Image Noise Removal

Rizzo Rosetta

Outline

- Image quality metrics
  - MSE - PSNR
- Impulse noise removal
  - Median Filtering
  - Outlier detection
- Additive White Gaussian noise removal
  - Mean Filtering
  - Gaussian Filtering
- Adaptive Filtering
  - Kim-Lee Algorithm

Image quality metrics

- Image quality metrics can be used to evaluate the realistic image enhancement algorithms.
- Metrics can be classified according to the availability of a reference image, to which the distorted image can be compared.
  - *full-reference*: a complete reference image is assumed to be known.
  - *reduced-reference* : image is only partially available, in the form of a set of extracted features given as side information to help evaluate the quality of the distorted image.
  - *no-reference:* metrics look only at the image or video under test and have no need of reference information.

Peak Signal to Noise Ratio(PSNR)

- If reference image is available, the quality of an image enhancement method is evaluated using numerical techniques that attempt to quantify likeliness using image-to-image comparison.
- PSNR is the most popular method to evaluate pictures quality.

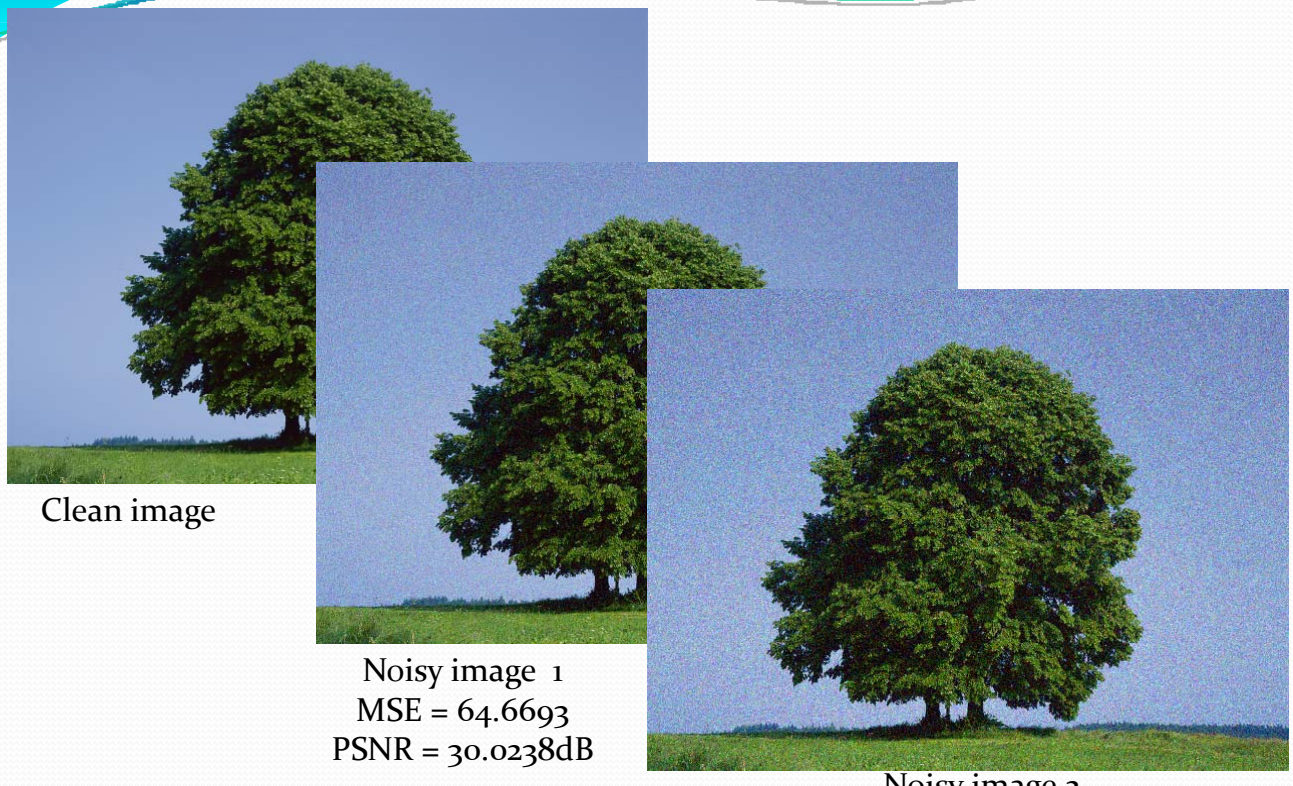
Peak Signal to Noise Ratio(PSNR)

- PSNR is obtained comparing the mean square error (MSE) to the maximum possible value of the luminance L (for a typical 8-bit image $L=2^8-1 = 255$) as follows:

$$PSNR = 10\log_{10}\frac{L^2}{MSE}$$

$$MSE = \frac{1}{N \cdot M}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[f(i,j)-F(i,j)\right]^2$$

- Where f(i,j) is the original value at pixel(i, j), F(i, j) is the reconstructed value, and M x N is the picture size.
- The result is a number expressed in decibels, ranging from 30 to 40 for medium to high quality images.



Clean image

Noisy image 1
MSE = 64.6693
PSNR = 30.0238dB

Noisy image 2
MSE = 84.2637
PSNR = 28.8744dB

Image noise

- Image noise is a random, unwanted, fluctuation of pixel values in an image.
- Noise sources are:
  - Photon Shot Noise
  - Dark Current (Thermal Noise)
  - Readout noise (Bias Noise)
  - KT/C Noise (Reset Noise)
  - Fixed Pattern Noise
  - Quantization Noise

Impulse Noise (salt and pepper noise)

- Each pixel in an image has probability $p/2$ ($0<p<1$) to be corrupted into either a white dot (salt) or a black dot (pepper)

$$Y(i,j)=\begin{cases}255 & \text{with probability } p/2\\ 0 & \text{with probability } p/2\\ X(i,j) & \text{with probability } 1\text{-}p\end{cases}$$

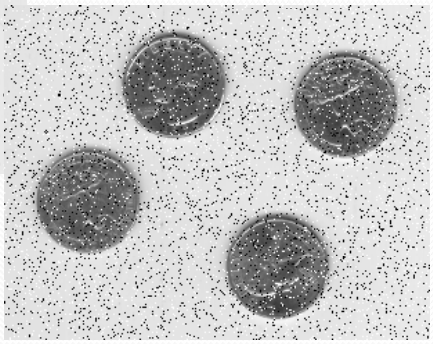
noisy pixels

clean pixels

Where:
- X = noise-free image,
- Y= noisy image
- $1 \le i \le H$
- $1 \le j \le W$

I – clean image

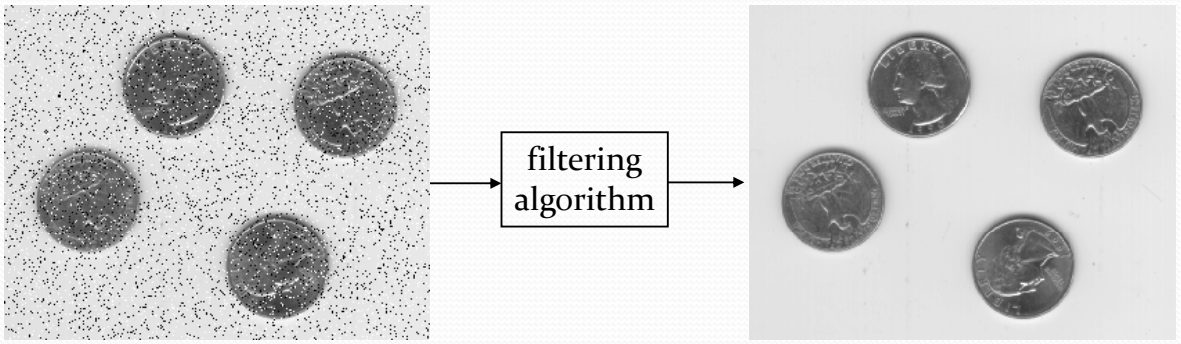
J – noisy image (2%)



K – noisy image (10%)

I = *imread*('eight.tif');
J = *imnoise* (I,'salt & pepper', 0.02);
K= *imnoise(I,*'salt & pepper', 0.1);

Impulse Noise Removal Problem



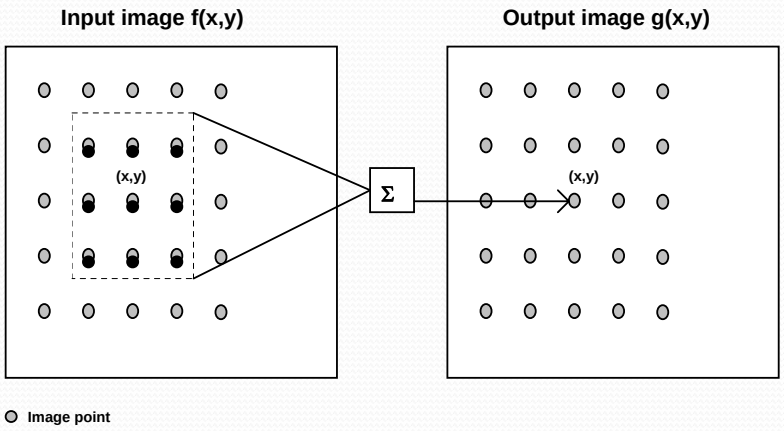
Noise Removal Techniques

- Linear filtering
  - Mean filter
  - Gaussian filtering
- Non-linear filtering
  - Median
- Adaptive algorithms

Linear filtering and convolution

2D convolution of an image *f(x,y)* with a filter *h(x,y):*

$$g(x,y) = \sum_{x'=-\infty}^{+\infty} \sum_{y'=-\infty}^{+\infty} h(x',y')\, f(x-x', y-y')$$



Mean Filter

- It is implemented by a local averaging operation where the value of each pixel is replaced by the average of all the values in the local neighborhood.

$\frac{1}{9} \times$

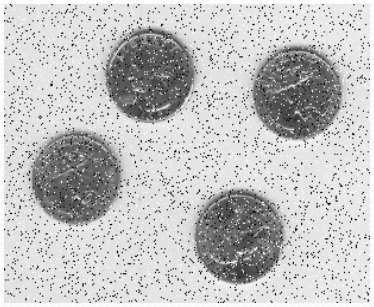
1	1	1
1	1	1
1	1	1

Filter *h*

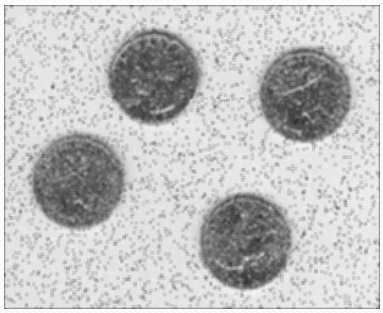
	p1	p2	p3
	p4	p5	p6
	p7	p8	p9

Image *f*

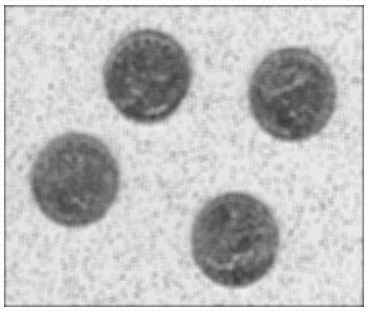
I - Clean image



K - Noisy image (10%)



A - Mean filering (3x3)

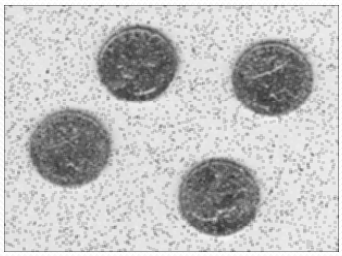


B - Mean filtering (5x5)

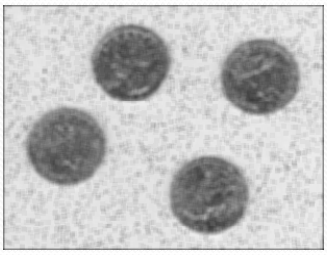
MATLAB CODE

```
I = imread('eight.tif');
K= imnoise(I,'salt & pepper',0.1);

filter_1 = fspecial('average', 3);
A = imfilter(K,  filter_1);

filter_2 = fspecial('average', 5);
B = imfilter(K,  filter_2);
```

Drawbacks of Mean Filter



A - Mean filering (3x3)



B - Mean filtering (5x5)

- These examples illustrate the two main problems with mean filtering:
  - A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighborhood.
  - When the filter neighborhood crosses an edge, the filter will interpolate new values for pixels on the edge and will create blur.

Median filtering

- Median filter is often a better filter for reducing noise than the mean filter, but it takes longer to compute.
- The median is calculated by first sorting all the pixel values from the surrounding neighborhood and then replacing the pixel being considered with the middle pixel value.
- Median filtering:
  - Preserve sharp edges.
  - Effective in removing impulse noise.

Median filtering

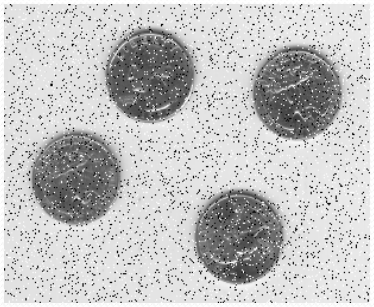
123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

115, 119, 120, 123, 124, 125, 126, 127, 150

Median value: 124

I - Clean image

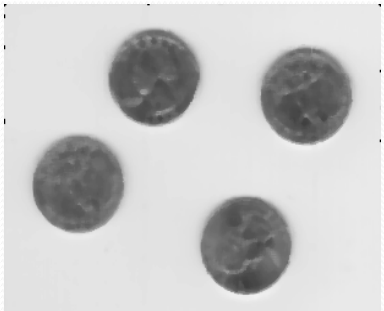


K - Noisy image (10%)

MATLAB CODE

```
I = imread('eight.tif');
K= imnoise(I,'salt & pepper',0.1);

C = medfilt2(K, [3 3]);
D = medfilt2(K, [5 5]);
```

C – Median filtering (3x3)



D – Median filtering (5x5)

Outlier detection

- The detection of noisy pixels in the image is performed by calculating the distances between the central pixel and its neighbors and counting the number of neighbors whose distance to the central pixel is not exceeding a predefined threshold.
- The number of pixels which are close enough to the pixel under consideration serves as an indicator whether the pixel is corrupted by impulse noise or not.

- The central pixel in the filtering window is detected as unaffected by impulse noise if there are *m* neighbors in the window, which are close enough.

- Otherwise, it will be replaced by the median of values in the window.

211	180	210
186	0	195
221	201	191

Advantages of Outlier detection

- The main advantages of this technique are its simplicity and enormous computational speed.
- Using outlier detection avoids introducing too much smoothing, caused by unnecessary filtering of the noise-free samples.

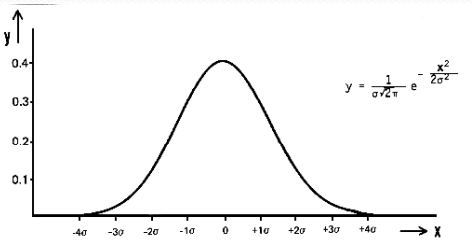
Additive White Gaussian noise (AWGN)

- Additive white Gaussian noise affect each color component and pixel position independently.
- Each pixel in an image is disturbed by a Gaussian random variable with zero mean and variance σ^2
- Plotting the amount of distortion of a pixel against the frequency with which it occurs produces a Gaussian distribution of noise.

$$Y(i,j) = X(i,j) + N(i,j),$$

$$N(i,j) \sim N(0,\sigma^2)$$

Where:
- X = noise-free image,
- Y= noisy image
- $1 \le i \le H$
- $1 \le j \le W$



I – clean image

J – noisy image (σ^2=0.002)

K – noisy image (σ^2=0.01)

I = *imread*('cameraman.tif');
J = imnoise(I,'gaussian',0, 0.002);
K= *imnoise(I*, gaussian', 0, 0.01);

I - Clean image

K - Noisy image (σ=0.1)

A - Mean filering (3x3)

B - Mean filtering (5x5)

MATLAB CODE

```
I = imread('cameraman.tif');
K= imnoise(I,'gaussian',0,0.01);

filter_1 = fspecial('average', 3);
A = imfilter(K,  filter_1);

filter_2 = fspecial('average', 5);
B = imfilter(K,  filter_2);
```


Gaussian Filters

- They are a class of linear smoothing filters with weights chosen according to a Gaussian function.
- It is a very good filter to remove noise drawn from a normal distribution.
- The 2D zero-mean discrete Gaussian function used for processing images is as given

$$g[i, j] = e^{\frac{-(i^2 + j^2)}{2\sigma^2}}$$

  - σ determines the width of the filter and hence the amount of smoothing

e.g. Gaussian kernel

$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Properties of Gaussian filters

- Amount of smoothing performed is the same in all directions, since Gaussian functions are isotropic.
- Gaussian filters smooth by replacing each image pixel with a weighted average of the neighboring pixels with weights monotonically decreasing with distance from central pixel. This property helps in retaining the edges.
- The width and hence the degree of smoothing of a Gaussian filter is parameterized by standard deviation.
- The larger the deviation, the wider the filter and the greater the smoothing and vice versa.

I - Clean image

K - Noisy image (σ=0.1)

MATLAB CODE

```
I = imread('cameraman.tif');
K= imnoise(I,'gaussian',0,0.01);

filter = fspecial ('gaussian', [3 3], 0.1);
A = imfilter(K,  filter);
filter2 = fspecial('gaussian', [3 3], 0.5);
B = imfilter(K,  filter2);
filter2 = fspecial('gaussian', [3 3], 1);
C = imfilter(K,  filter2);
```

A - Gaussian filtering (3x3) σ=0.1, μ =0

B- Gaussian filtering (3x3) σ=0.5, μ =0

C - Gaussian filtering (3x3) σ=1, μ =0

High vs. Low Gaussian filter strength

A - Gaussian filtering
(3x3) σ=0.1 , μ =0

B- Gaussian filtering
(3x3) σ=0.5, μ =0

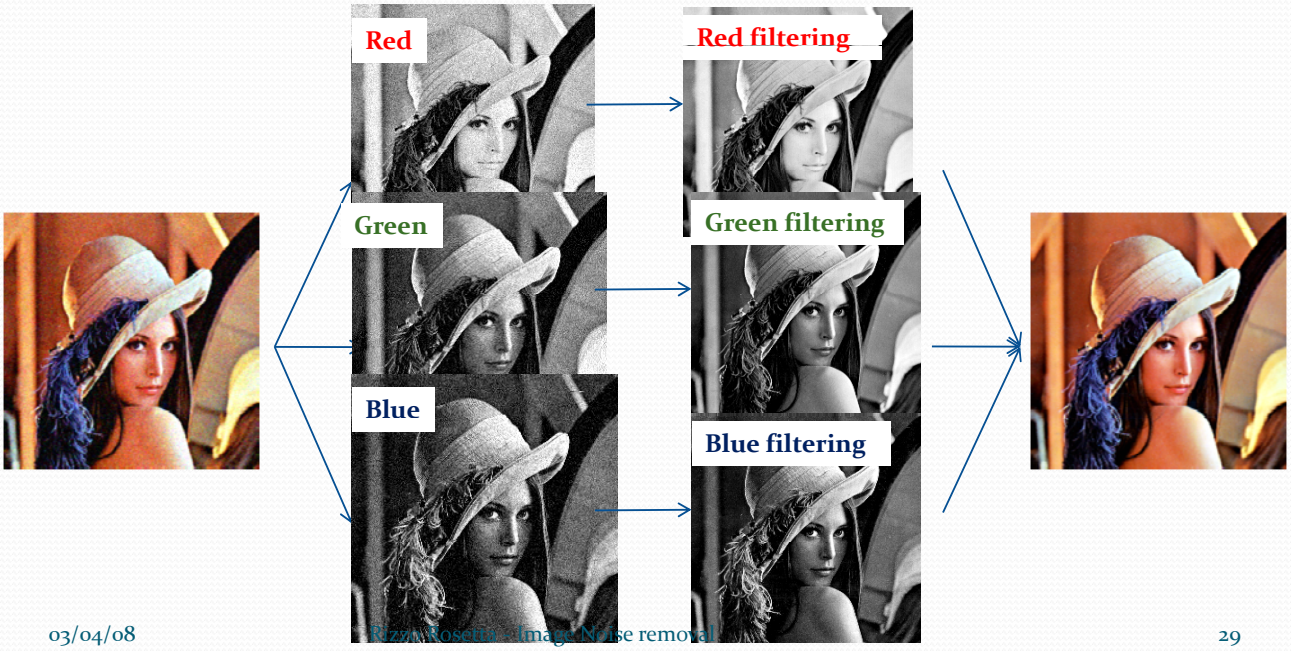
C - Gaussian filtering
(3x3) σ=1, μ =0

Mean vs. Gaussian filtering

A - Mean filtering (3x3)
PSNR= 31.6dB

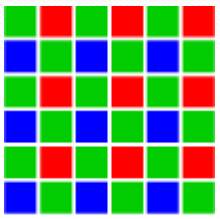
B - Gaussian filtering (3x3)
σ=0.5, μ =0
PSNR= 31.6 dB

RGB image filtering



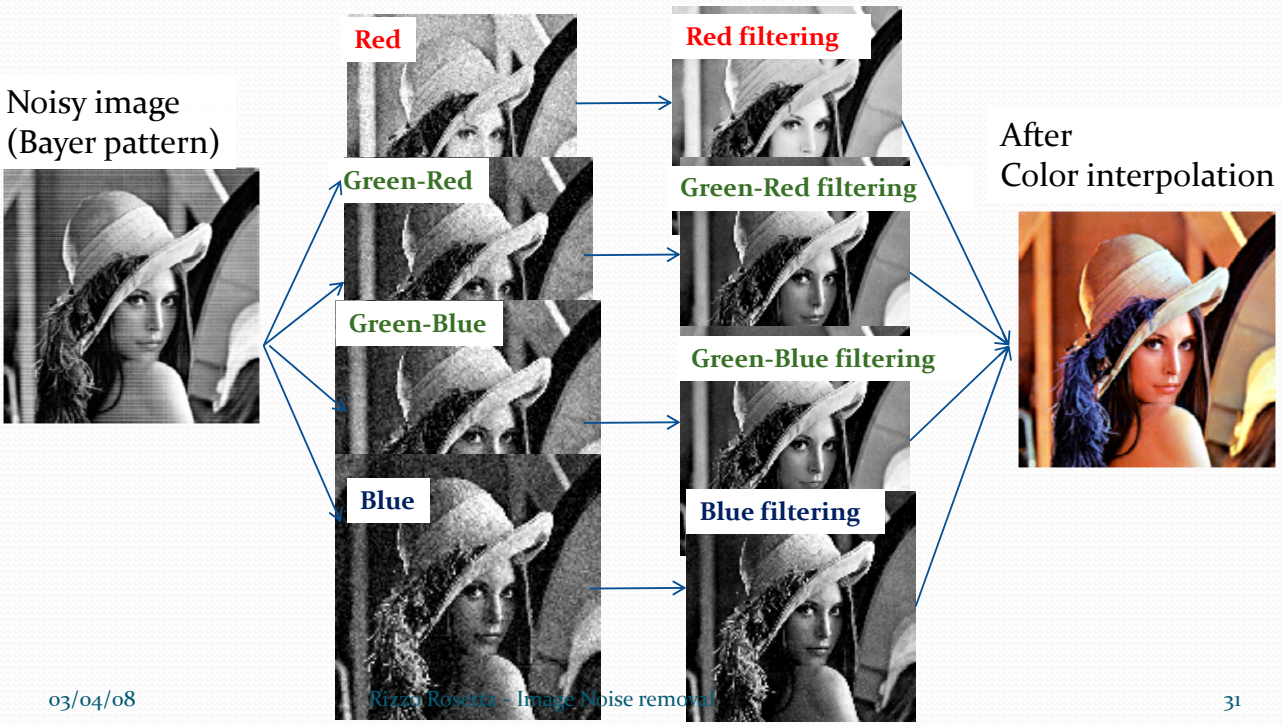
Synthetic Bayer pattern image

MATLAB code



```
function RGB_2_Bayer(image_name)
image = imread(image_name);
R = image(:,:,1);
G = image(:,:,2);
B = image(:,:,3);

bayerPattern = uint8(zeros (size(image, 1), size(image,2)));

for (ii = 1 : 2 : (size(image,1)))
    for (jj= 1 : 2 : (size(image,2)))
        bayerPattern(ii,jj) = G(ii, jj);          %GR
        bayerPattern(ii,jj+1) = R(ii, jj+1);      %R
        bayerPattern(ii+1,jj) = B(ii+1,jj);       %B
        bayerPattern(ii+1,jj+1) = G(ii+1,jj+1); %GB
    end
end

name = ['BP_',image_name,'.png'];
imwrite(bayerPattern,name,'png');
end
```

Bayer pattern image filtering



Noise estimation

- Gaussian Noise reduction filter strength must be *adaptive*. Filter strength must scale with the level of noise in the image.
  - High noise level -> High filter strength.
  - Low noise level -> Low filter strength.

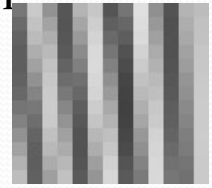
- But:
  - Textured areas must be not heavily filtered in order to maintain details.
  - Flat areas can be more heavily filtered to reduce pixel fluctuations.

Noise estimation

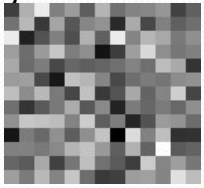
- Fine texture hide noise (*texture masking*)
- In flat areas pixel fluctuations are supposed to be caused exclusively by random noise: for this reason flat areas are better than textured zone to estimate noise level and avoid noise overestimation.
- To locate flat areas we can use a *texture detector.*

Kim-Lee Algorithm (1)

- "*Image feature and Noise Detection Based on Statistical Hypothesis tests and Their Applications in Noise Reduction*" Y.-H. Kim e J. Lee
- The fundamental idea behind the proposed detection algorithm is based on the observation that there is a strong sample correlation in at least one direction in an image feature area whereas no significant sample correlation appears in a noisy area.



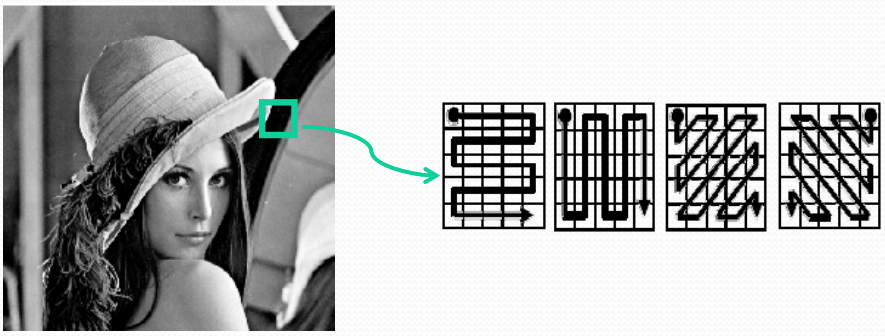
Edge - σ^2=1000



Flat area - σ^2=1000

Kim-Lee Algorithm (2)

- Kim-Lee algorithm implements a feature and noise detector based on statistical hypotheses and a statistic test.
  - For each pixel x_{ij} the algorithm verifies its spatial correlation with its neighbor samples.
    - If there is a strong sample correlation in at least one direction -> image feature
    - else –> flat area (random noise)



Kim-Lee Algorithm (3)

- The degree of feature (denoted as δ_f) for each pixel x_{ij} is introduced. It represents heuristically whether the pixel x_{ij} with its neighborhood forms an image feature, purely random noisy data, or between, as following:
  - $\delta_f(i,j) = 0$ -> x_{ij} is an image feature
  - $\delta_f(i,j) = 1$ -> x_{ij} is a random noisy data
  - $0 < \delta_f(i,j) < 1$ -> image feature with certain degree

flat area
textured area
intermediate area

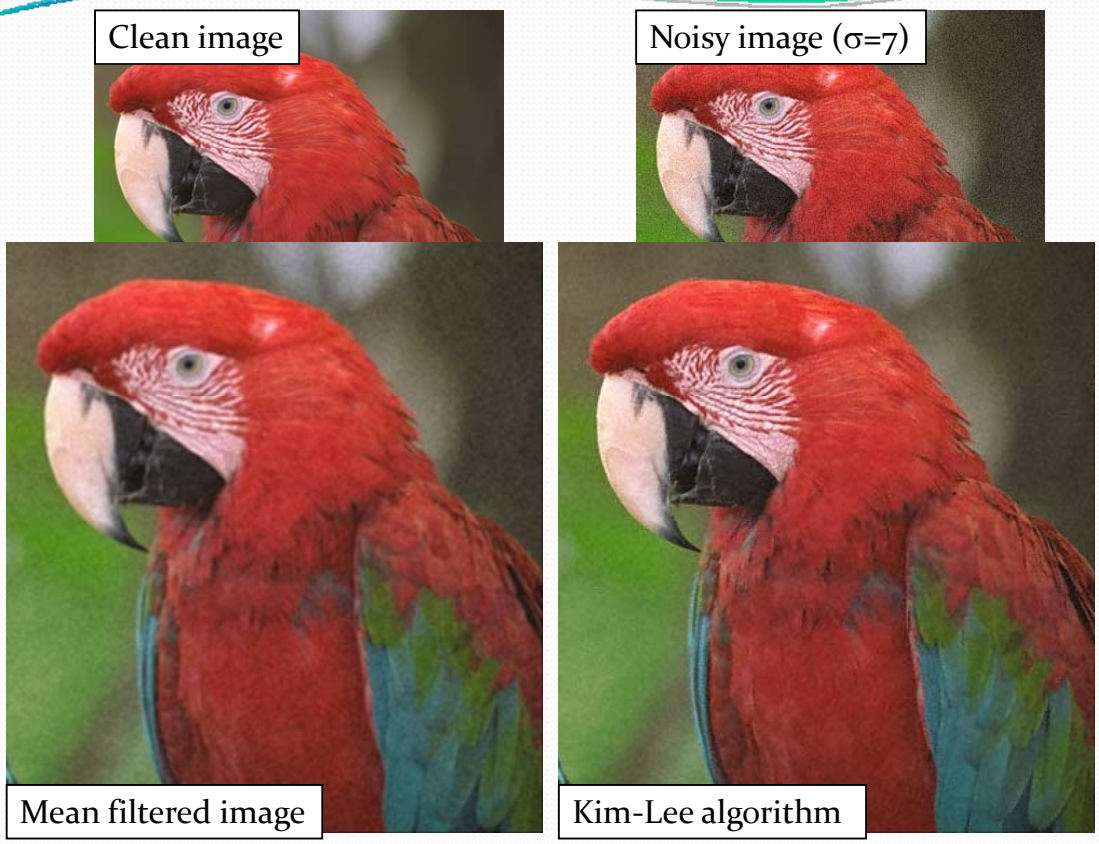
Mask δ_f

Kim-Lee noise removal

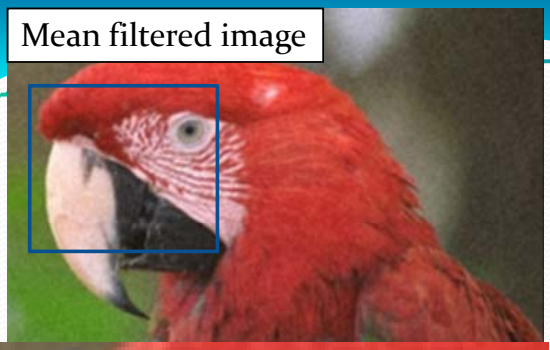
- Proposed detection algorithm can be used in noise reduction by combining it with a simple average filter.
- Let $\{y_{ij}^{avg}\}$ be an average filter output for a given input image $\{x_{ij}\}$. Then the feature and noise adaptive average filter can be chosen as:

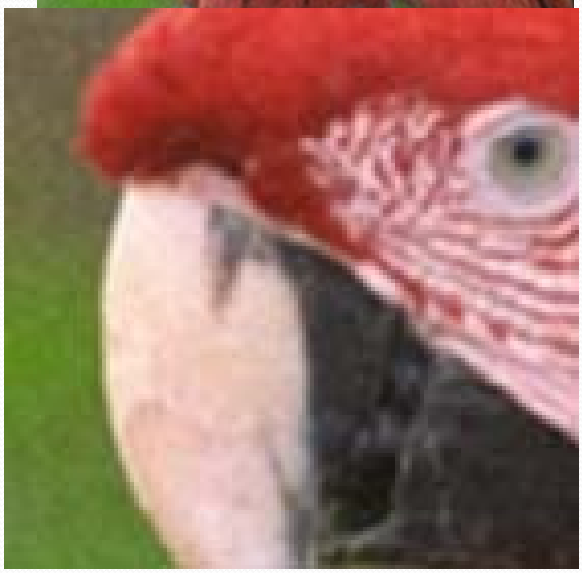
$$y_{ij}^{adapt} = \delta_f \cdot x_{ij} + (1-\delta_f) \cdot y_{ij}^{avg}$$

- Note that $y_{ij}^{adapt} = x_{ij}$ *when* δ_f*=1 (feature) and* $y_{ij}^{adapt} = y_{ij}^{avg}$ when δ_f*=0 (noise).* Depending on the detection of the feature and noise, the output is adaptively adjusted so that the feature can be preserved whereas noise can be smoothed. It is relevant that the adaptation given here does not require noise variance to be known, which makes it different from other conventional adaptive noise reduction algorithms.



Mean filtered image





Kim-Lee algorithm