ELL715 - Assignment 5

Report

Deepali Gupta
2013MT60079

1. (Code in q1.m)
   - First, prediction was done using average of left and top neighbouring pixels.
   - Next, optimal weights were found for above using minimum least squares method. The formula used was :

     Let X be the matrix of all possible length 5 sequences, a be the vector of linear prediction coefficients and p be the values we wish to predict. We set up the following equation:

     Which we write as:

     $$Xa = p$$

     Now multiply both sides by X^T

     $$X^T Xa = X^T p$$
     $$(X^T X)^{-1} X^T Xa = (X^T X)^{-1} X^T p$$
     $$a = (X^T X)^{-1} X^T p$$

   - The same procedure was followed using all nearest neighbours.
   - The entire process was repeated for a 100 x 100 grayscale image.
   - Results :

0	0	1	5	6
0	0	1	5	6
2	2	4	7	8
3	3	7	4	2
6	6	5	1	0

Matrix

Prediction using two neighbours:
f(i,j) = (0.2909)*f(i-1,j) + (0.6662)*f(i,j-1)

Prediction using 8 nearest neighbours:

f(i,j) = (-1.6814)*f(i-1,j-1) + (0.6552)*f(i-1,j) + (0.0490)*f(i-1,j+1) + (0.3688)*f(i,j-1) + (-0.3838)*f(i,j+1) + (0.7092)*f(i+1,j-1) + (0.6309)*f(i+1,j) + (-0.2882)*f(i+1,j+1)

Image

Prediction using two neighbours:
f(i,j) = (0.4406)*f(i-1,j) + (0.5518)*f(i,j-1)

Prediction using 8 nearest neighbours:
f(i,j) = (-0.0649)*f(i-1,j-1) + (0.3450)*f(i-1,j) + (-0.1531)*f(i-1,j+1) + (0.3853)*f(i,j-1) + 0.3853)*f(i,j+1) + (-0.1527)*f(i+1,j-1) + (0.3404)*f(i+1,j) + (-0. 0812)*f(i+1,j+1)

2. (Code in q2.m)

Image

Salt and pepper noise was applied using MATLAB function imnoise(I, 'salt & pepper')

Image with salt and pepper noise

Contra harmonic filter was applied

Contra harmonic filter results with Q=-2

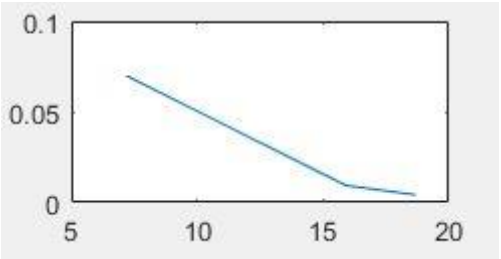
MSE = very large error

Contra harmonic filter results with Q=5
MSE = 0.0664

Median filter result
MSE = 0.0097

Adaptive median filter results
MSE = 0.0042

From the results, we gather that median filter is better than contra harmonic filter. Also, within local noise removal filters, median is the best, with the optimal one being the adaptive median filter as it gives the least mean square error.

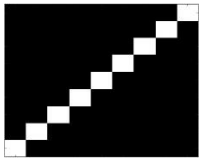


Plot of error against SNR

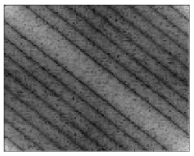
3. (Code in q3.m)

Original Image

Blurred image



Blurring filter mask

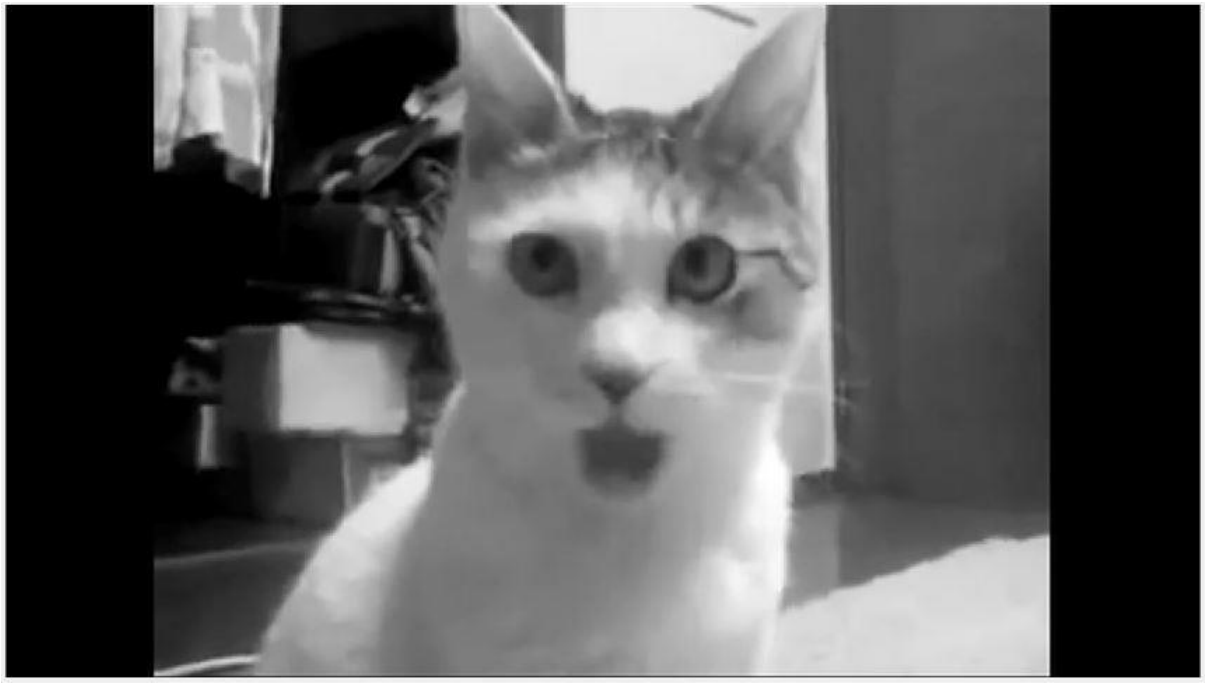


Blurring filter

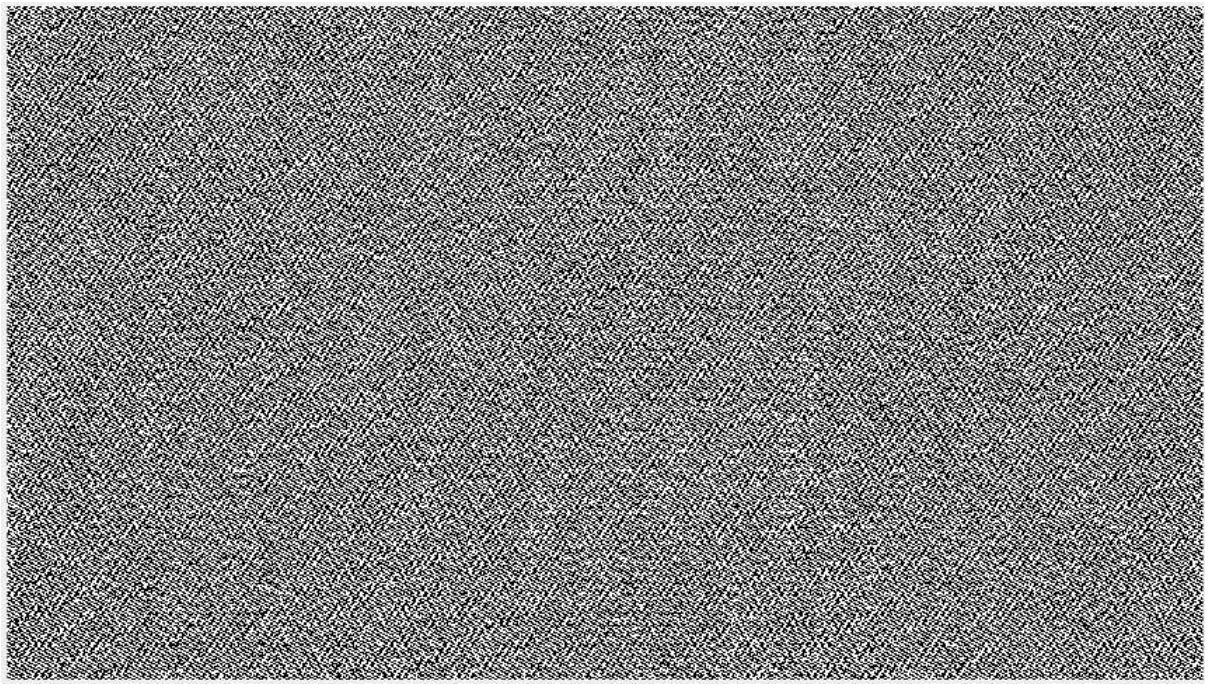
Inverse filter

Pseudo inverse filter

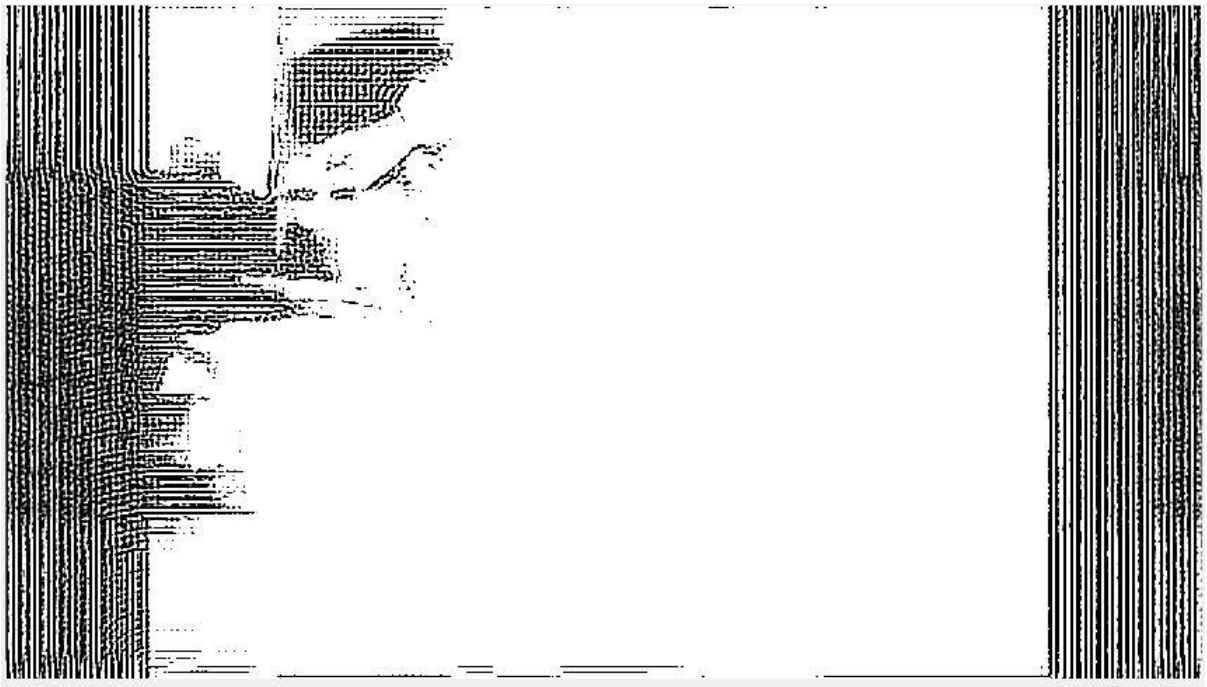
Using video frames:



Blurred image



Inverse filtering



Pseudo inverse filtering

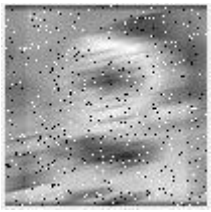
4. (Code in q4.m)

A motion blur was simulated that you might get from camera motion. Created a point-spread function, PSF, corresponding to the linear motion across 31 pixels (LEN=31), at an angle of 11 degrees (THETA=11). To simulate the blur, convolved the filter with the image using imfilter.

Original Image

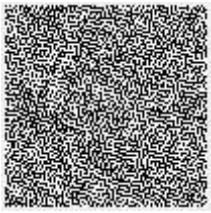
Blurred image

Recovery of blur without noise



Blurred and noisy image

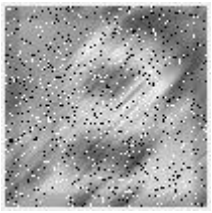
Recovery using noise histogram



Recovery assuming no noise

Recovery assuming constant noise K
The value of K was estimated by minimising the error and better perception of image

Recovery using CLS



After adding WGN

Recovery with noise variance estimate

Recovery assuming constant noise

Recovery using CLS