-D motion-compensated low-pass filtering.
- Video generating
- Report writing

Zhenghao Sun:

- Paper reading
- Dataset preprocess
- Implement algorithms including:
  - 3-D non-local means,
  - Block-Matching and 3D Filtering
- Experiment with various scenarios and types of noise.

7. Reference

[1] A Review of Image Denoising Algorithms, with a New One - Antoni Buades
[2]Combined Spatial and Temporal Domain Wavelet Shrinkage Algorithm for Video Denoising - EricJ.Balster,Member,IEEE,YuanF.Zheng, Fellow, IEEE,andRobertL.Ewing, SeniorMember,IEEE
[3]Efficient video denoising based on dynamic nonlocal means - Yubing Han, Rushan Chen
[4]Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm - Marc Lebrun, Antoni Buades, Jean-Michel Morel
[5]Multiresolution Bilateral Filtering for Image Denoising - Ming Zhang and Bahadir K. Gunturk
[6]Image and video denoising by sparse 3D transform-domain collaborative filtering - Ymir Mäkinen, Lucio Azzari, Enrique Sánchez-Monge