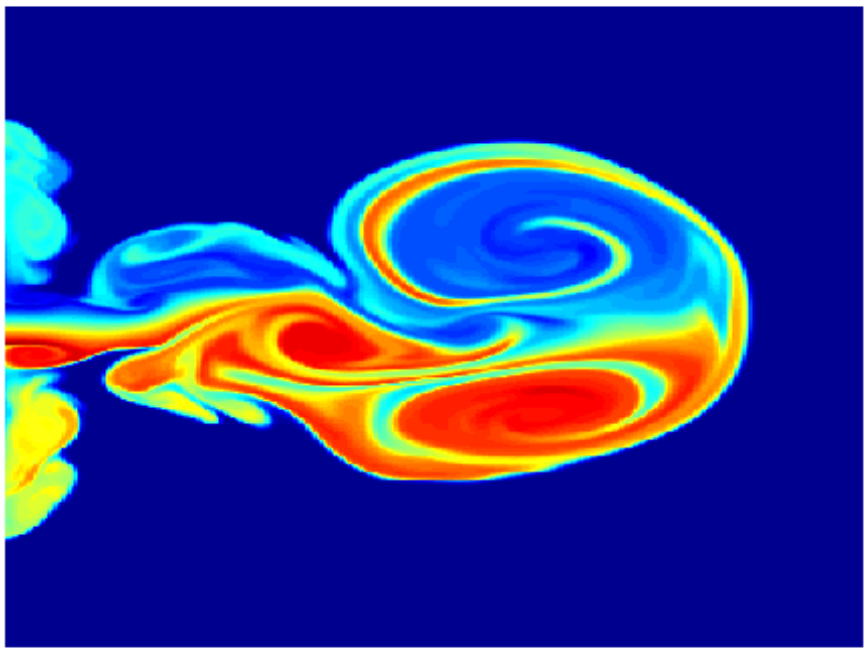
HW2

Hongru Li 95318168

1 Color Images

(a)

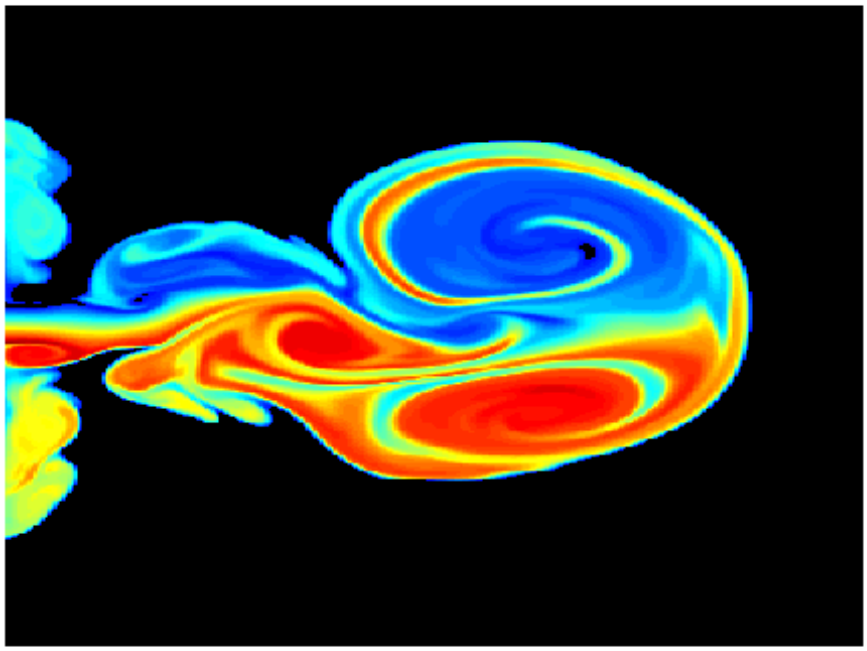


(b) Map is 64x3 double; X is 400x300 double

(c) X is gray-level(64 levels) image and map is how to represent each gray-level in RGB; high-level tends to use red and low level tend to use Blue

(d) Code:

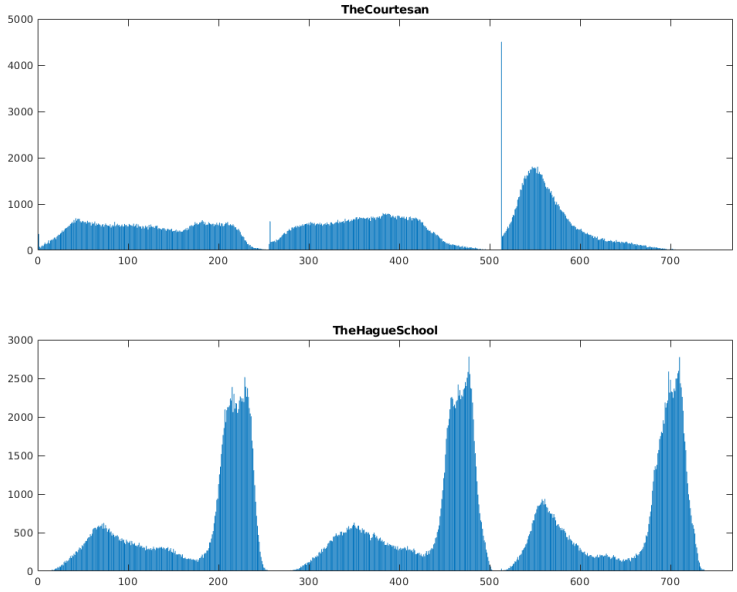
```
for i=1:8
map(i, 3)=0;
end
```



(e)

```
function hist = colorhist(image, n1, n2, n3)
%COLORHIST Summary of this function goes here
%   0 is put into bin 1
image = uint16(image);
hist = zeros(1, n1+n2+n3);
offset = [0, n1, n1+n2];
[m,n,c]=size(image);
for ch=1:c
   for w=1:m
      for h=1:n
         hist(offset(ch)+image(w,h,ch)+1) = hist(offset(ch)+image(w,h,ch)
+1)+1;
      end
   end
end
end
```

(f) Yes, the histogram are very different. The Hague School has brighter values(large RGB pixel values). The Courtesan has higher contrast.



2 Homomorphic Filtering

(a) The fundamental of this function is the following low pass filter:

F = 1/(1+(w/cutoff)^2n)

The five parameters in this function are: boost, CutOff, order, lhistogram_cut, uhistogram_cut

Boost controls the ratio of high frequency compared to low frequency in the filter; Cutoff is the cut off frequency in low/high pass filter; order is the ‘n’ in the above filter function, the higher n makes filter closer to an ideal filter. Histogram_cut is something used to cut the extreme value in reconstructed image from ifft.

(b)

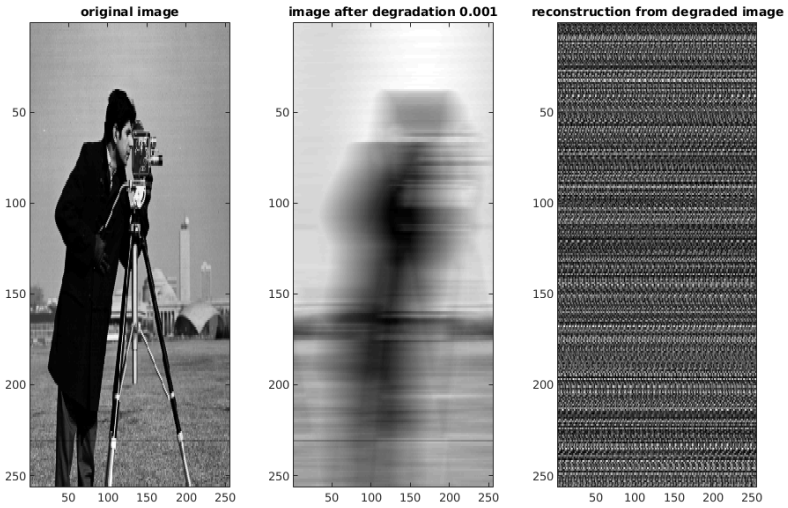
newim = homomorphic(Img, 0, 0.05, 10, 0, 5)

(c) The high boost filter calls the lowpass filter to make either low pass or high pass filter. For the low pass filter itself, f(0)=1 and f(+infinite)=0. The boost is the ratio of result high pass to low pass. So for boost <=1 f(0)=1 and f(+infinite)=1/boost; for boost>1, f(0)=1, f(+infinite)=boost

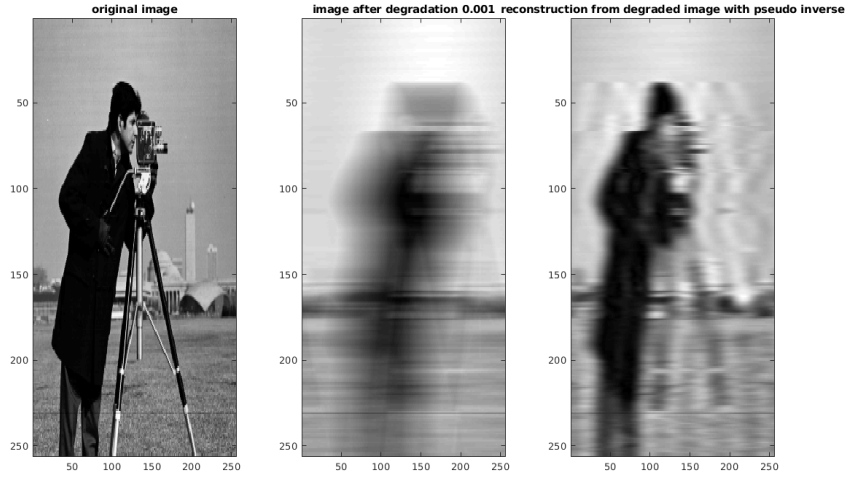
3 Estimation by Mathematical Modelling

(a) It add blur of horizontal motion by multiple the Fourier of the image by the H(u, v) which was derived in class. To reconstruct, the Fourier of the degradation image is divided by H(u, v). It can also add noise to the image but since st = 0.0 by default there is no noise added.

(b) Not able to reconstruct. Sine if degradation function has small value then noise will dominate the estimate.

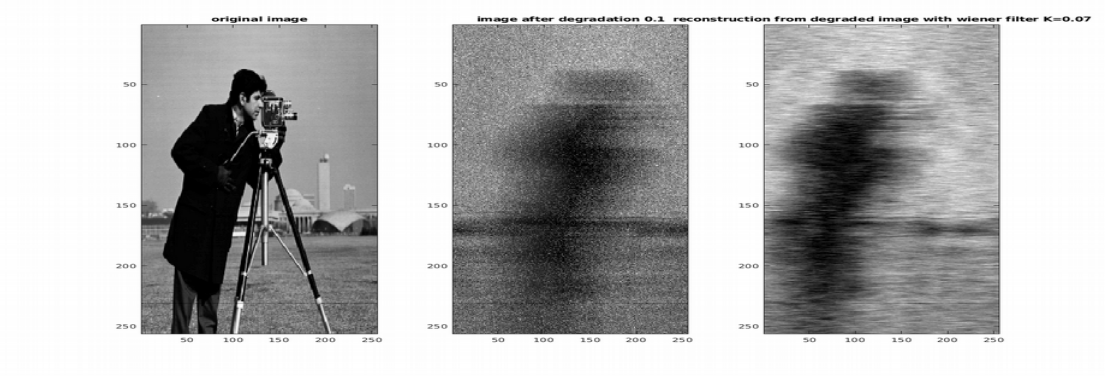


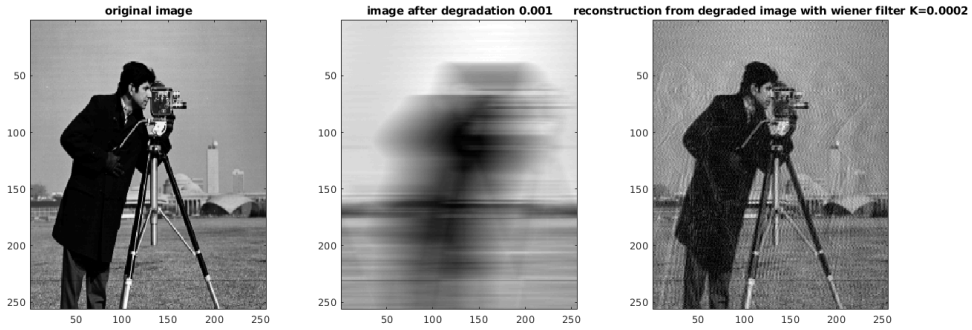
(c) ‘Divided by near zero’ mainly happens at high frequency but natural image has most of energy at low frequency. Thus set cutoff frequency to discard high frequency component can avoid divided by near zero value problem.



(d) The below image was reconstructed setting threshold = 0.03

(e) wFilt = (1./H).*(H1./(H1+K));





(f) The pseudo inverse completely give up restore Fourier when |H(u,v)| is below threshold. However Wiener attempt to restore no matter how small H(u,v) is. I think pseudo inverse is especially worse at middle

frequency where H(u,v) is lower than threshold but is not that small.

4 Restoration of Halftone

(a)%% Loading the image and computing the FFT
A = im2double(imread('halftone.png'));

figure; imshow(A);
title('Original Image');
F = fftshift(fft2(A));
mag = abs(F); %Get the absolute value of fourier

%%
ratio = 0.05; %a threshold to find the 'peeks' in fourier
mask = mag > ratio*max(mag(:)); % find the 'peaks'
mask = bwmorph(mask,'dilate',2); % link close cluster points together
mask = bwmorph(mask,'shrink',100); % find the single core to represet clusters
mask(size(F,1)/2+1,size(F,2)/2+1) = 0; % exclude the (0,0), this the original image low frequency compenent not halftone

%%
rad = 10; %a mask the radius of circle to cover the halftone frequency

```
component
[x,y] = find(double(mask)); %get the circle center position
F2 = F;

for i=1:size(x,1)
   H = notch('ideal',size(F,1),size(F,2),rad,y(i),x(i)); % a high pass filter to
remove the frequency caused by halftone
   F2 = F2.*H;
End
```

(b)

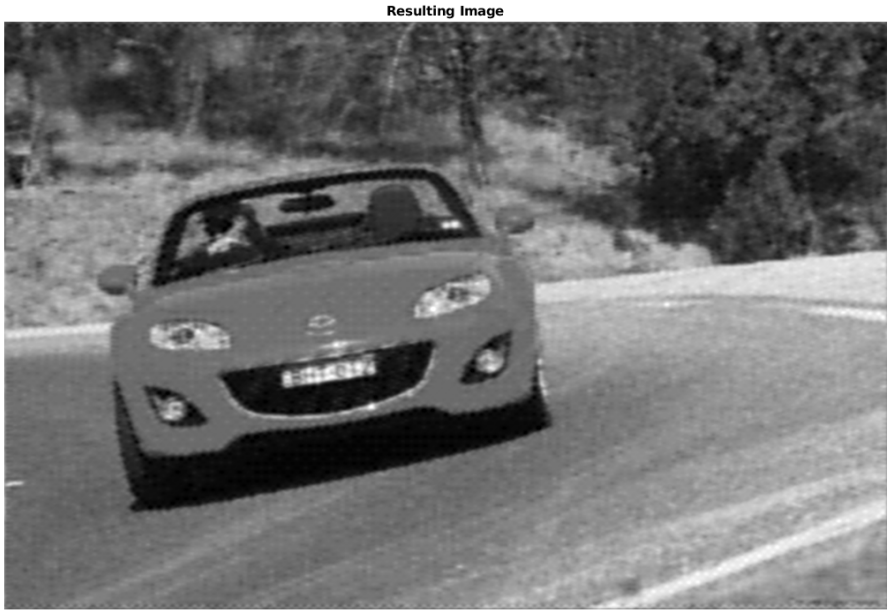
I did 4 adjustments:

```
%%
ratio = 0.015; %changed to 0.015
mask = mag > ratio*max(mag(:)); %
mask = bwmorph(mask,'dilate',2); %
mask = bwmorph(mask,'shrink',100); %
mask(size(F,1)/2+1,size(F,2)/2+1) = 0; %

%%
rad = 30; % change to 30 to better eliminate the peeks
[x,y] = find(double(mask)); %
F2 = F;

for i=1:size(x,1)
   H = notch('gaussian',size(F,1),size(F,2),rad,y(i),x(i)); % change to use
gaussian, the ideal filter will introduce artifact
   F2 = F2.*H;
end

H = 1-notch('gaussian',size(F,1),size(F,2),80,size(F,2)/2,size(F,1)/2); % pass
through low pass filter to remove halftone at high frequency
F2 = F2.*H;
```



5 Geometric Image Manipulation

(a)

```
%{
RAFEEF GARBI - EECE 570: Fundamentals of Visual Computing 2019
HW 2 - PROBLEM 5- Student File
%}
clear all; close all; clc;

%%
pltstp=40; %plot step of polar
maxAngRange=0:10:720; % step size of twist

A=uint8(255*checkerboard(16));% initial checkerboard
subplot(131); imshow(A);

%%
%Mid point of the image
midr=ceil(size(A,1)/2); % rows increase downwards
midc=ceil(size(A,2)/2); % columns increase to the right

hold on
plot(midc,midr,'r.','markersize',20);
```

```
text(midc+3,midr+3,'origin','FontSize',20,'color',[1 0 0]);
nRows=size(A,1);
nCols=size(A,2);

%%
[r,c]=ndgrid(1:nRows,1:nCols);
x=c-midc;
y=midr-r;

%%
[th,rho]=cart2pol(x,y);% change to polar coordinate

for maxAngDeg=maxAngRange,

    maxRho=max(rho(:));
    maxAng=maxAngDeg*pi/180;%max angle in rad

    th2=th+(rho/maxRho)*maxAng;
    rho2=rho;

    subplot(132);hold off
    polar(th2(1:pltstp:end),rho2(1:pltstp:end),'go')%plot the polar coordinate

    [x2,y2]=pol2cart(th2,rho2);%convert from polar to x-y

    c2=x2+midc;
    r2=midr-y2;

    % find the corresponding points in A and B
    B=uint8(zeros(size(A)));
    for r=1:nRows, %
       for c=1:nCols,
          if r2(r,c)>=1 && r2(r,c)<=nRows && ...
                c2(r,c)>=1 && c2(r,c)<=nCols, % don't go out of A


             B(r,c,:)=A(round(r2(r,c)),round(c2(r,c)),:);% Points in B are rotated
from A
          end
       end
    end
    subplot(133)
    imshow(B);
    % set bacground of image to white
```

```
    set(gcf,'Color',[1 1 1]);

    pause(0.5)
end
```

(b) Some info is missing since coordinate rounding, So I pass the image to a median filter and replace the mission points using the pixel after passing median filter

```
A = imread('swirled.bmp');

%Mid point of the image
midr=ceil(size(A,1)/2); % rows increase downwards
midc=ceil(size(A,2)/2); % columns increase to the right

nRows=size(A,1);
nCols=size(A,2);

[r,c]=ndgrid(1:nRows,1:nCols);
x=c-midc;
y=midr-r;

[th,rho]=cart2pol(x,y);% change to polar coordinate

maxAngDeg = 720;

maxRho=max(rho(:));
maxAng=maxAngDeg*pi/180;%max angle in rad

th2=th+(rho/maxRho)*maxAng;
```

```
rho2=rho;

[x2,y2]=pol2cart(th2,rho2);%convert from polar to x-y

c2=x2+midc;
r2=midr-y2;

% find the corresponding points in A and B
B=uint8(zeros(size(A)));
for r=1:nRows, %
   for c=1:nCols,
      if r2(r,c)>=1 && r2(r,c)<=nRows && ...
            c2(r,c)>=1 && c2(r,c)<=nCols, % don't go out of A
         B(round(r2(r,c)),round(c2(r,c)),:)=A(r,c,:);% Points in B are rotated
from A
      end
   end
end

B2=medfilt2(B, [6,6])
for r=1:nRows, %
   for c=1:nCols,
      if B(r,c,:)==0
         B(r,c,:) = B2(r,c,:);
      end
   end
end
imshow(B)
```

6 Removing Occlusions from Video

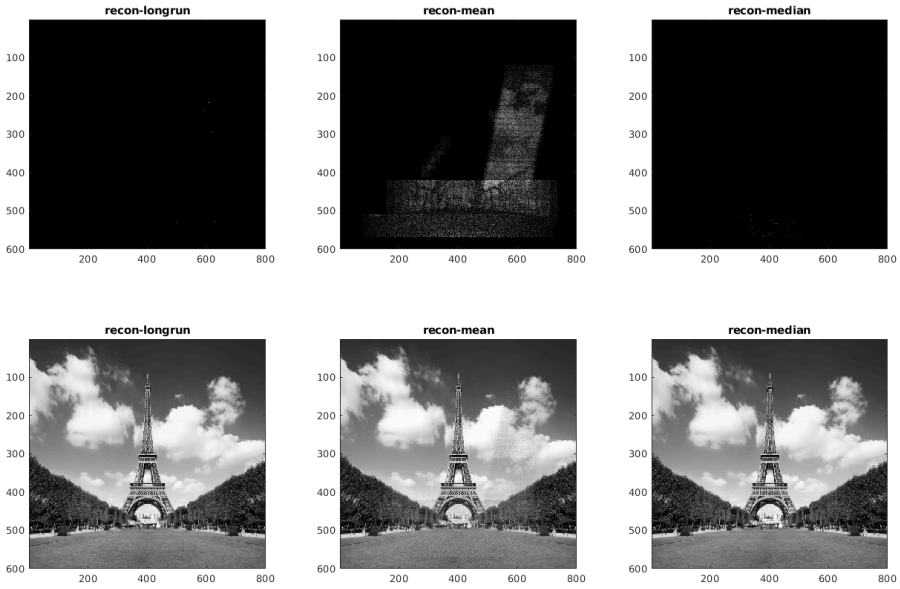
(a) Accumulate difference record the pixel value last for longest time for each pixel. Reconstruct the background image use the pixel value last longest time to represent the value in background image.

(b)

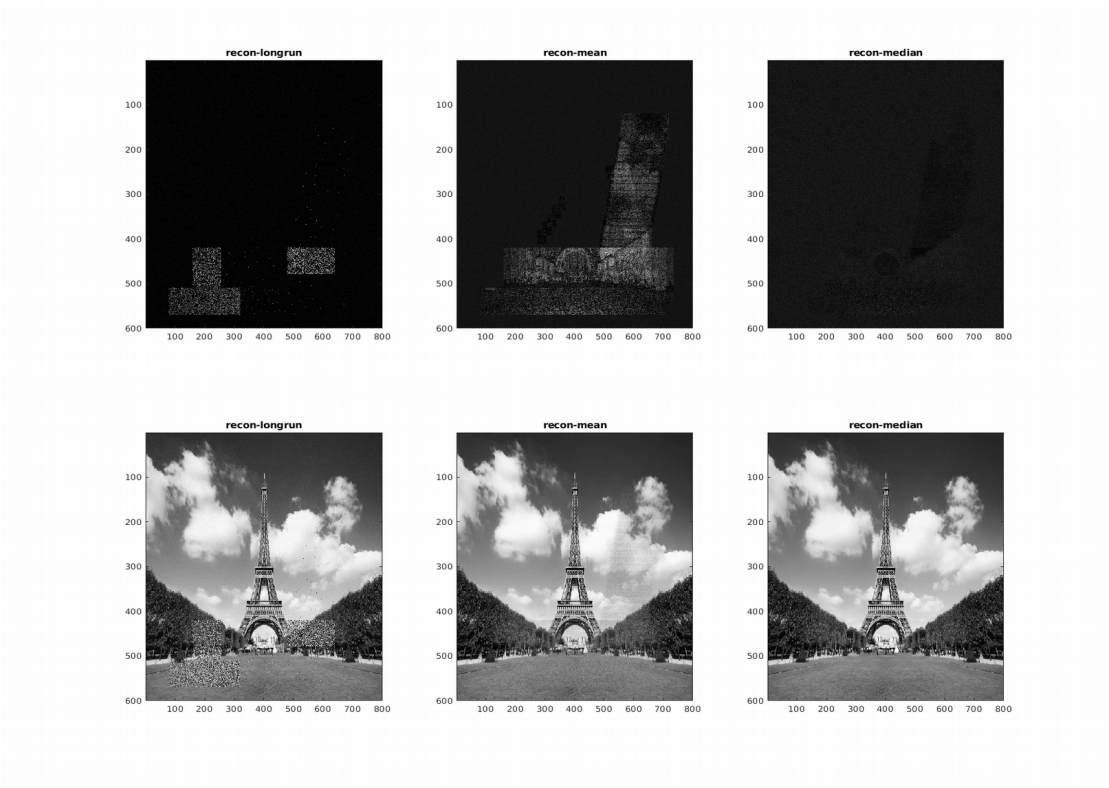
```
%% accumulate for median and mean
vid_mean = mean(vid, 4);
vid_median = median(vid, 4);
```

When there is no noise, median and ‘longrun’ method is better than ‘mean’ method. When you compute the mean, you take the outliers(which is not the

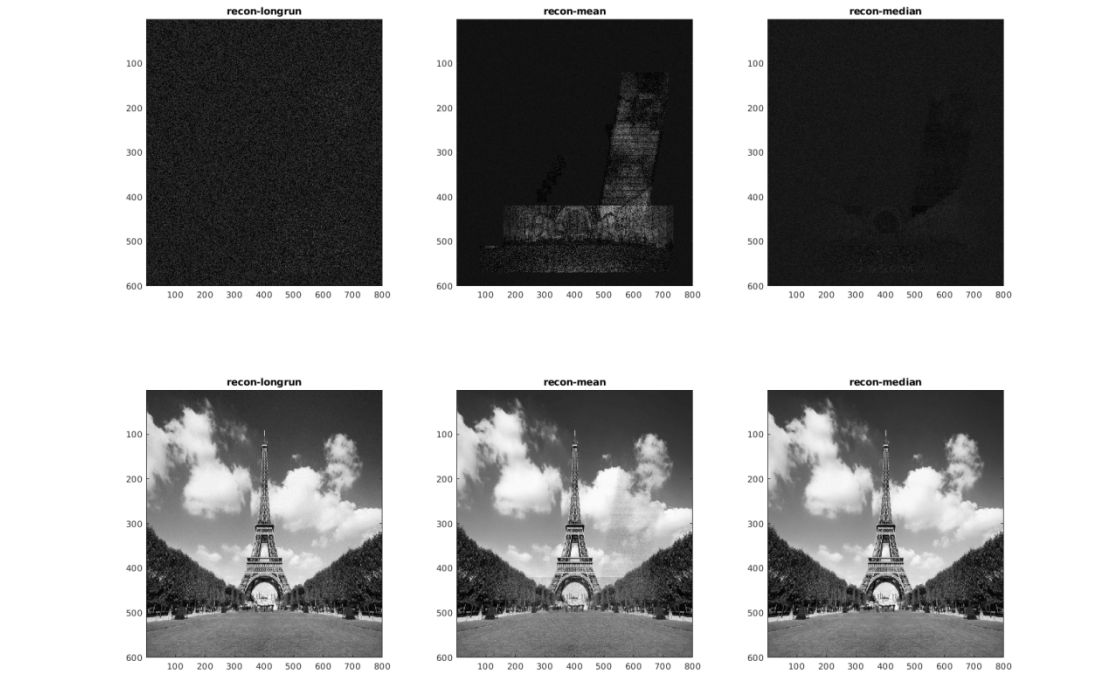
background) into your calculation, and thus the image reconstructed is disturbed by the non-background values.



(c) nz the the factor of noise. And the ‘longrun’ method is not as good as before since noise make it hard to find the value last for longest time with very small threshold value. For mean method, still you can see the artifact of averaging in reconstruct image. The median method seems robust for the result.



(d)



I changed the T=0.06 and the longest run method get improved. But the absolute error is noisy for the background. Both mean or median method makes the value of absolute error uniform for the entire image. Still the best method is taking median.