The Report of Digital Video Processing Lab Assignment 4

IIR & FIR filters for video signals

Xiaowen Ke (27817312)

03. 25. 2017

Section 1 : Question 1

Spatial video noise filters: Write a Matlab program to perform *spatial Gaussian filtering* of each image $\mathrm{I(t)}, 1 \leq \mathrm{t} \leq \mathrm{N}$. In a video sequence I. N is the number of frames in I.

1. Download one of the video sequences provided on the course website. Read, e.g., use imread(), frames of the video sequence into your Matlab program.
2. Add Gaussian noise with power $\sigma_\eta^2 = 0.05$ to each input (noise-free) image I(t) to create a noisy image $I_\eta(t)$. To add such noise, you may use Matlab's imnoise().
3. Visualize the noise by viewing the noisy image $I_\eta(t)$, e.g., by using imshow().
4. Filter or reduce the added noise from $I_\eta(t)$ using a spatial Gaussian filter as follows:
   - Create a Gaussian filter window G of size W=3 and standard deviation $\sigma_g = 0.5$. You may want to revisit fspecial() here.
   - Using G, filter $I_\eta(t)$ to get the noise-reduced image $\hat{I}(t)$. Also, measure the computation time.
   - Repeat step 4 for a Gaussian filter with window size W=9.
5. Repeat step 2-4 for $\sigma_\eta^2 = 0.005$.
6. Calculate the PSNR between the input noise-free I(t) and noise-reduced $\hat{I}(t)$ and plot the noise-reduction gain in PSNR (or PSNR gain) for all the frames of I for both noise powers. The following Matlab function calculates the PSNR between images I and J.
7. Comment on your results using the following criteria:
   1. Subjective evaluation (visual comparison of noise reduced, noisy, and original images).
      - How many frames to select for visual comparison?
        Select a set of images that you think are most representative for the video sequence of your choice. For example, select one frame from

start, one from middle, and one from the end of the sequence. But this depends how much content change is in the video sequence.

2. Objective evaluation based on the PSNR, and
3. Computation time of the filter with different parameter values (e.g., $\sigma_\eta^2 =$ 0.005 versus 0.005).

1.1 Motivation

This question will help us to know how to load a video by Matlab. Then use the fuctions of toolbox to add some noise and remove them. Besides, we should know how to evaluate the denoising effect by calculating the PSNR.

1.2 Objective

We need to separate a video into frames then add Gauassian noise. Besides, we change the value of noise and window size of the filter. For each case, we calculate the PSNR and the noise-reduction gain.

1.3 Theory

The definition of *psnr* function:

```
function psnr = PSNR(I,J)
mse=(double(I)-double(J)).^2 ;
psnr=10*log(255^2/mean(mean(mse)));
```

1.4 Method

Firstly, I used the function *VideoReader* and *read* to load an video. Then I added the noise using *imnoise* and do the denoising using *fspecial* and *imfilter*. Finally, I used *psnr* to calculate the PSNR.

The resulting Matlab code I developed is as follows:

```
% ELEC6631, Digital Video Processing
% Matlab code for Lab assignment 4 question 1
% Xiaowen Ke on 25.03.2017
close all;
clear all;
clc;

tic
% Load the video
input_video='stefan_cif.avi';
video=VideoReader(input_video);
I=read(video);
```

```
N_frames=size(I,4);
% Add the noise
for t=1:N_frames

I_noised(:,:,:,t)=imnoise(I(:,:,:,t),'gaussian',0,0.05);
    subplot(1,2,1),imshow(I_noised(:,:,:,t));
    title('Noisy video');
    drawnow
end
% Denoising
for t=1:N_frames
    G=fspecial('gaussian',9,0.5);
    I_denoised(:,:,:,t)=imfilter(I_noised(:,:,:,t),G);
    subplot(1,2,2),imshow(I_denoised(:,:,:,t));
    title('Denoised video');
    drawnow
end
% Calculate the psnr and psnr gain
for t=1:N_frames
    PSNR(t)=psnr(I(:,:,:,t),I_denoised(:,:,:,t));
    PSNR1(t)=psnr(I(:,:,:,t),I_noised(:,:,:,t));
    PSNR_gain(t)=PSNR(t)-PSNR1(t);
end

figure(2)
plot(PSNR,'green');
hold on
plot(PSNR_gain,'blue');
legend('PSNR','PSNR gain');
title('PSNR and PSNR gain');
xlabel('Frame numner'),ylabel('PSNR gain');
toc
```

1.5 Results and discussion

1.5.1 For $\sigma_\eta^2 = 0.05$ and W = 3:

Figure 1 shows the denoising effect when the Gaussian noise power $\sigma_\eta^2 = 0.05$ and the Gaussian filter window size W = 3. As we can see from the figure, after adding the Gaussian noise to the video, the video was getting more sharp. From the denoised video on the right, we can see the noise degree is reduced (smoother), especially on the flat area (the green ground).

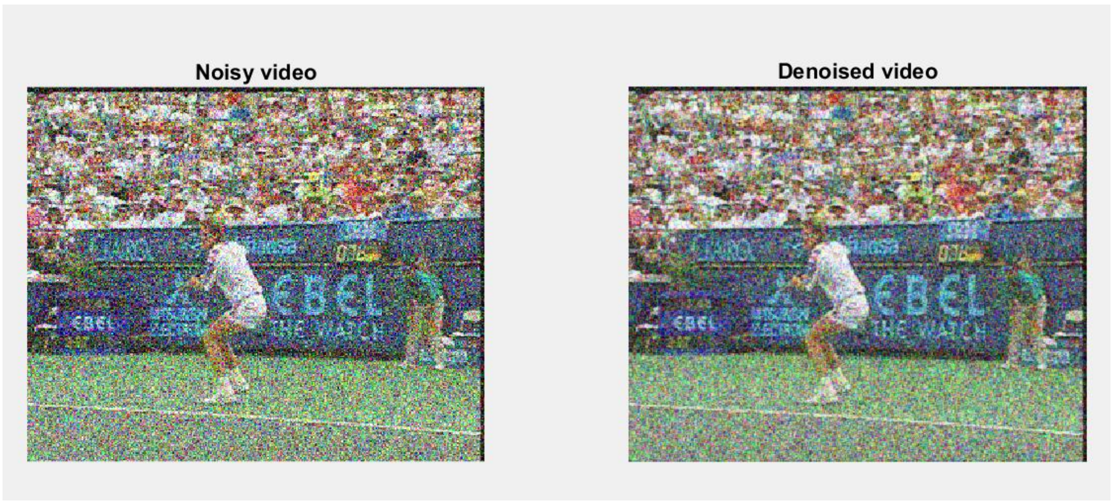


Figure 1 Denoising effect ($\sigma_\eta^2 = 0.05$ and W = 3).

Figure 2 shows the PSNR of the denoised video with the respect to the original video and the PSNR gain from the first frame to the 91 frame. As we can see from the figure, The PSNR is stay around 17dB, which is not high. From the figure, we can see the curves are stable from the first frame to the last frame, which implies that the denoising ability of spatial Gaussian filter is stable and frame-independent. The PSNR gain reflects the denoising effect of this filter. It reduced over 50% of the noise.

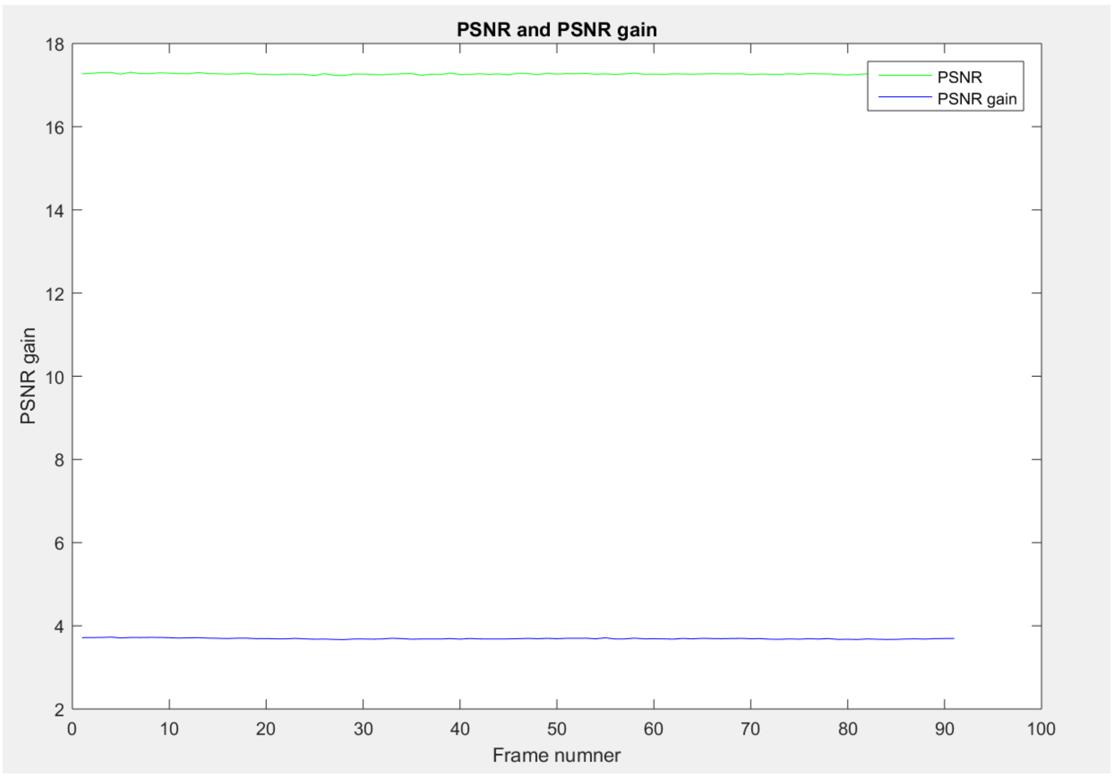


Figure 2 PSNR and PSNR gain ($\sigma_\eta^2 = 0.05$ and W = 3).

Computation time: 35.181879 seconds.

1.5.2 For $\sigma_\eta^2 = 0.05$ and W = 9:

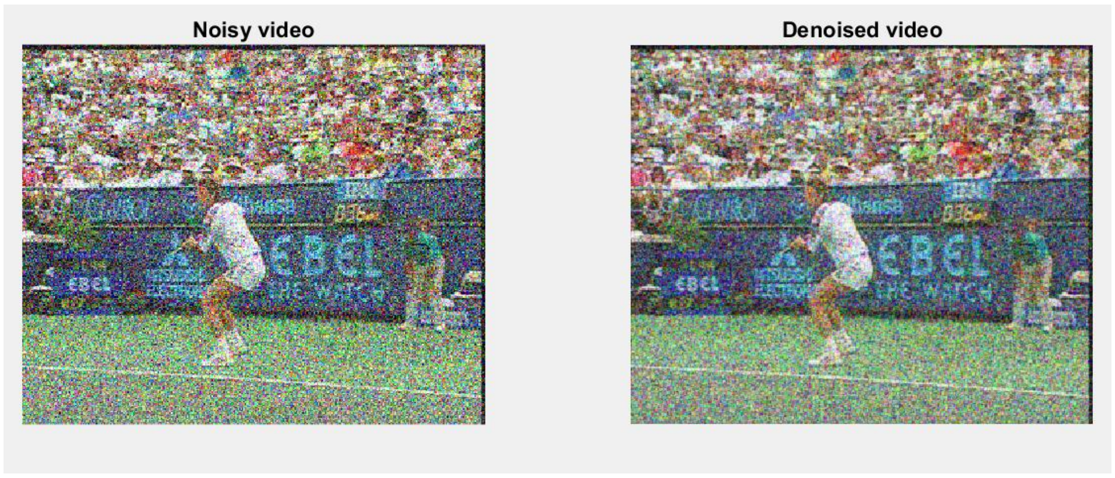


Figure 3 Denoising effect ($\sigma_\eta^2 = 0.05$ and W = 9).

Figure 3 shows the denoising effect when the Gaussian noise power $\sigma_\eta^2 = 0.05$ and the Gaussian filter window size W = 9. From the figure 3 and figure 4, we can conclude that there is no obvious improvement of the denoising effect when the window size is getting larger.

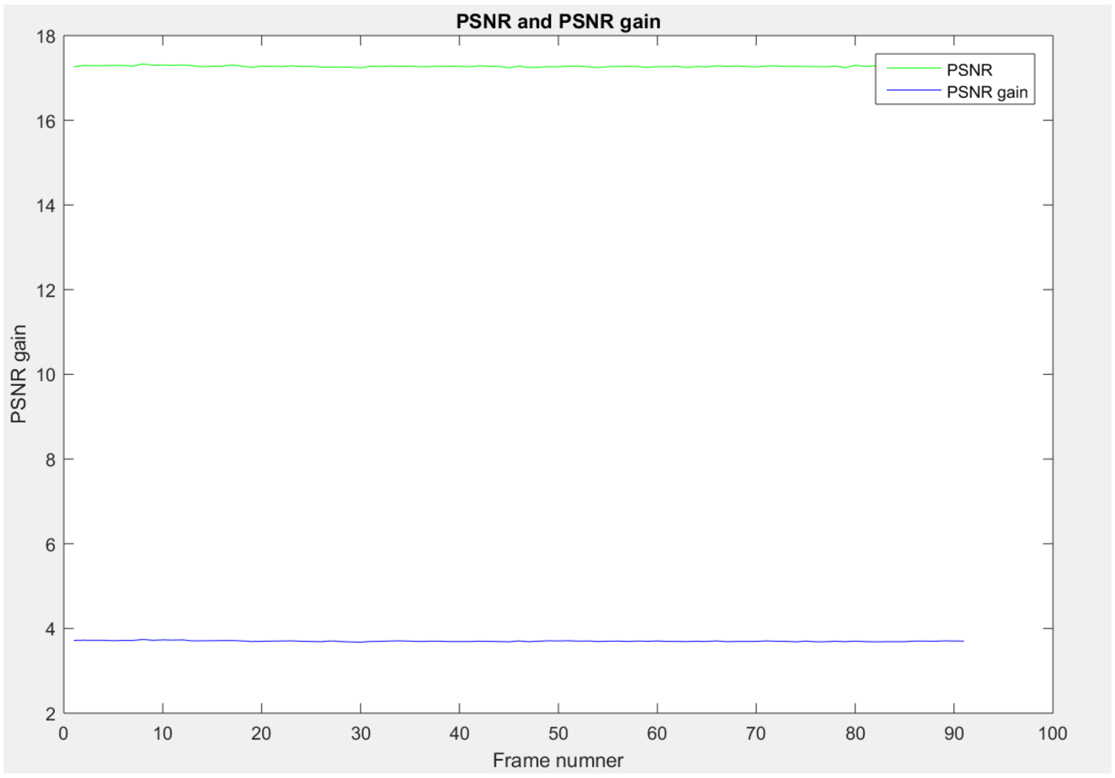


Figure 4 PSNR and PSNR gain ($\sigma_\eta^2 = 0.05$ and W = 9).

Computation time: 36.533856 seconds.

1.5.3 For $\sigma_\eta^2 = 0.005$ and W = 3:

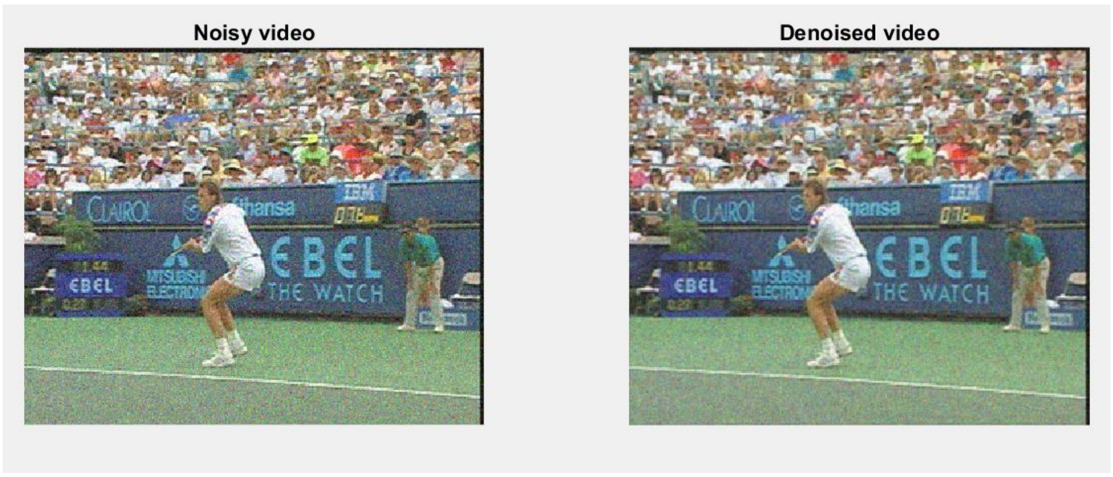


Figure 5 Denoising effect ($\sigma_\eta^2 = 0.005$ and W = 3).

Figure 5 shows the denoising effect when the Gaussian noise power is changed to be smaller $\sigma_\eta^2 = 0.005$ and the Gaussian filter window size W = 3. As we can see from the figure, the noise level is relative low. From the denoised video on the right, we can see less noise.

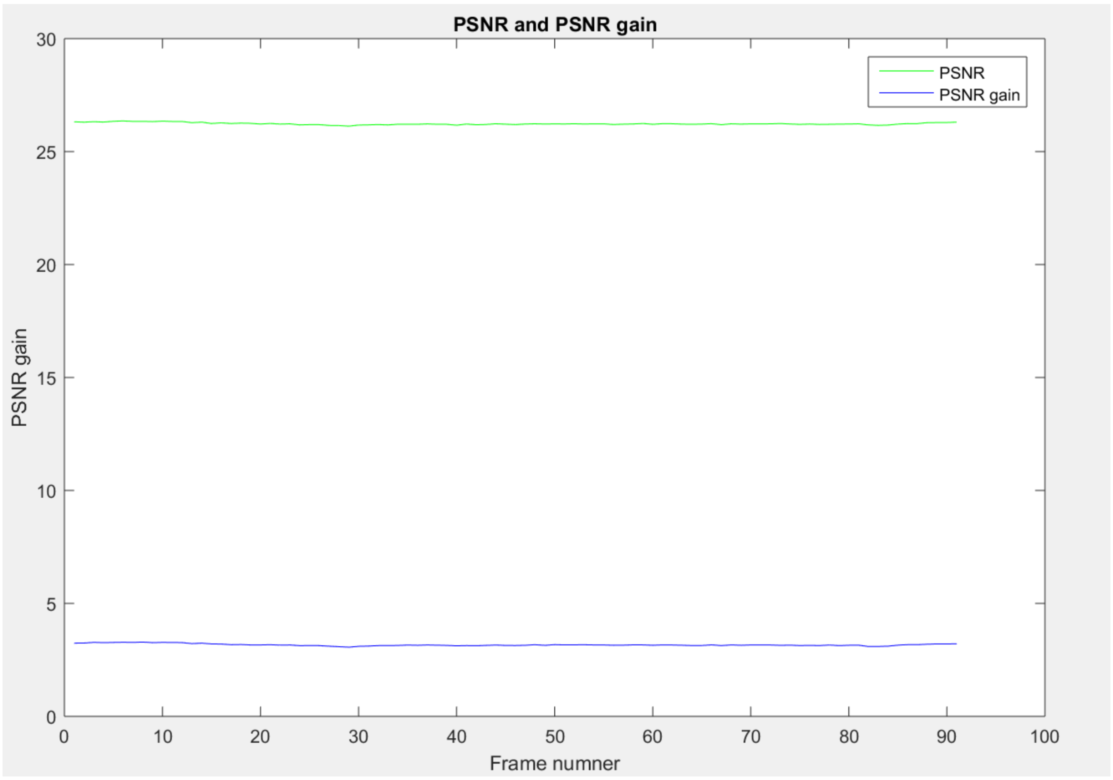


Figure 6 PSNR and PSNR gain ($\sigma_\eta^2 = 0.005$ and W = 3).

From the figure 6, we can see that the value of PSNR is much higher than it when $\sigma_\eta^2 =$

0.05. But the PSNR gain is almost the same as the case of $\sigma_\eta^2 = 0.05$. It implies the same denoising ability of the Gaussian filter.
Computation time: 35.337653 seconds.

1.5.4 For $\boldsymbol{\sigma_\eta^2 = 0.005}$ and W = 9:

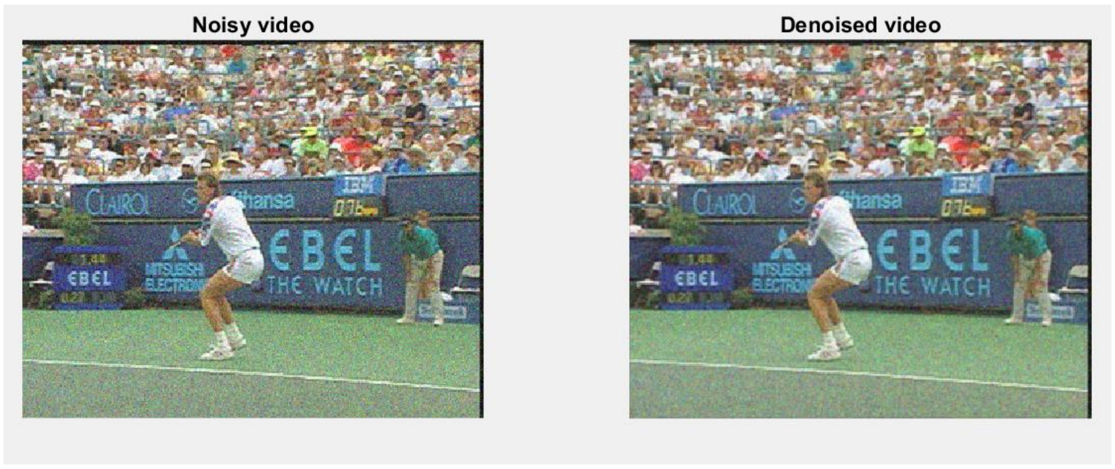


Figure 7 Denoising effect ($\sigma_\eta^2 = 0.005$ and W = 9).

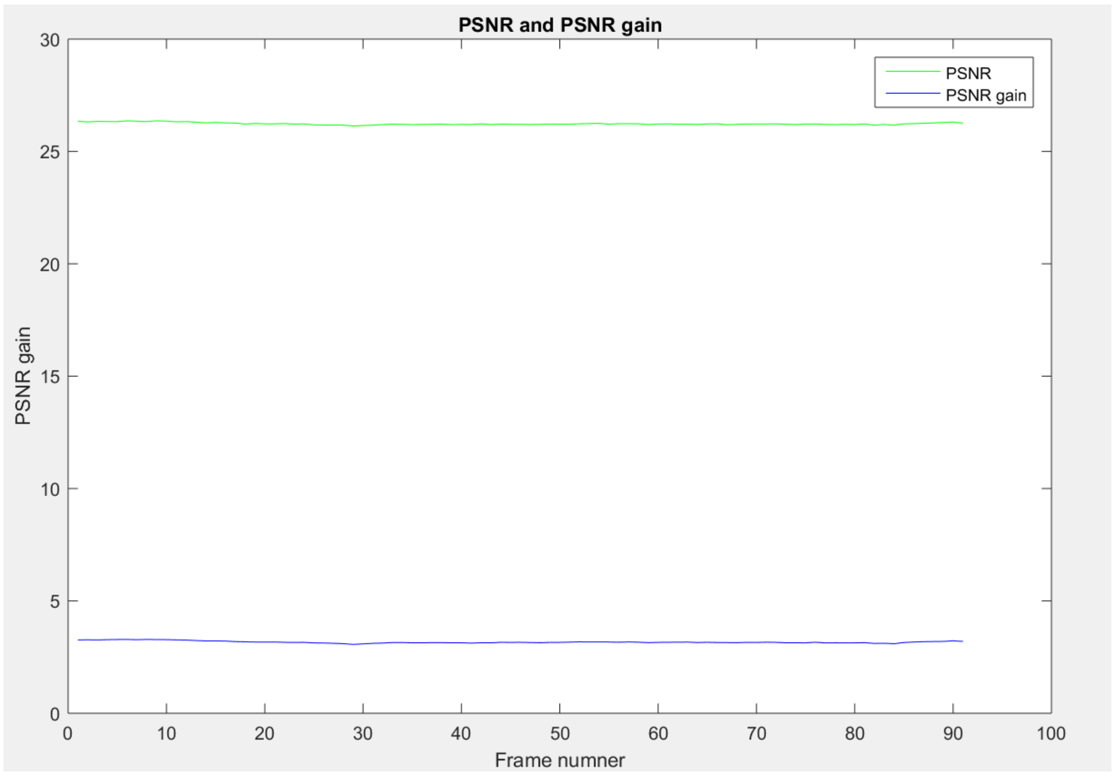


Figure 8 PSNR and PSNR gain ($\sigma_\eta^2 = 0.005$ and W = 9).

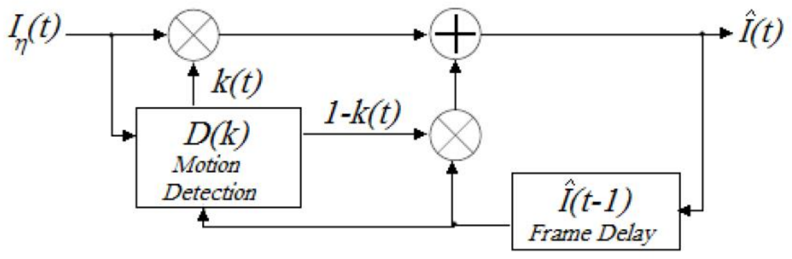
Figure 7 and figure 8 shows that there is no obvious improvement of the denoising effect when the window size is getting larger.
Computation time: 36.311780 seconds.

1.6 Conclusion

From the above results, we can conclude that the spatial Gaussian filter is not considering the temporal correlation between frames. The denoising effect stays almost the same when the size of filter window changed.

Section 2 : Question 2

Temporal video noise filters: The following figure shows the basic concept of a first-order temporal recursive video noise filter:



Let $I_\eta(t)$ be the noisy image of a video signal at time t and $I_\eta(x,y,t)$ be the pixel intensity in $I_\eta(t)$ at the spatial location (x,y). The filter output is

$$\widehat{I(t)} = k(t)I_\eta(t) + \big(1 - k(t)\big)I(\widehat{t-1}).$$

Where $I(\widehat{t-1})$ is the noise-reduced image at time t-1 (delayed by 1 memory unit, i.e., 1 frame). $k(t)$ is determined by the amount of motion for each pixel between t and t-1. $k(t)$ is called the coefficient (or control) frame and each of its elements takes a value between 0 and 1. Hence, the pixel-form of (1) is

$$\hat{I}(x,y,t) = k(x,y,t)I_\eta(x,y,t) + (1 - k(x,y,t))\hat{I}(x,y,t-1)$$

One way to calculate $k(x,y,t)$ is using

$$k(x,y,t) = 0.1 + 0.9\frac{D(x,y,t)}{d_{max}}$$

Where $D(x,y,t)$ is the amount of motion (i.e., difference) detected between $I_\eta(x,y,t)$ and $\hat{I}(x,y,t-1)$ and d_{max} is the maximum motion difference expected.

Note that the amount of detected D (linearly) increases or decreases the value of parameter k.
We use $31 \le d_{max} \le 255$ to normalize $D(x,y,t)$ and we add 0.1 to ensure that some filtering takes place on all pixels even if $D(x,y,t)$ is 0. Note that the parameters d_{max} is fixed for an image and one could experiment with it to get better results. Note also that the relation between k and D is linear. Other (e.g., positive/negative quadratic)

relations are possible to enhance (adapt) the performance of (2).
$D(x, y, t)$ can be calculated using

$$D(x, y, t) = \left|I_\eta(x, y, t) - \hat{I}(x, y, t-1)\right| * A_W$$

Where A_W is a spatial averaging filter (see lab assignment 4) of size W and $*$ is convolution operator. Equation (4) suggests differencing $I_\eta(t)$ and $\hat{I}(t-1)$ and then smoothing the absolute value by applying a spatial averaging filter.
Develop a modular Matlab program for temporal video noise filtering as follows:

1. Download a video sequence I from the course website.
2. Add to each image Gaussian noise of $\sigma_\eta^2 = 0.05$ to create I_η. To add such noise, you may use Matlab's imnoise().
3. Create a difference frame $D(t)$ by using A_3 (i.e., W=3 for the averaging filter).
4. Calculate the control frame k(t) by choosing $d_{max} = 31$.
5. Find $\hat{I}(t)$ for all frames of the input video sequence.
6. Calculate the PSNR between $\hat{I}(t)$ and I(t) and plot it for all the frames of $\hat{I}(t)$. The following Matlab function calculates the PSNR between images I and J.
7. Add to the PSNR plot, from the previous step, the PSNR results of the Gsussian filter from the previous question.
8. Measure the computation time of the temporal filter. Use 'tic' and 'toc' in Matlab.
9. Comment on your results using the following criteria:
    - Subjuective evaluation (visual comparison of the noise reduced, noisy, and original images)
    - Objective evaluation based on the PSNR, and
    - Computation time of both spatial and temporal filters.

Repeat the above for a second but very different video sequence with different content (e.e., different global motion) and see if your observation above with the first video sequence are confirmed.

2.1 Motivation

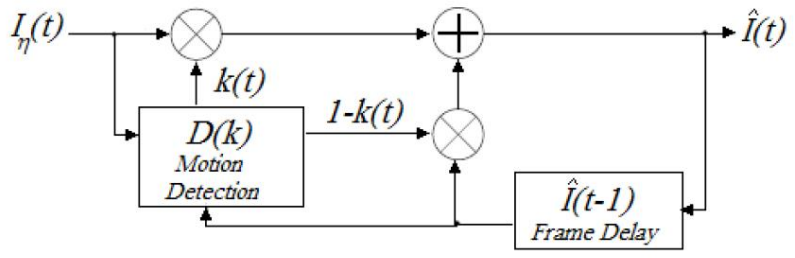
This question aims to help us to understand the theory of temporal recursive video noise filter.

2.2 Objectives

We need to separate a video into frames then add Gauassian noise. Then, we need to apply a temporal recursive video noise filter to remove the noise. Finally, we need to calculate the psnr which reflects the denoising effect.

2.3 Theory

Temporal video noise filters: The following figure shows the basic concept of a first-order temporal recursive video noise filter:



Let $I_\eta(t)$ be the noisy image of a video signal at time t and $I_\eta(x,y,t)$ be the pixel intensity in $I_\eta(t)$ at the spatial location (x,y). The filter output is

$$\widehat{I(t)} = k(t)I_\eta(t) + \big(1 - k(t)\big)I(\widehat{t-1}).$$

Where $I(\widehat{t-1})$ is the noise-reduced image at time t-1 (delayed by 1 memory unit, i.e., 1 frame). $k(t)$ is determined by the amount of motion for each pixel between t and t-1. $k(t)$ is called the coefficient (or control) frame and each of its elements takes a value between 0 and 1. Hence, the pixel-form of (1) is

$$\hat{I}(x,y,t) = k(x,y,t)I_\eta(x,y,t) + (1 - k(x,y,t))\hat{I}(x,y,t-1)$$

One way to calculate $k(x,y,t)$ is using

$$k(x,y,t) = 0.1 + 0.9\frac{D(x,y,t)}{d_{max}}$$

Where $D(x,y,t)$ is the amount of motion (i.e., difference) detected between $I_\eta(x,y,t)$ and $\hat{I}(x,y,t-1)$ and d_{max} is the maximum motion difference expected. Note that the amount of detected D (linearly) increases or decreases the value of parameter k.
We use $31 \le d_{max} \le 255$ to normalize $D(x,y,t)$ and we add 0.1 to ensure that some filtering takes place on all pixels even if $D(x,y,t)$ is 0. Note that the parameters d_{max} is fixed for an image and one could experiment with it to get better results. Note also that the relation between k and D is linear. Other (e.g., positive/negative quadratic) relations are possible to enhance (adapt) the performance of (2).
$D(x,y,t)$ can be calculated using

$$D(x,y,t) = \left|I_\eta(x,y,t) - \hat{I}(x,y,t-1)\right| * A_W$$

Where A_W is a spatial averaging filter (see lab assignment 4) of size W and * is convolution operator. Equation (4) suggests differencing $I_\eta(t)$ and $\hat{I}(t-1)$ and then smoothing the absolute value by applying a spatial averaging filter.

2.4 Method

The resulting Matlab code I developed is as follows:

```
% ELEC6631, Digital Video Processing
% Matlab code for Lab assignment 4 question 2
% Xiaowen Ke on 25.03.2017
clear all;
close all;
clc;

tic
% Load the video
input_video='stefan_cif.avi';
video=VideoReader(input_video);
I=read(video);
N_frames=size(I,4);
% Process of the first frame
I_noised(:,:,:,1)=I(:,:,:,1);
I_denoised(:,:,:,1)=0.1.*I_noised(:,:,:,1)+(1-
0.1).*I_noised(:,:,:,1);
% Temporal recursive video noise filter
G=fspecial('average',3);
for t=2:N_frames
    % Add the gaussian noise

I_noised(:,:,:,t)=imnoise(I(:,:,:,t),'gaussian',0,0.05);
    subplot(1,2,1),imshow(I_noised(:,:,:,t));
    title('Noisy video');
    drawnow
    % Temporal recursive video noise filter
    D(:,:,:,t)=abs(I_noised(:,:,:,t)-I_denoised(:,:,:,t-
1));
    D(:,:,:,t)=imfilter(D(:,:,:,t),G);
    K(:,:,:,t)=0.1+0.9.*(D(:,:,:,t)./31);
    I_denoised(:,:,:,t)=K(:,:,:,t).*I_noised(:,:,:,t)+(1-
K(:,:,:,t)).*I_denoised(:,:,:,t-1);
    subplot(1,2,2),imshow(I_denoised(:,:,:,t));
    title('Denoised video');
    drawnow
end
% Calculate the psnr and psnr gain
for t=1:N_frames
    PSNR(t)=psnr(I(:,:,:,t),I_denoised(:,:,:,t));
    PSNR1(t)=psnr(I(:,:,:,t),I_noised(:,:,:,t));
    PSNR_gain(t)=PSNR(t)-PSNR1(t);
end

figure(2)
plot(PSNR,'green');
hold on
plot(PSNR_gain,'blue');
legend('PSNR','PSNR gain');
title('PSNR and PSNR gain');
xlabel('Frame numner'),ylabel('PSNR gain');
toc
```

2.5 Results and discussion

Figure 9 shows the denoising effect of the temporal recursive video noise filter when the Gaussian noise power $\sigma_\eta^2 = 0.05$. As we can see from the figure, they are the last frame of the video, which has a very bad denoising effect. The reason is that the correlation of two neighbor frames (contain fast moving objects) is low. Therefore, the effect of the temporal recursive video noise filter is bad. And the smearing effect is severe so that we cannot track the moving motion with the video gong on. As a result, the denoising is going to be meaningless.

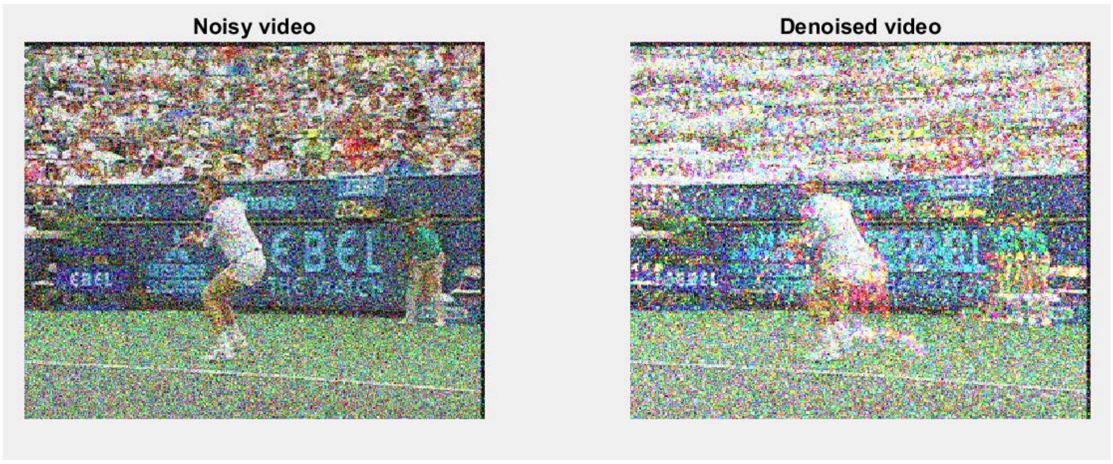


Figure 9 Denoising effect.

Figure 10 reflects the reason mentioned above. Since this is a video with fast moving motions, the beginning several frames have good PSNR value, then drops very fast.

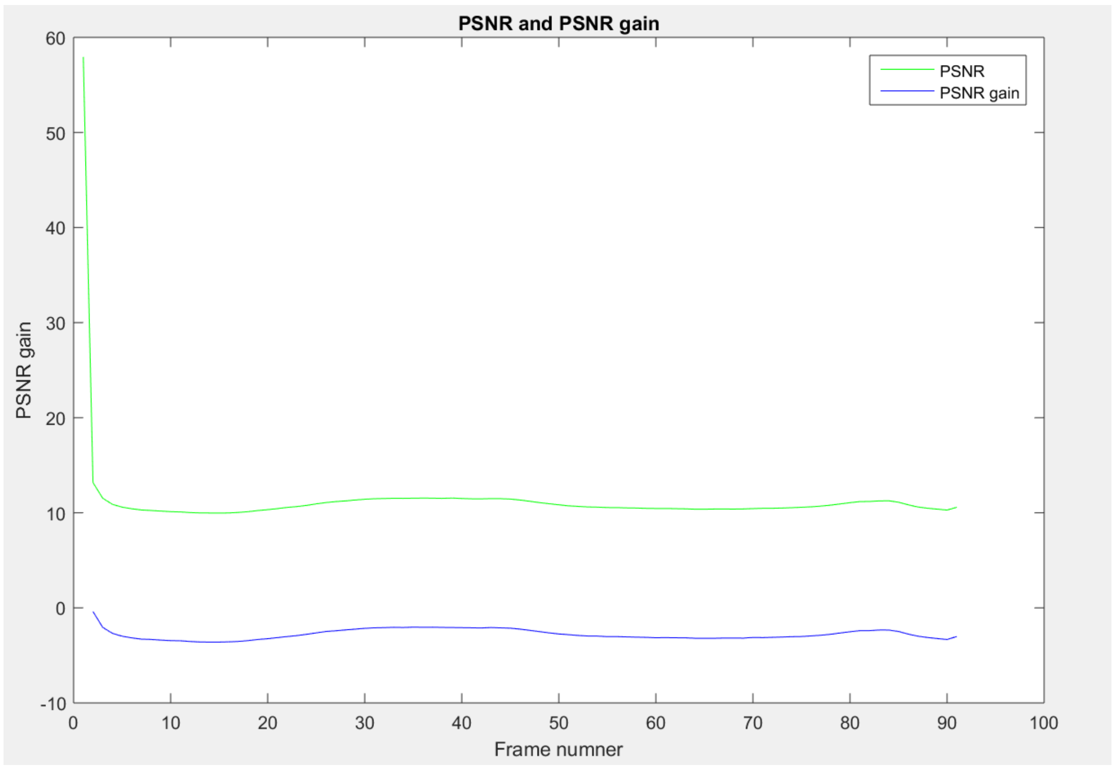


Figure 10 PSNR and PSNR gain.

2.6 Conclusion

From the above results, we can conclude that the Temporal video noise filters is generally only applicable with the random noise of the image filter, or low-speed moving target detection. It is worth to mention that the value of the K is closely related to the speed of the target movement (the size of the image content change). When the target movement is fast, the K value is large. When K is large, the noise reduction effect is obvious. But the residual image of the moving image is also very serious. On the contrary, reduce the K value, can reduce the residual image, but the noise reduction effect is poor. However, the residual image will also affect the denoising effect of following frames.