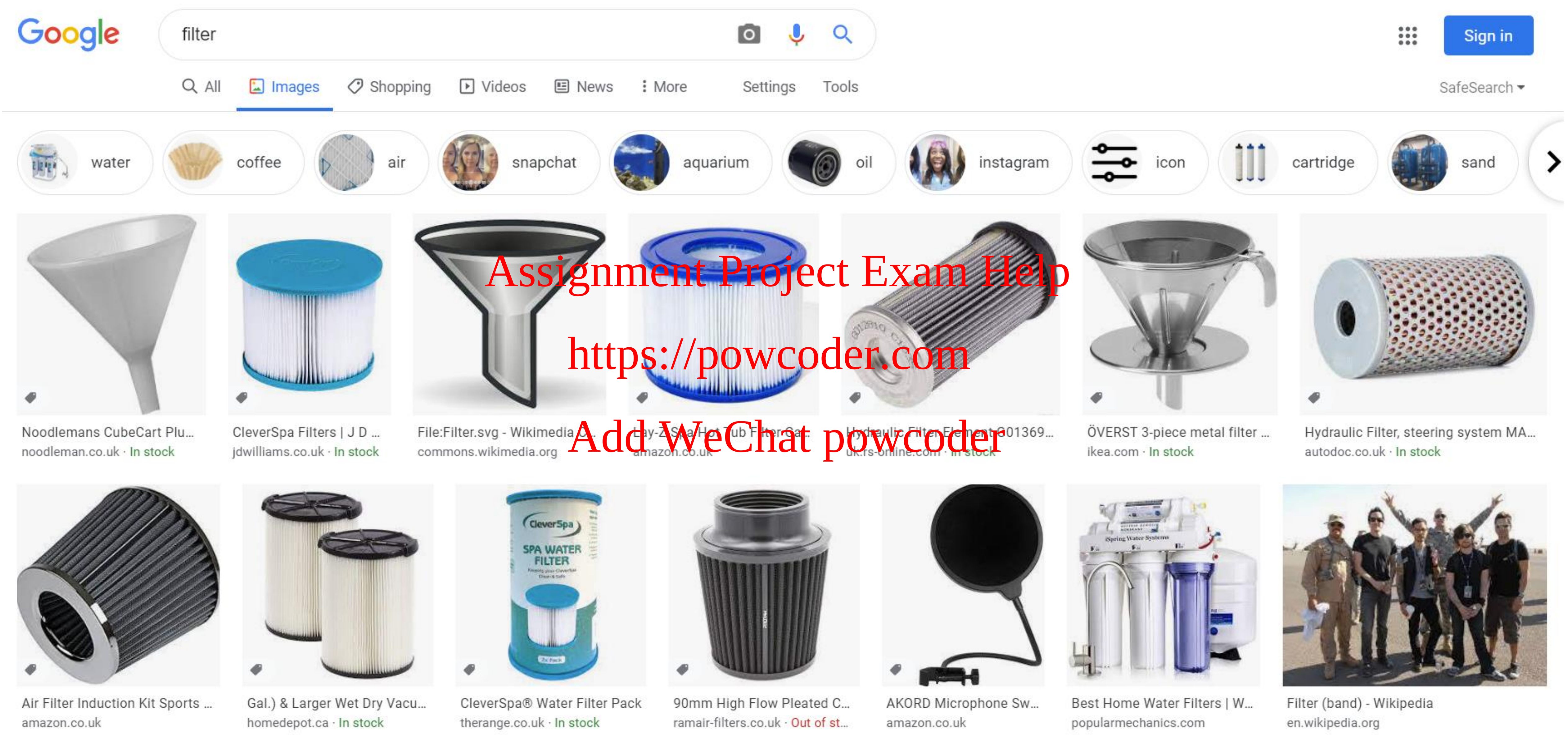
EBU7240

Computer Vision

- Restoration: spatial filtering -

Semester 1, 2021

Changjae Oh


Google
filter
Sign in
All
Images
Shopping
Videos
News
More
Settings
Tools
SafeSearch
water
coffee
air
snapchat
aquarium
oil
instagram
icon
cartridge
sand
Noodlemans CubeCart Plu...
noodleman.co.uk · In stock
CleverSpa Filters | J D ...
jdwilliams.co.uk · In stock
File:Filter.svg - Wikimedia C...
commons.wikimedia.org
amazon.co.uk
ÖVERST 3-piece metal filter ...
ikea.com · In stock
Hydraulic Filter, steering system MA...
autodoc.co.uk · In stock
Air Filter Induction Kit Sports ...
amazon.co.uk
Gal.) & Larger Wet Dry Vacu...
homedepot.ca · In stock
CleverSpa® Water Filter Pack
therange.co.uk · In stock
90mm High Flow Pleated C...
ramair-filters.co.uk · Out of st...
AKORD Microphone Sw...
amazon.co.uk
Best Home Water Filters | W...
popularmechanics.com
Filter (band) - Wikipedia
en.wikipedia.org

Content

- **Neighborhood in an image**
- **Convolution: review**
  – From 'Signal Processing' Lecture
- **Spatial Filtering**
  – Low-pass (or high-pass) filter
  – Gaussian (or Laplacian) filter
  – Mean, median, or mode filter
  – Advanced filters: Bilateral filter, Non-local means filter

Neighborhood in an image

- Spatial filtering is performed for each pixel (i, j) using neighborhoods of (i, j).
  - $I(i, j)$: input image $O(i, j)$: output image
  - $w(s, t)$: filtering kernel (a.k.a. mask)

$$O(i, j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) I(i + s, j + t)$$

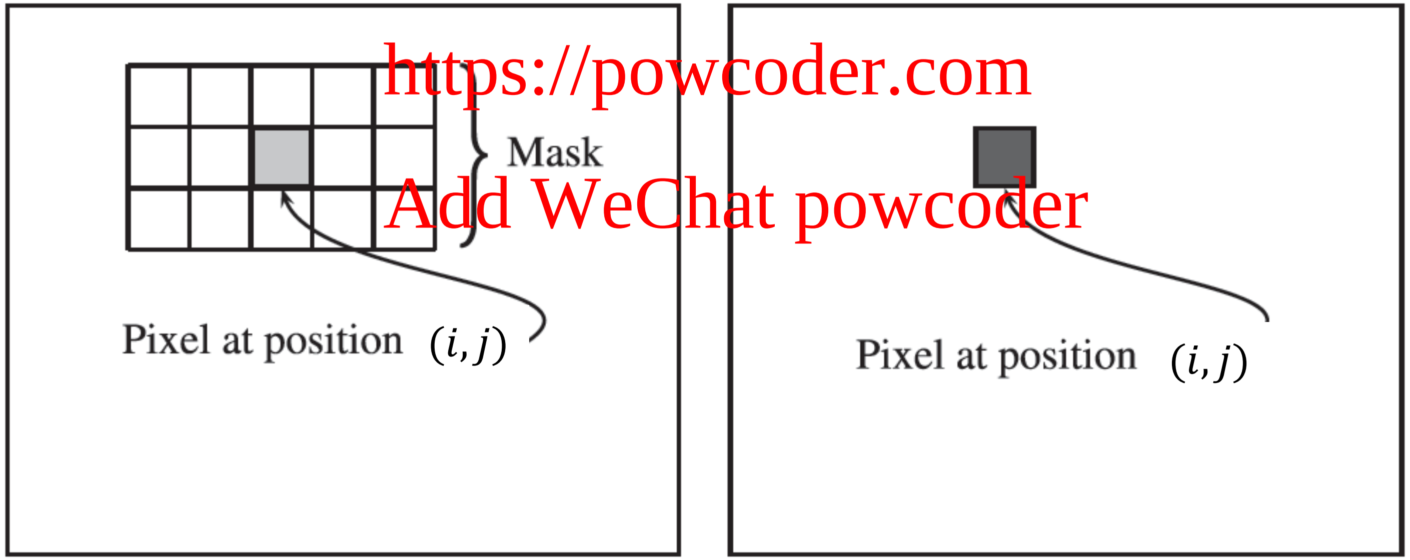
- The filtering operation is defined according to what kind of $w(s, t)$ is used.

- The filter serves as an essential building block for many applications.
  - Blurring, sharpening, image restoration, and so on

Spatial Filtering: Filter Kernel (Mask)

- **Filter kernel** should be defined accordingly, depending on applications.

$$O(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(i+s, j+t)$$

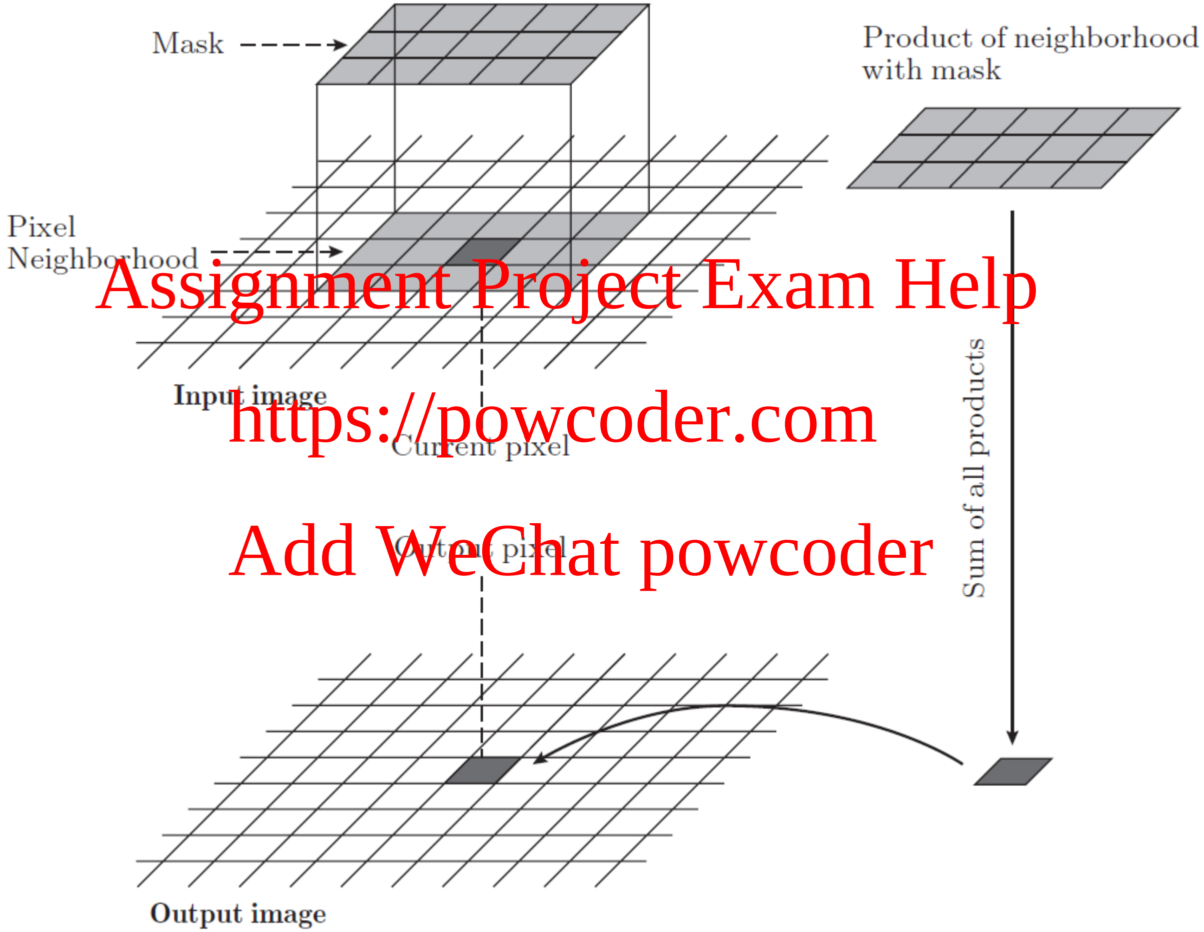


(a) (b)

Filtering = SUM (Mask .× Neighborhood)

$$O(i,j) = \sum \left(\begin{bmatrix} w(-1,-2) & w(-1,-1) & w(-1,0) & w(-1,1) & w(-1,2) \\ w(0,-2) & w(0,-1) & w(0,0) & w(0,1) & w(0,2) \\ w(1,-2) & w(1,-1) & w(1,0) & w(1,1) & w(1,2) \end{bmatrix} .\times \begin{bmatrix} I(i-1,j-2) & I(i-1,j-1) & I(i-1,j) & I(i-1,j+1) & I(i-1,j+2) \\ I(i,j-2) & I(i,j-1) & I(i,j) & I(i,j+1) & I(i,j+2) \\ I(i+1,j-2) & I(i+1,j-1) & I(i+1,j) & I(i+1,j+1) & I(i+1,j+2) \end{bmatrix} \right)$$

Filtering = SUM (Mask .× Neighborhood)



Convolution: From the perspective of signal processing

- **Filtering = Convolution!**
  - When the filter kernel is symmetric

- Let's start with 1-D convolution (from signal processing)
  - Convolution integral

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

For continuous signal

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m]$$

For discrete signal

$$y[n] = \cdots + x[-1]\mathrm{h}[n+1] + x[0]h[n] + x[1]h[n-1] + \cdots$$

Convolution: From the perspective of signal processing

- **From the perspective of signal processing**
  - An output signal $y[n]$ is obtained by passing an input signal $x[n]$ to a discrete system with a response function $h[n]$.

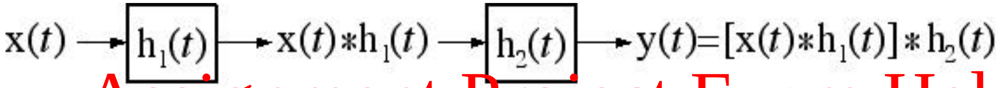
$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m]$$

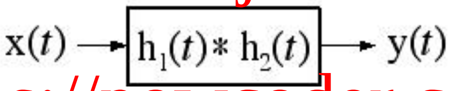
$$x[n] \longrightarrow \boxed{h[n]} \longrightarrow y[n]$$

Note **convolution** in spatial domain = **multiplication** in frequency domain

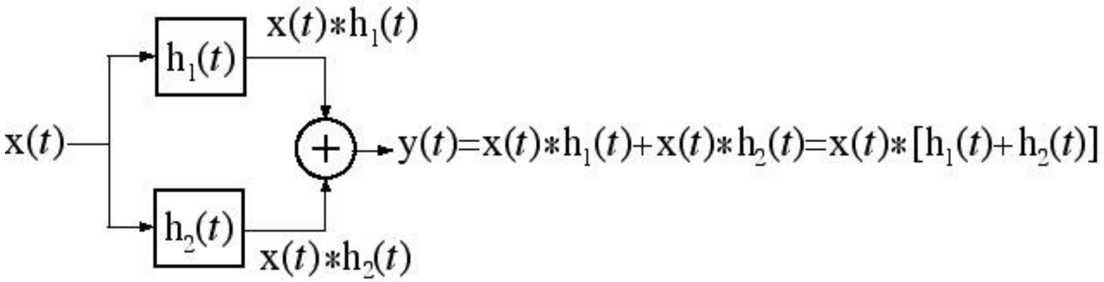
Convolution Properties

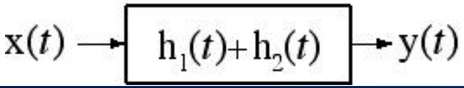
- **Associative property**
  - Impulse response of a cascade connection
    = Convolution of the individual impulse responses





- **Distributive property**
  - Impulse response of a parallel connection of LTI systems
    = Sum of the individual impulse responses.



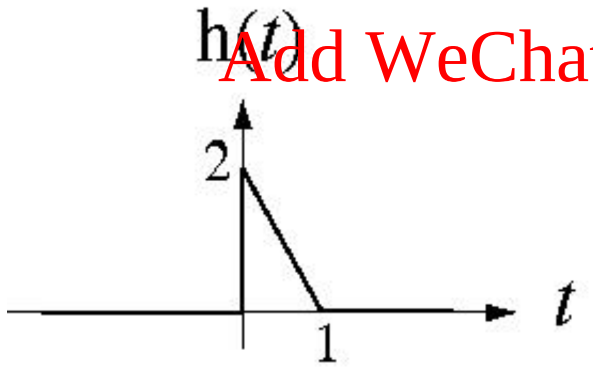


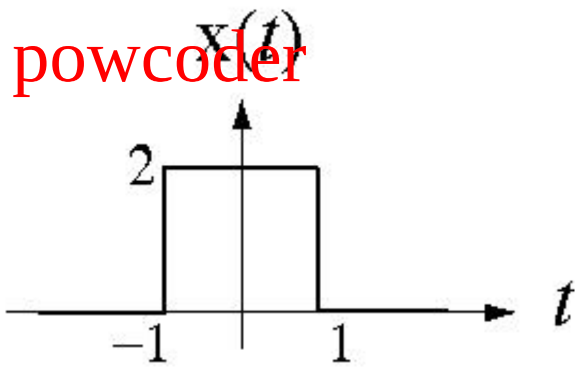
Example

- **The convolution on the continuous domain**

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

For illustration, let an input signal *x(t)* and the impulse response *h(t)* be the two functions below.





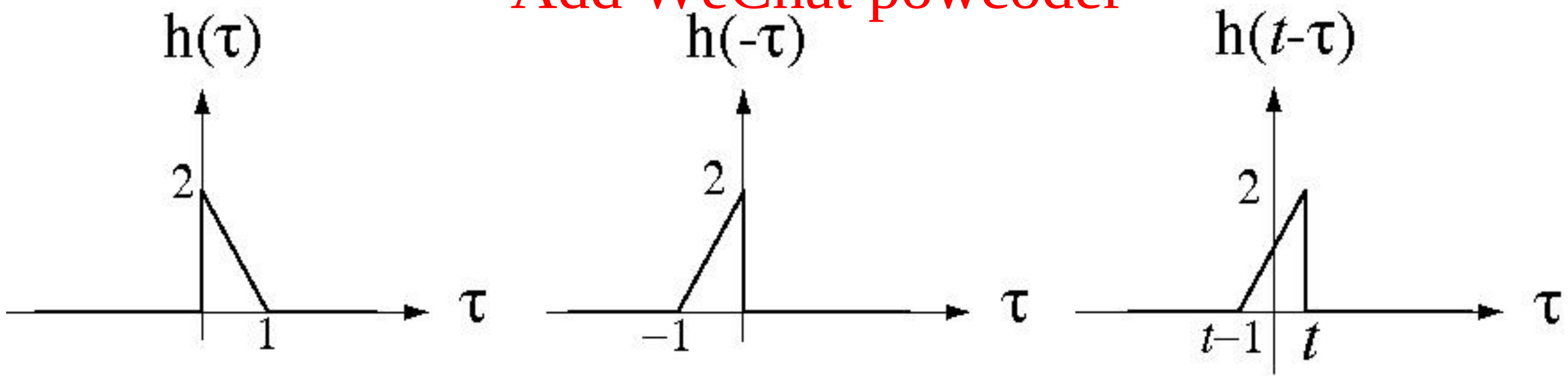
Example

- **The convolution on the continuous domain**

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

  – The functional transformation from $h(t)$ to $h(t-\tau)$

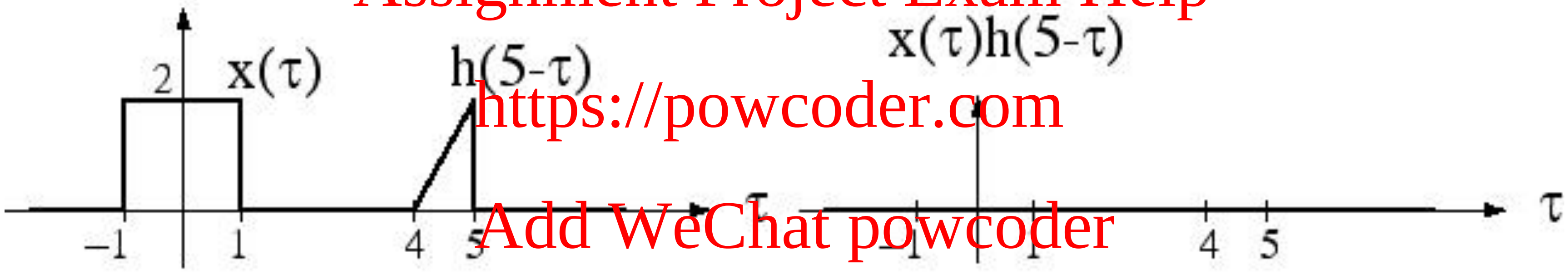
$$\mathrm{h}(\tau) \xrightarrow{\tau \to -\tau} \mathrm{h}(-\tau) \xrightarrow{\tau \to \tau - t} \mathrm{h}(-(\tau - t)) = \mathrm{h}(t-\tau)$$



Example

- **The convolution on the continuous domain**

The convolution value at t:
The area under the product of $x(t)$ and $h(t-\tau)$.
For example, let $t=5$.



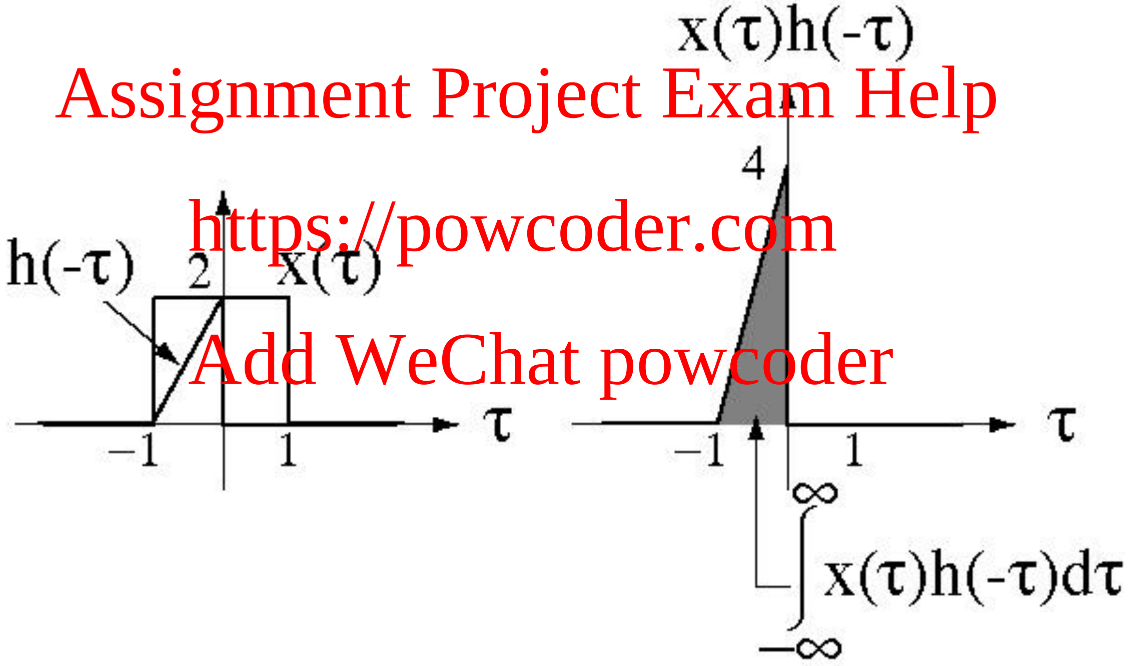
For $t=5$, the area under the product is zero.

$$y(5)=x(5)*h(5)$$
$$=\int_{-\infty}^{\infty} x(\tau)h(5-\tau)d\tau=0$$

Example

- **The convolution on the continuous domain**

When $t = 0$,

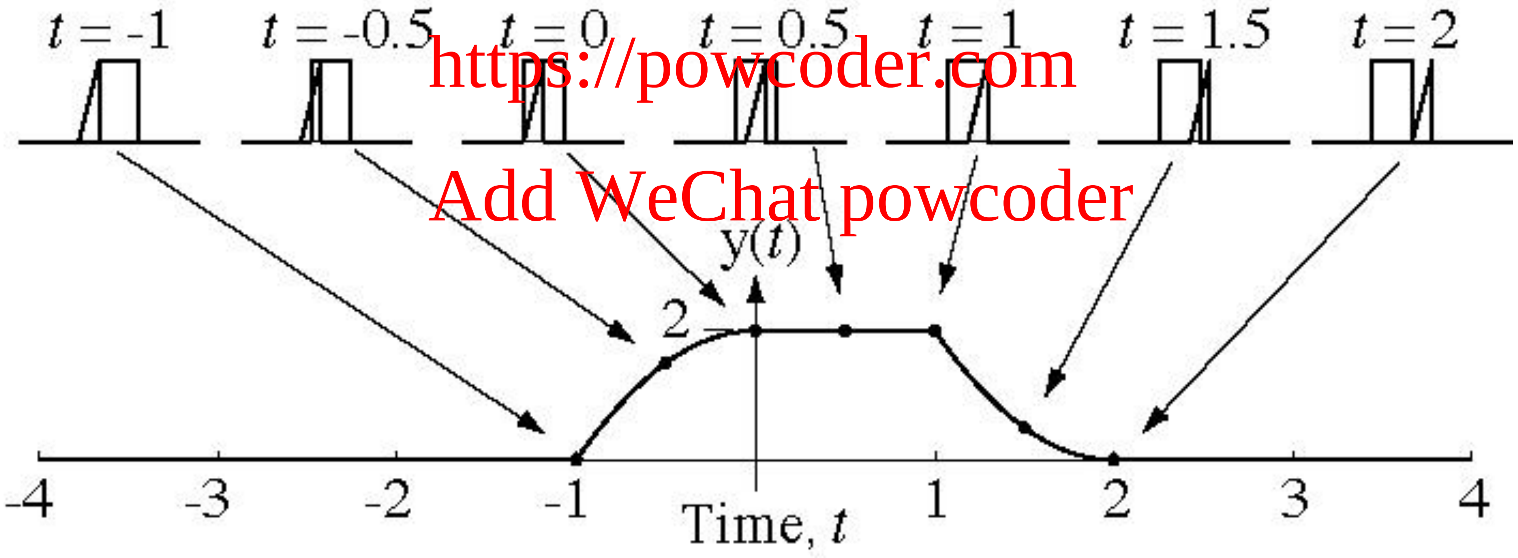


$$y(0) = x(0) * h(0) = \int_{-\infty}^{\infty} x(\tau)h(-\tau)d\tau = 2$$

Example

- **The convolution on the continuous domain**

The process of convolving to find $y(t)$ is illustrated below.

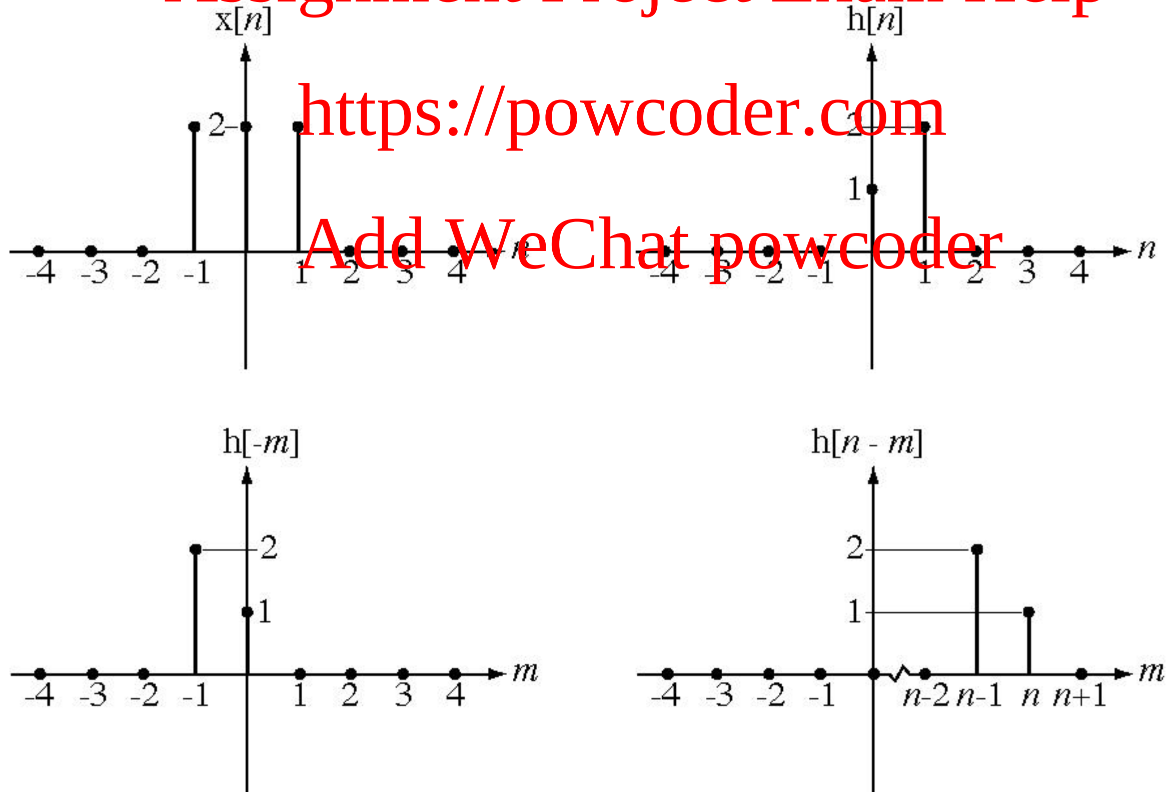


Example

- **The convolution on the discrete domain**

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m]$$

What is $y[n]$?



Note) x: M points, h: N points
→ y: $M + N - 1$ points

Convolution vs. Filtering

1D convolution

$$y[i] = x[i] * h[i]$$
$$= \sum_{s=-\infty}^{\infty} h[s]\,x[i-s]$$

Note) This formulation is different from previous slides, but the result is identical. Namely, $x * h = h * x$.

1D filtering

$$O[i] = \sum_{s=-a}^{a} w[s]\,I[i+s]$$

Remember 2D filtering

$$O(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\,I(i+s, j+t)$$

Suppose $h[i]$ exists for $-a \le i \le a$ and is symmetric, i.e., $h[-i] = h[i]$.
Then, 1D convolution = 1D filtering!
This also applies to N-D convolution and N-D filtering ($N \ge 2$).

2D convolution - example

Original

*

0	0	0
0	1	0
0	0	0

= ?

Source from D. Lowe

2D convolution - example

Original

*

•0	•0	•0
•0	•1	•0
•0	•0	•0

=

Filtered
(no change)

2D convolution - example

Original

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \ ?$$

Source from D. Lowe

2D convolution - example

Original

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} =$$

Shifted right
By 1 pixel

Source from D. Lowe

2D convolution - example

Original

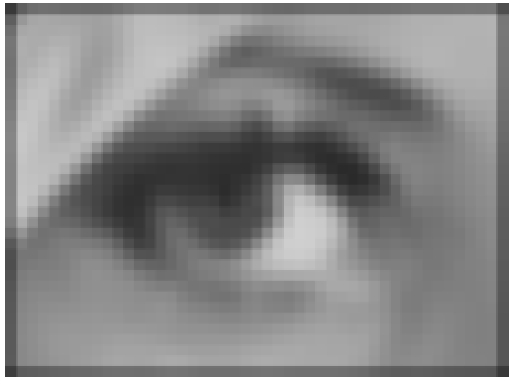
$$* \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Source from D. Lowe

2D convolution - example

Original

$$* \ \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$



Blur (with a box filter)

Source from D. Lowe

2D convolution - example

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

(Note that filter sums to 1)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}}_{\text{“details of the image”}}$$

Source from D. Lowe

2D convolution - example

- Removing Blurred region means:

−

=

- Adding it back means:

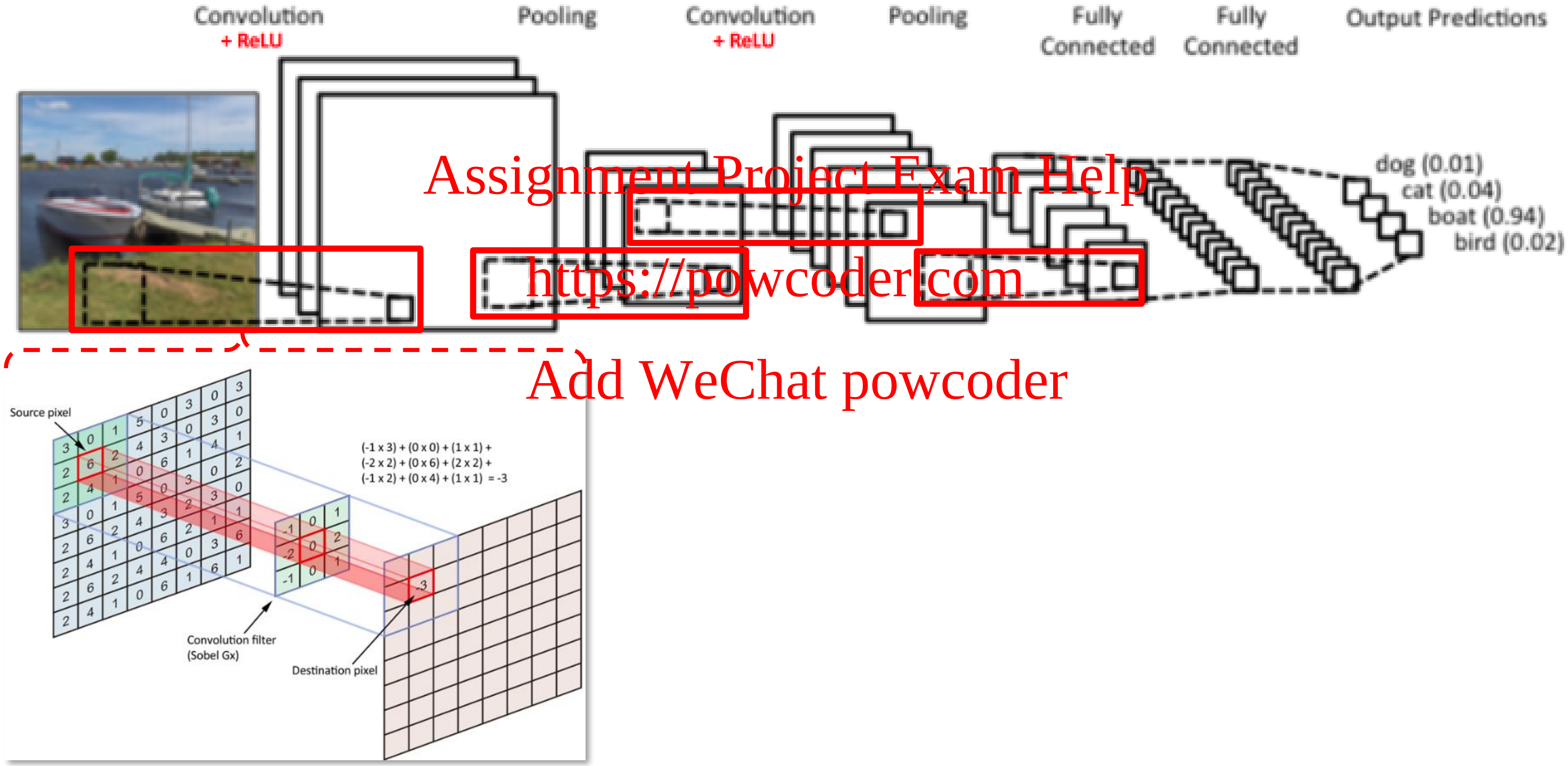
+ a

=

Source from L. Fei-Fei

2D convolution - example

- Deep learning: Convolutional Neural Network



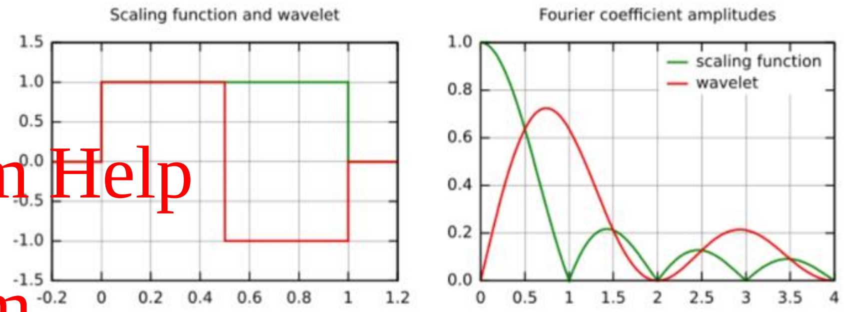
Credit: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

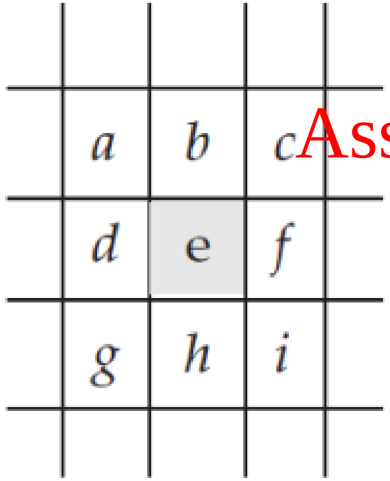
2D Image Filtering: Uniform Mean Filter

- **Uniform mean filter**
  - The simplest **low-pass** filter

EBU6018- remember?

Haar functions in frequency domain





$$\rightarrow \frac{1}{9}(a+b+c+d+e+$$

Filter kernel

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

2D Image Filtering: Uniform Mean Filter

- **How to process pixels at image boundary?**
  - 1) Mirroring at an image boundary

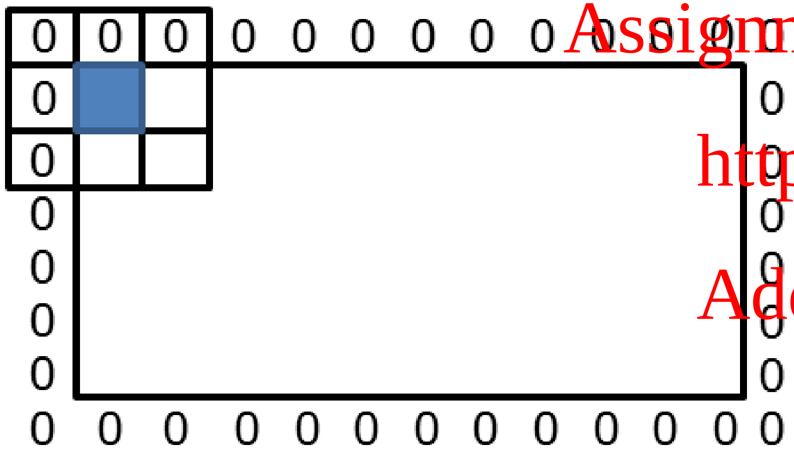
2	2	3	4	5	4	3	2	1	2	3	4	4
2	2	3	4	5	4	3	2	1	2	3	4	4
3	3										3	3
4	4										4	4
5	5										5	5
6	6										6	6
7	7	6	5	4	3	2	1	2	3	4	5	5
7	7	6	5	4	3	2	1	2	3	4	5	5

$$O(i,j) = \sum_{s=-1}^{1}\sum_{t=-1}^{1} w(s,t)\, I'(i+1+s, j+1+t)$$

I': mirror image

2D Image Filtering: Uniform Mean Filter

- **How to process pixels at image boundary?**
  - 2) Zero padding at an image boundary
  - Common way in convolutional neural networks

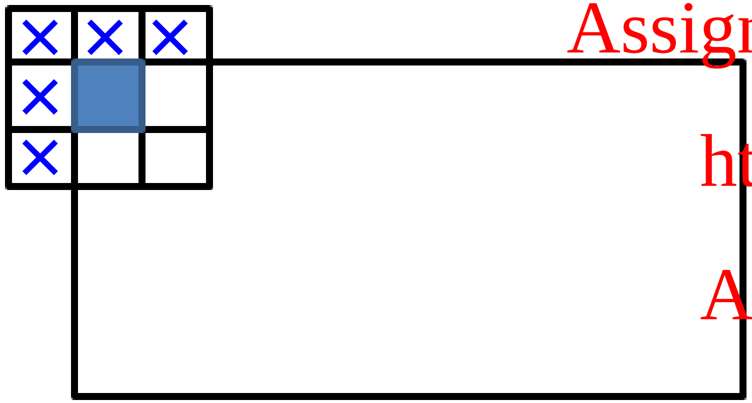


$$O(i,j) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t) I'(i+1+s, j+1+t)$$

I': zero-padded image

2D Image Filtering: Uniform Mean Filter

- **How to process pixels at image boundary?**
  - 3) Adjusting filter kernel



$$O(i,j) = \frac{\sum_{s=-1}^{1}\sum_{t=-1}^{1} w'(s,t) I(i+s, j+t)}{\sum_{s=-1}^{1}\sum_{t=-1}^{1} w'(s,t)}$$

$$w'(s,t) = (0 \le i+s \le H-1 \;\&\; 0 \le j+t \le W-1 \; ? \; w(s,t) : 0)$$

2D Image Filtering: Uniform Mean Filter

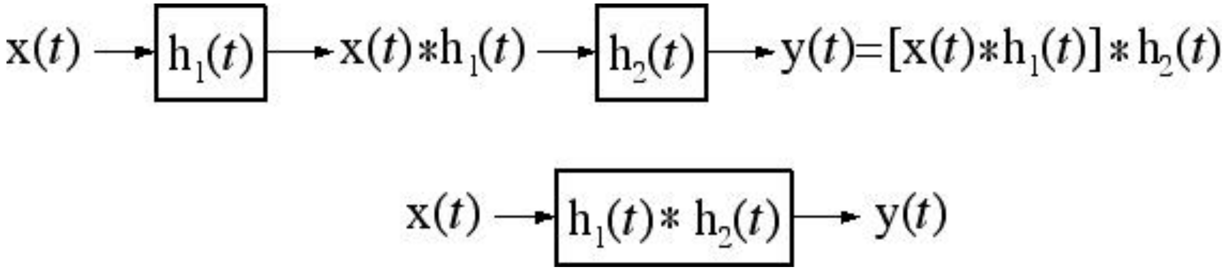
(a) original image, (b) 3×3 filter with zero padding: $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, (c) 9×9 filter with zero padding: $\frac{1}{81}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$,(d) 25×25 filter with zero padding: $\frac{1}{625}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$, (e) 25×25 filter with mirroring: $\frac{1}{625}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$

2D Image Filtering: Uniform Mean Filter

- **Uniform mean filtering is separable.**
  - Separable filtering is VERY important in terms of runtime.

$$w(s,t) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = w_s(s,0) * w_s(0,t)$$

Cascade Connection



2D Image Filtering: Gaussian Filter

- **Gaussian distribution**
  - One of the most commonly used parametric models
  - $\boldsymbol{Fourier\ transform}\,(Gaussian\ func.) = \ Gaussian\ func.$
  - Rotationally symmetric
  - Separable filter
  - $Gaussian\ *\ Gaussian\ =\ Gaussian$

1-D

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

2-D

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-\frac{(x-\mu_x)^2}{2\sigma_x{}^2}-\frac{(y-\mu_y)^2}{2\sigma_y{}^2}\right)$$

N-D

$$f(\boldsymbol{m}) = \frac{1}{|2\pi|^{N/2}|\Sigma|^{1/2}}\exp(-(\boldsymbol{m}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{m}-\boldsymbol{\mu}))$$

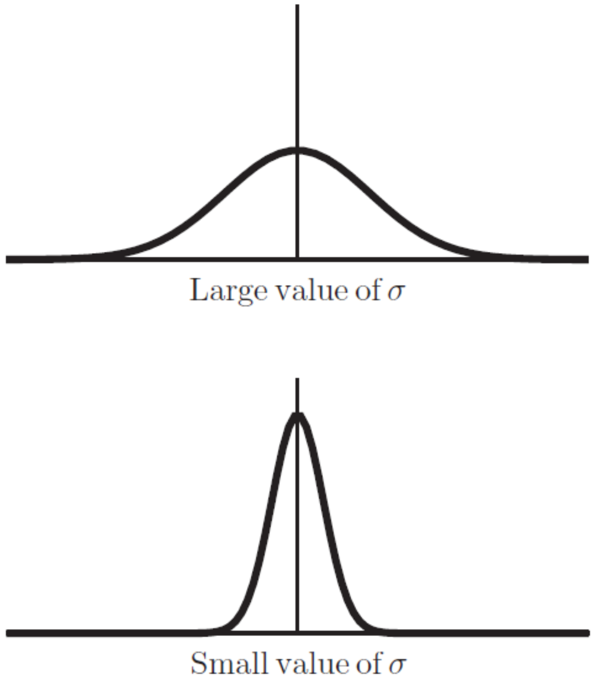
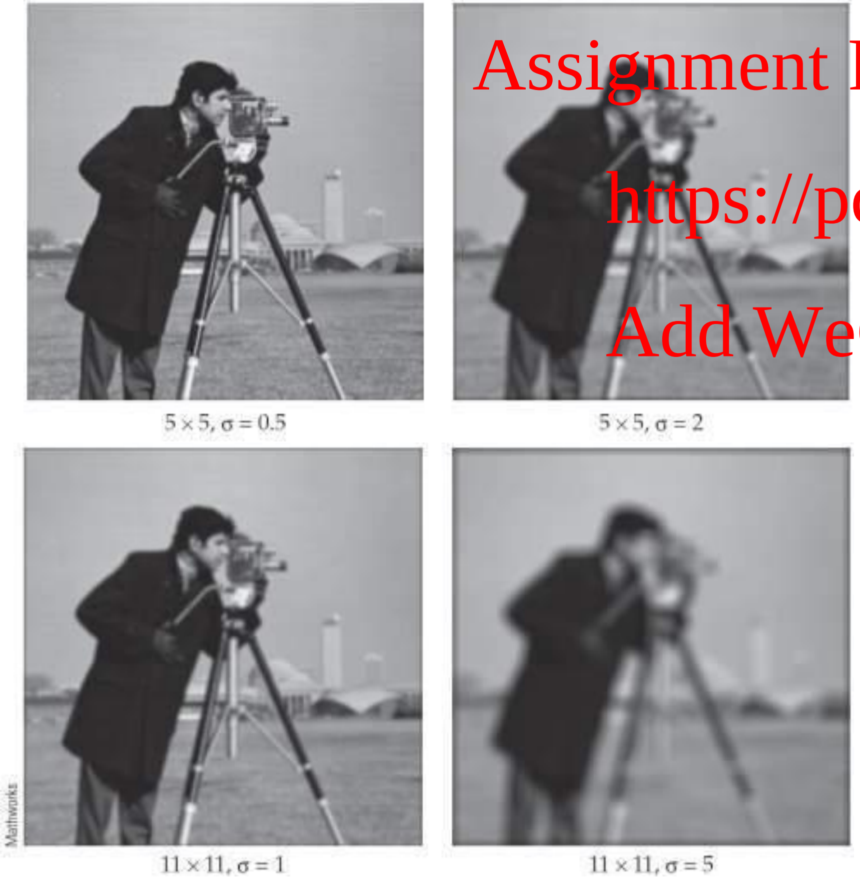


FIGURE 5.7 *One-dimensional Gaussians.*

Q: What if σ approaches infinite?

2D Image Filtering: Gaussian Filter

- **Gaussian filter's advantage**
  – It considers spatial distances within neighborhoods
  – Blurred results looks more natural compared to mean filter.



<u>Practical Implementation</u>

$$O(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(i+s, j+t)$$

Zero mean Gaussian filter

$$w(s,t) = \frac{1}{\sum_{m=-a}^{a} \sum_{n=-b}^{b} \exp\left(-\frac{m^2}{2{\sigma_s}^2} - \frac{n^2}{2{\sigma_t}^2}\right)} \exp\left(-\frac{s^2}{2{\sigma_s}^2} - \frac{t^2}{2{\sigma_t}^2}\right)$$

In results, $\mu_s = \mu_t = 0$ and $\sigma_s = \sigma_t = \sigma$

2D Image Filtering: Gaussian Filter

- **Gaussian filter's advantage**
  - Separable

$$\frac{1}{2\pi\sigma_s\sigma_t}\exp\left(-\frac{s^2}{2{\sigma_s}^2}-\frac{t^2}{2{\sigma_t}^2}\right)=\frac{1}{\sqrt{2\pi}\sigma_s}\exp\left(-\frac{s^2}{2{\sigma_s}^2}\right)\frac{1}{\sqrt{2\pi}\sigma_t}\exp\left(-\frac{t^2}{2{\sigma_t}^2}\right)$$

<u>**Practical implementation**</u>

$$O(i,j)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)I(i+s,j+t)$$

$$w(s,t)=\frac{1}{\sum_{m=-a}^{a}\sum_{n=-b}^{b}\exp\left(-\frac{m^2}{2{\sigma_s}^2}-\frac{n^2}{2{\sigma_t}^2}\right)}\exp\left(-\frac{s^2}{2{\sigma_s}^2}-\frac{t^2}{2{\sigma_t}^2}\right)$$

$$\Updownarrow$$

$$O(i,j)=\sum_{s=-a}^{a}w_s(s)\sum_{t=-b}^{b}w_t(t)\,I(i+s,j+t)$$

$$w_s(s)=\frac{1}{\sum_{m=-a}^{a}\exp\left(-\frac{m^2}{2{\sigma_s}^2}\right)}\exp\left(-\frac{s^2}{2{\sigma_s}^2}\right)$$

$$w_t(t)=\frac{1}{\sum_{n=-b}^{b}\exp\left(-\frac{n^2}{2{\sigma_t}^2}\right)}\exp\left(-\frac{t^2}{2{\sigma_t}^2}\right)$$

Frequency: Low- and High-pass Filters

- **Frequency in an image**
  - How often do intensity values vary in neighbourhoods?
- **Low-pass filter:** uniform average filter, Gaussian filter
- **High-pass filter:** Sobel filter, Laplacian filter

(this estimates intensity change.)

Example of Low-pass filter: $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$

Example of High-pass filter: $\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$

2D Image Filtering: Sobel Filter

- Using the first order gradient

$$\nabla I = (I_x, I_y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$$

Practical implementation

$$\nabla I = (I_x, I_y) = (|S_x * I|, |S_y * I|)$$

For color image,
$M(x,y) = (M_R(x,y) + M_G(x,y) + M_B(x,y))/3$

Sobel filter output $M(x,y) = \sqrt{{I_x}^2 + {I_y}^2}$ or $|I_x| + |I_y|$

S_x

-1	-2	-1
0	0	0
1	2	1

S_y

-1	0	1
-2	0	2
-1	0	1

2D Image Filtering: Laplacian Filter

- **Using the second order gradient**

Laplacian Filtering output $G(x,y) = \left| \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \right|$

- 1st derivative of an image $I(x,y)$

$$\nabla I(x,y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

$$I_x(x,y) = \frac{\partial I}{\partial x} = I(x+1,y) - I(x,y) \qquad I_y(x,y) = \frac{\partial I}{\partial y} = I(x,y+1) - I(x,y)$$

- 2nd derivative of an image $I(x,y)$

$$\nabla^2 I(x,y) = \left(\frac{\partial^2 I}{\partial x^2}, \frac{\partial^2 I}{\partial y^2} \right)$$

$$\frac{\partial^2 I}{\partial x^2} = I_x(x,y) - I_x(x-1,y) = I(x+1,y) + I(x-1,y) - 2I(x,y)$$

$$\frac{\partial^2 I}{\partial y^2} = I_y(x,y) - I_y(x,y-1) = I(x,y+1) + I(x,y-1) - 2I(x,y)$$

2D Image Filtering: Laplacian Filter

- **Using the second order gradient**

Laplacian filter L

0	1	0
1	-4	1
0	1	0

or

1	1	1
1	-8	1
1	1	1

More general form

$\frac{\alpha}{1+\alpha}$	$\frac{1-\alpha}{1+\alpha}$	$\frac{\alpha}{1+\alpha}$
$\frac{1-\alpha}{1+\alpha}$	$\frac{-4}{1+\alpha}$	$\frac{1-\alpha}{1+\alpha}$
$\frac{\alpha}{1+\alpha}$	$\frac{1-\alpha}{1+\alpha}$	$\frac{\alpha}{1+\alpha}$

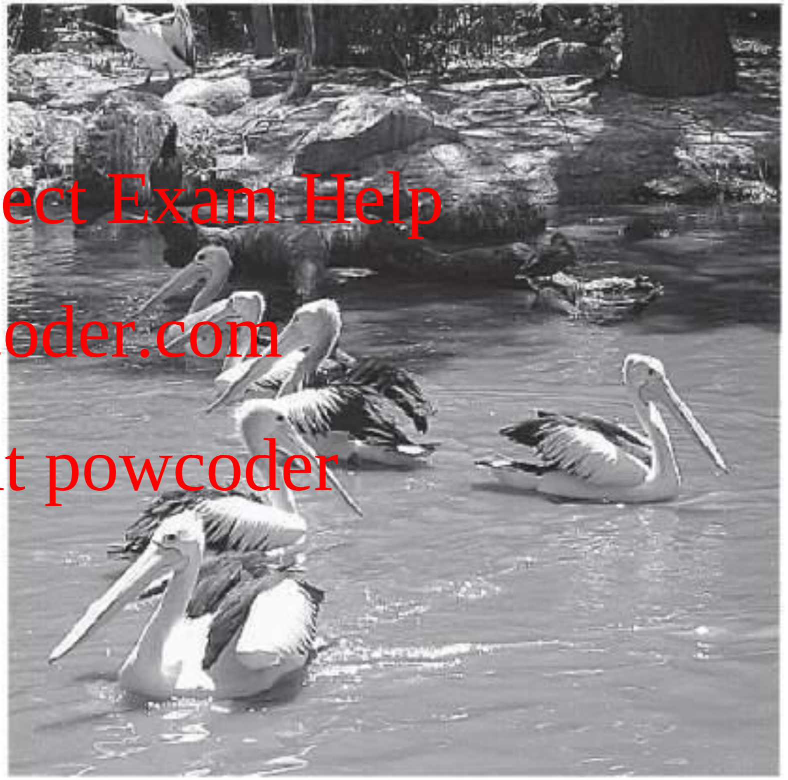
$$O = |L * I|$$

For color image,
$O(x,y) = \left(O_R(x,y) + O_G(x,y) + O_B(x,y)\right)/3$

Unsharp Masking

- Makes an image look sharper by boosting high-frequency components.

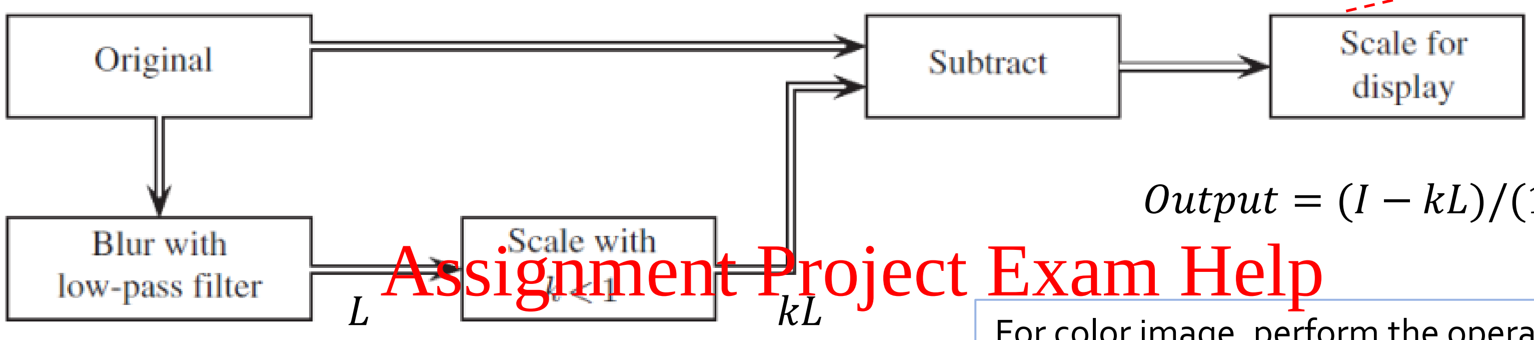
Original image



Unsharp mask result

Unsharp Masking

Scaling matters!



$$Output = (I - kL)/(1 - k)$$

For color image, perform the operation for RGB channels independently.

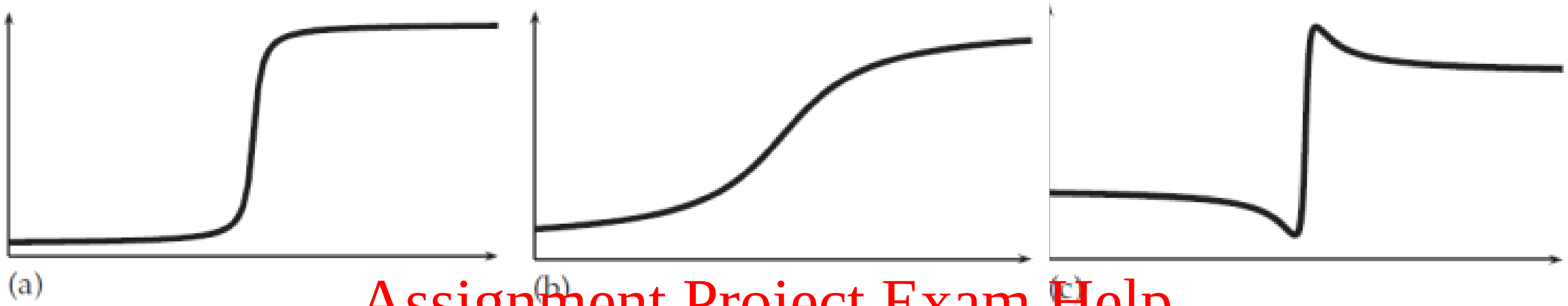
Matlab Code

```
x = double(imread('cameraman.tif'))/255;

f = fspecial('average');
xf = filter2(f,x);
figure, imshow(xf)

% k: parameter determining the amount of reducing low-frequency component
% 0<=k<=1
% As k increases, the output image looks sharper. But, it is recommended using k below 0.5;
k = 0.5;
fi = zeros(3);      fi(2,2)=1;
f2 = (fi - k*f)/(1-k);
xf2 = filter2(f2,x);
figure, imshow(xf2)
```

Unsharp Masking



$Output = (I - kL)/(1 - k)$

$$w = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - k \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Scaling

$$w = \frac{1}{1-k} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{k}{1-k} \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

2D Image Filtering: Nonlinear Filter

- **Linear vs. Nonlinear filters?**
  - Definition of linearity, and its advantages

- **Some instances of nonlinear filter**
  - Max, Min, Median filter


Original


Max filter


Min filter


Median filter

2D Image Filtering: Nonlinear Filter

- **Median Filter**
  1. Sort pixels within the window centered at reference pixel
  2. Select the median value as an output.

Q: Mean vs. Median?

Original

Max filter

Min filter

Median filter

Summary

- **Spatial Filtering**
  - Depends on how to define a filter kernel.
  - Linear vs. Nonlinear filter: Linear filter is **separable**!
  - Mean, median, maximum, minimum, low-pass, high-pass, Gaussian, Laplacian filter
  - Edge Sharpening: combination of low-pass and high-pass filters.

- **Convolution**
  - Same as filter
  - Commutativity, Cascade property, Parallel property

EBU7240

Computer Vision

- Restoration: noise and noise removal -

Semester 1, 2021

Changjae Oh

Contents

- **Image Degradation Model**
  - Image Noise
- **Noise removal**
  - Salt-and-Pepper Noise Removal
  - Gaussian Noise Removal
  - Periodic Noise Removal

Image Degradation Model

- **Image restoration**
  - Aims to reduce the image degradation
- **Types of image degradation**
  - Noise, our-of-focus blur, motion blur

Image Degradation Model

- When an input degraded image $g(x,y)$ is given, our goal is to estimate $f(x,y)$.

$$g(x,y) = h(x,y) * f(x,y) + n(x,y)$$

- $n(x,y)$: Additive noise
- $h(x,y)$: Blur kernel, which is the same as filtering mask.

- In frequency domain (using 2D DFT)?

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

$$\Rightarrow F(u,v) = \big(G(u,v) - N(u,v)\big)/H(u,v)$$

(x,y): 2D image coordinate
(u,v): 2D frequency coord.

Is that it? It is not such a simple problem.
1) $N(u,v)$ is not known.
2) What if $H(u,v) = 0$?

Image Noise

- **Noise**
  - Any kind of degradation in an image caused by external disturbance
  - To make the problem simple, we pre-assume the noise models.
    The most appropriate restoration may vary depending on the noise types.

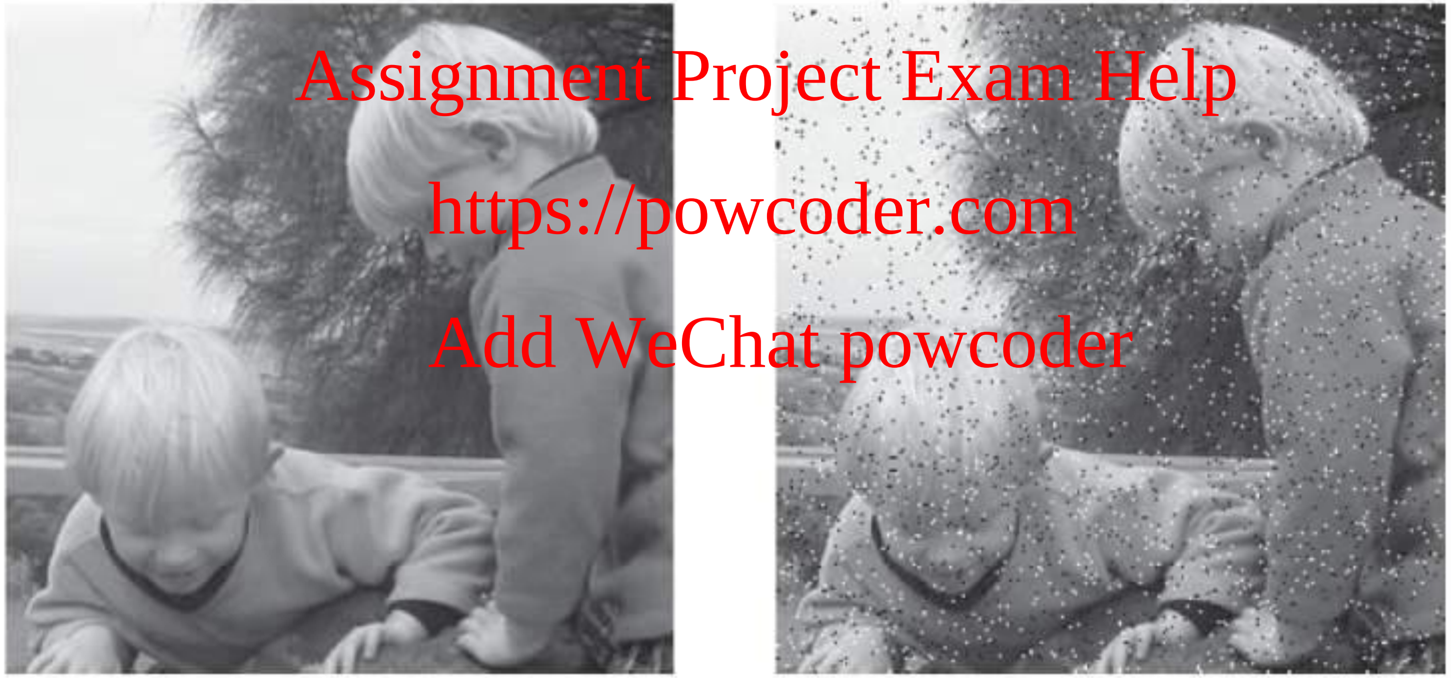
- **Noise types**
  - Periodic noise, Salt and Pepper noise, Gaussian noise, Speckle noise

- **In the beginning, let's think of the case using $h(x,y)$=1**

$$g(x, y) = f(x, y) + n(x, y)$$

Image Noise: Salt and Pepper Noise

- Sharp and sudden disturbances
- Image is randomly scattered as white (salt) or black (pepper) pixels.



(a) (b)

FIGURE 8.1 *Noise on an image. (a) Original image. (b) With added salt and pepper noise.*

Image Noise: Gaussian Noise

- **Additive White Gaussian Noise (AWGN)**
  - **Additive** noise: $I_G(x, y) = I(x, y) + N(x, y)$
  - **White** noise: randomly fluctuated and normally distributed
  - **Most approaches assume this type of noise.**

  - Usually, zero mean AWGN is assumed ($\mu = 0$).
    The probability density function (PDF) P of a Gaussian random variable z is

$$P(z = N) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$$

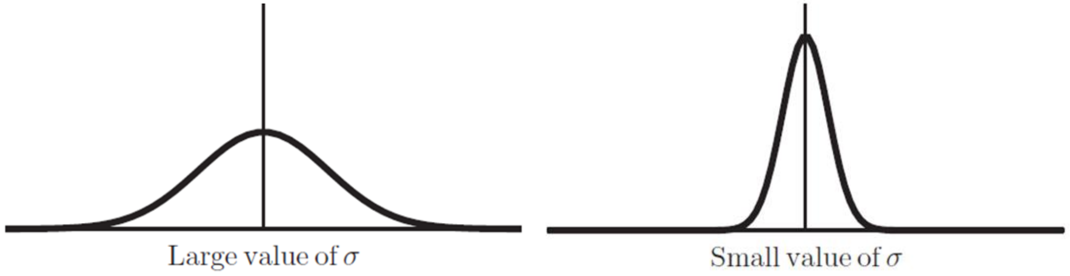
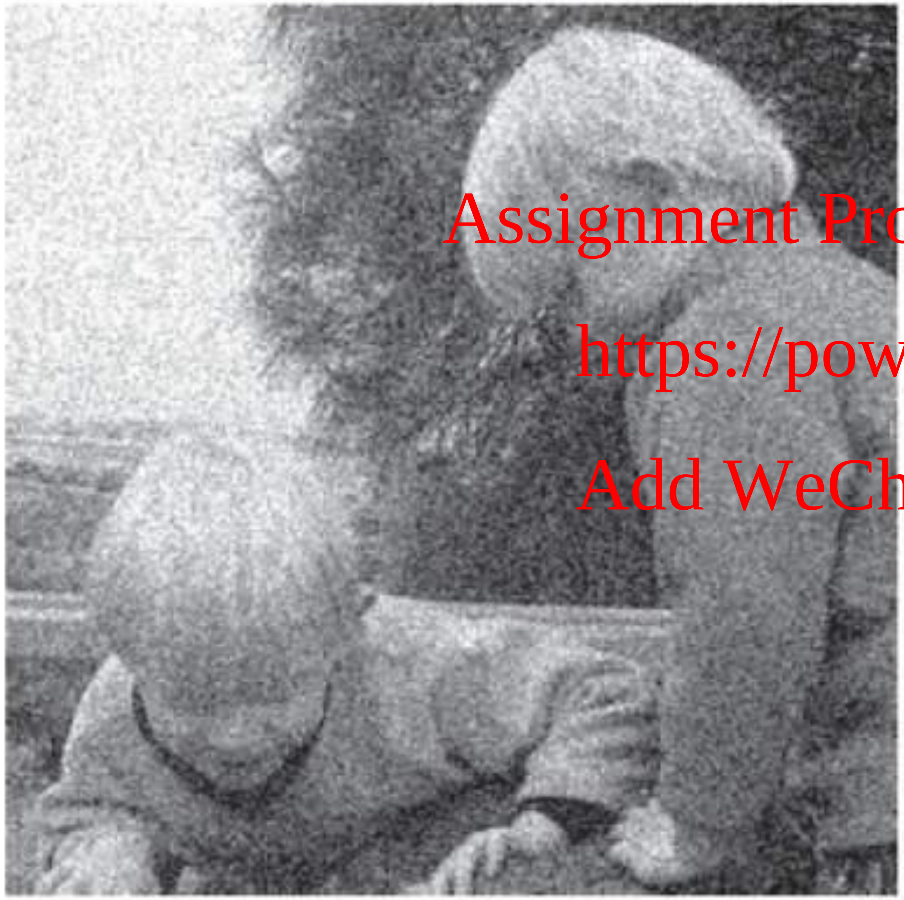


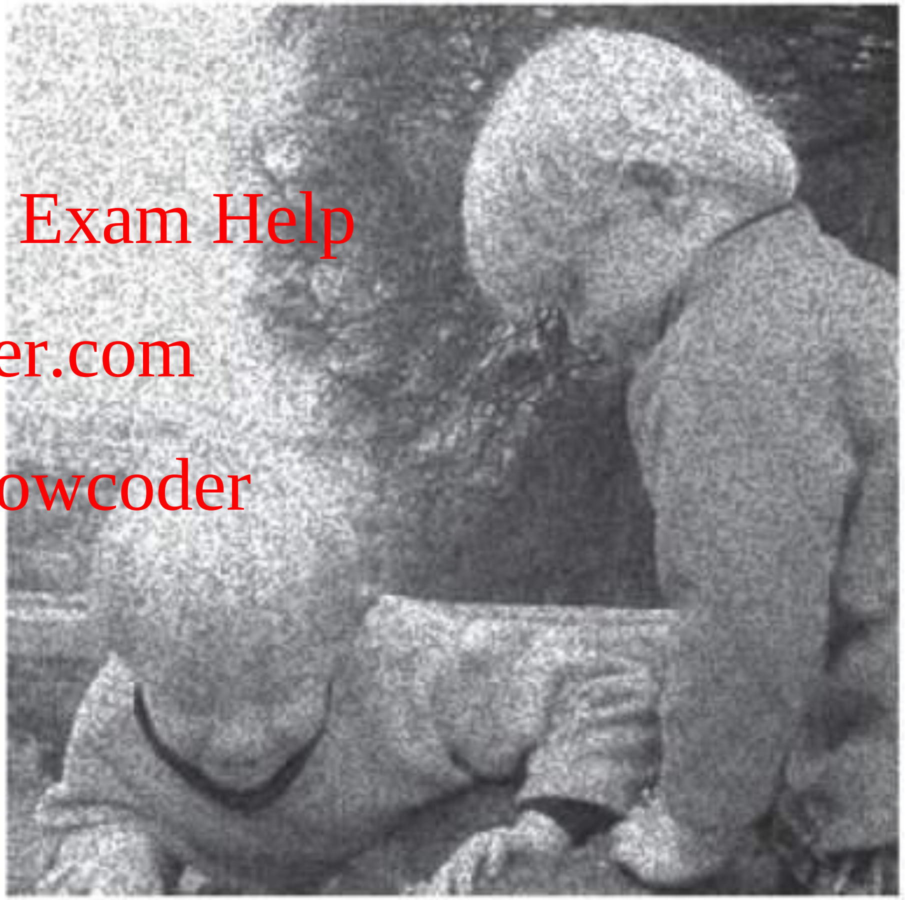
Image Noise: Speckle Noise

- **Multiplicative Noise**
  - $I_1(x,y) = I(x,y) + I(x,y)N(x,y)$
  - $N(x,y)$: zero mean uniform distributed function with σ
  - This noise is usually in the active radar, synthetic aperture radar (SAR), medical ultrasound and optical coherence tomography images.
  - The reduction of Speckle noise is MUCH more difficult due to the multiplication

Image Noise: Gaussian Noise vs. Speckle Noise



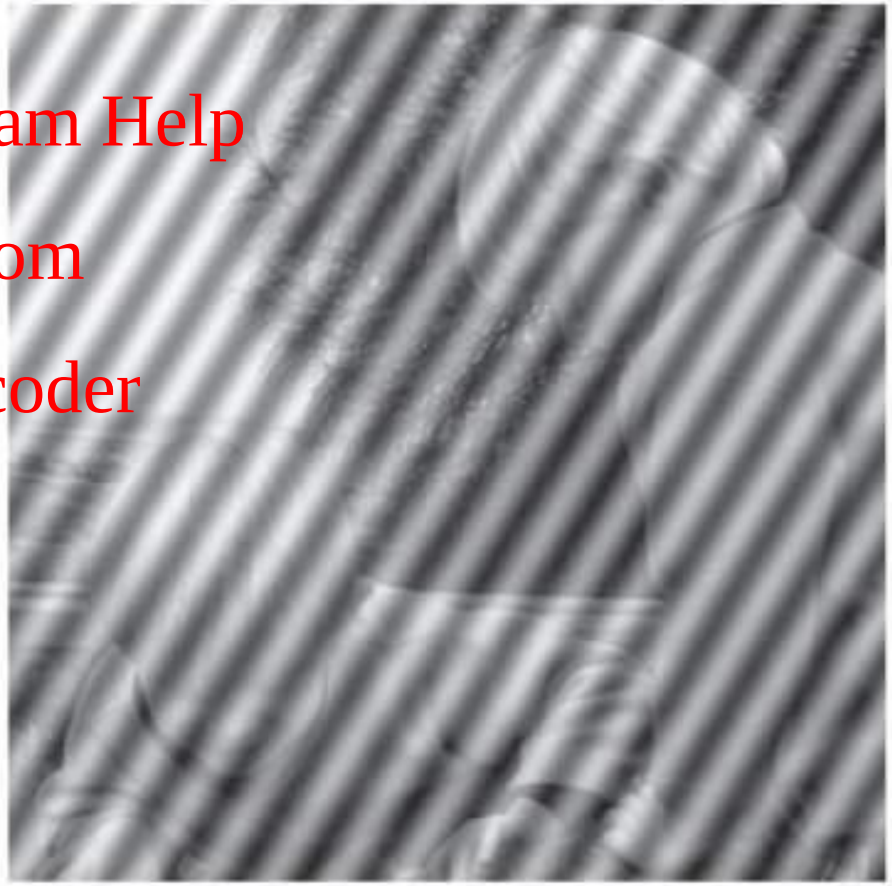
Gaussian Noise



Speckle Noise

Image Noise: Periodic Noise

- **Periodic fluctuation**
  - Electrical or electromechanical interference during image acquisition
  - Spatially dependent noise
  - It can be modeled as sinusoid waves



Contents

- **Image Degradation Model**
  - Image Noise
- **Noise removal**
  - Salt-and-Pepper Noise Removal
  - Gaussian Noise Removal
  - Periodic Noise Removal

Salt and Pepper Noise Removal

- **Low-Pass Filtering**
  - For instance, uniform averaging filter or Gaussian averaging filter
  - Not so effective

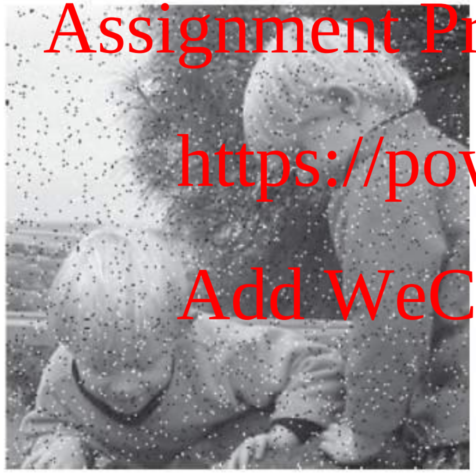
- **Median Filtering**
  - Median filter works well in the salt-and-pepper noise removal.
  - Rank-order Filtering is a general formulation of median filter.

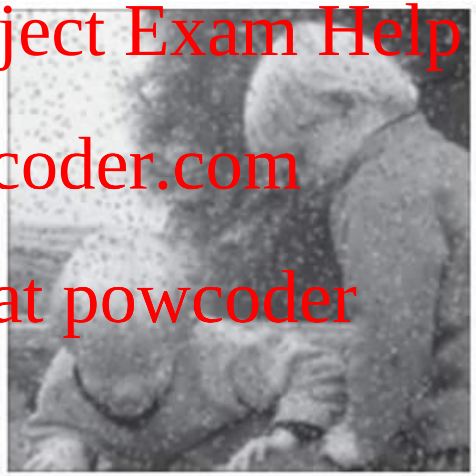
- **Outlier Rejection Method**

Salt and Pepper Noise Removal

- **Low-Pass Filtering**
  - NOT effective in removing the salt and pepper noise.


Original Image


10% salt and pepper noise
(salt: 5%, pepper: 5%)


3x3 average filter


7x7 average filter

Salt and Pepper Noise Removal: Median Filtering

- **Median filtering procedure**
  1. Sort pixels within the window centered at reference pixel
  2. Select the median value as an output.

$$\begin{array}{|c|c|c|}\hline 50 & 65 & 52 \\ \hline 63 & 255 & 58 \\ \hline 61 & 60 & 57 \\ \hline\end{array} \longrightarrow 50 \quad 52 \quad 57 \quad 58 \quad \boxed{60} \quad 61 \quad 63 \quad 65 \quad 255 \longrightarrow 60$$

  - Good performance in the salt-and-pepper noise removal

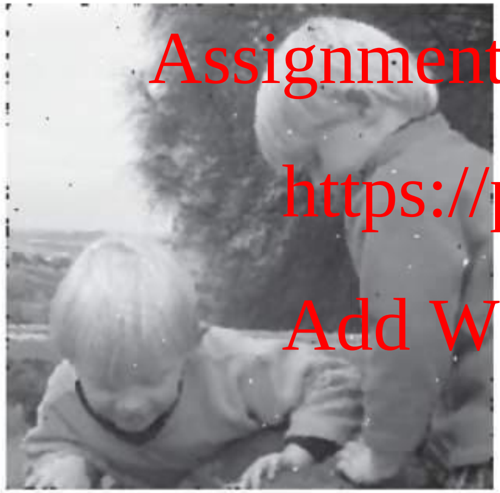
    Why?

3 × 3 median filter

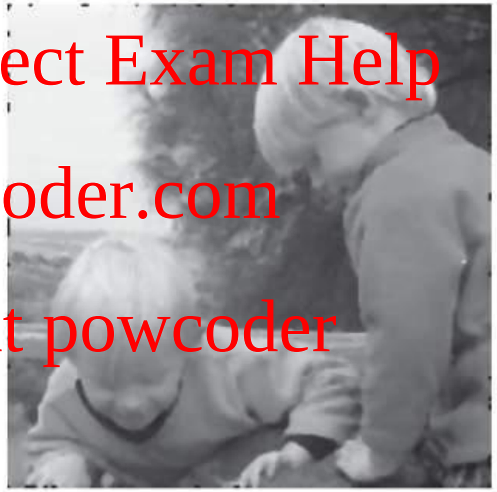
Salt and Pepper Noise Removal: Median Filtering

- More results using median filter

20% salt and pepper noise (salt: 10%, pepper: 10%)



Result using 3×3 median filter



Result applying 3×3 median filter twice

Result applying 5×5 median filter

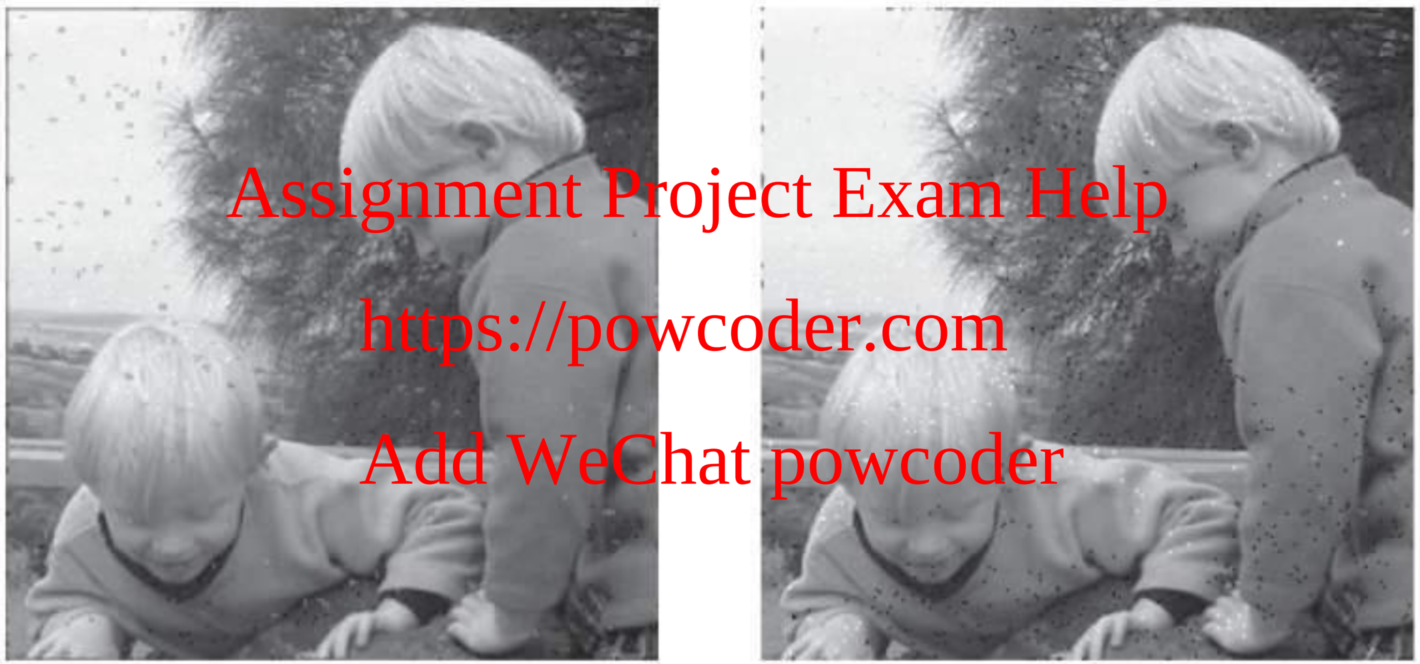
Salt and Pepper Noise Removal: Outlier Rejection Method

- **The brute force implementation of median filtering is very slow.**
  - Different method was proposed to remove the salt-and-pepper noise efficiently.

- **Key idea: Outlier detection → rejection**
  - Outliers usually tend to be different from neighboring pixels' intensities.

- **Outlier rejection method**
  1. Choose a threshold value D
  2. For a given pixel, compare its value p with the mean m of the values of its eight neighbors
  3. If $|p - m| > D$, then classify the pixel as noisy, otherwise not.
  4. If the pixel is noisy, replace its value with m.

Salt and Pepper Noise Removal: Outlier Rejection Method



(a) (b)

FIGURE 8.9 *Applying the outlier method to 10% salt and pepper noise. (a)* $D = 0.2$. *(b)* $D = 0.4$.

Gaussian Noise Removal

- **Why does an image average filter work well for Gaussian Noise removal?**

- **Additive White Gaussian Noise (AWGN) $N(x, y)$ is added as below.**
  - $I_G(x, y) = I(x, y) + N(x, y)$

1. **Suppose we have 100 noisy images I_G.**
   - $I_G{}^i(x, y) = I(x, y) + N^i(x, y) \quad i = 1, \ldots, 100$

2. **Average 100 noisy images**

10x10 kernel

$$\frac{1}{100}\sum_{i=1}^{100} I_G{}^i(x, y) = I(x, y) + \frac{1}{100}\sum_{i=1}^{100} N^i(x, y)$$

This will be close to 0, as AWGN N is a zero-mean Gaussian PDF.

Gaussian Noise Removal

- **Why does an image average filter work well for Gaussian Noise removal?**

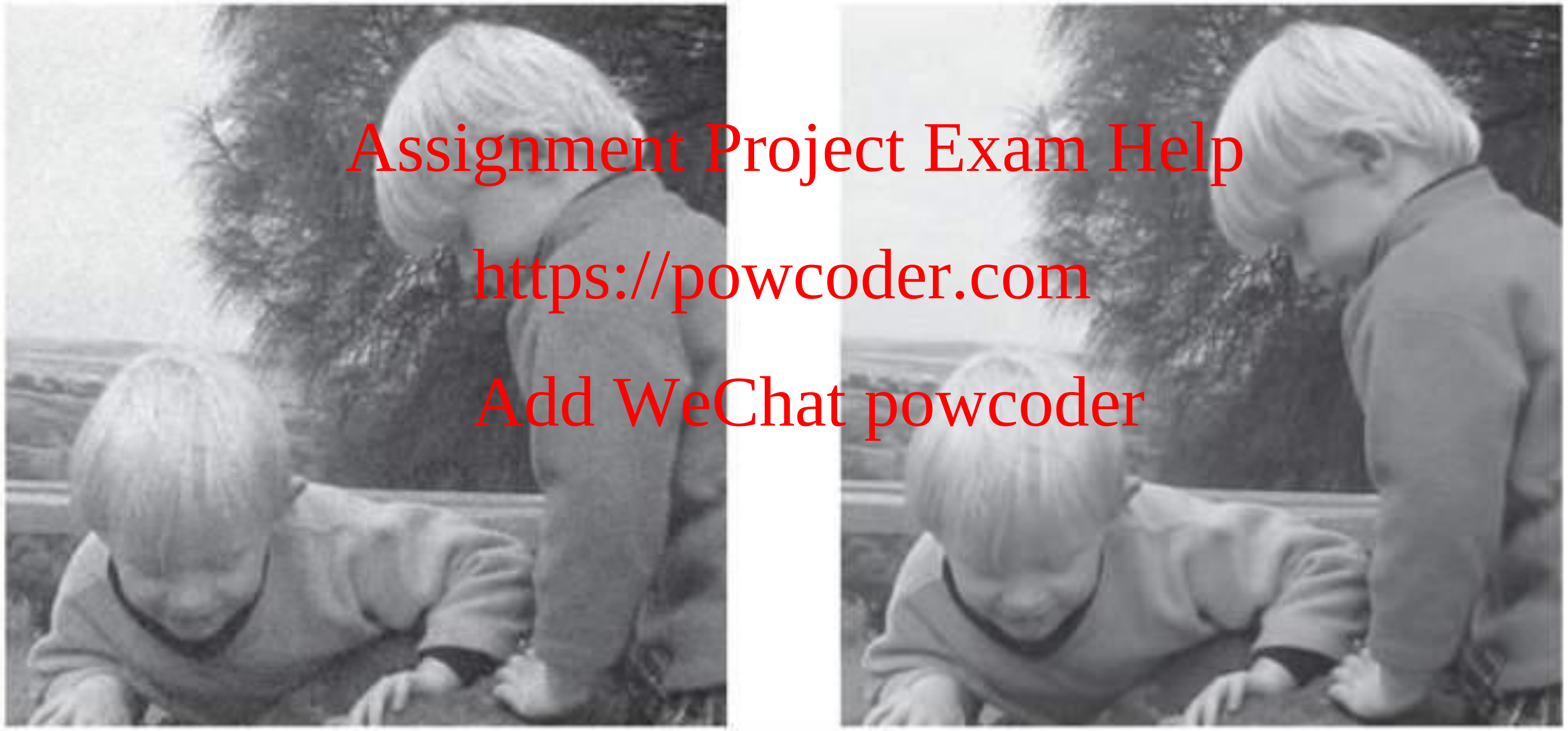
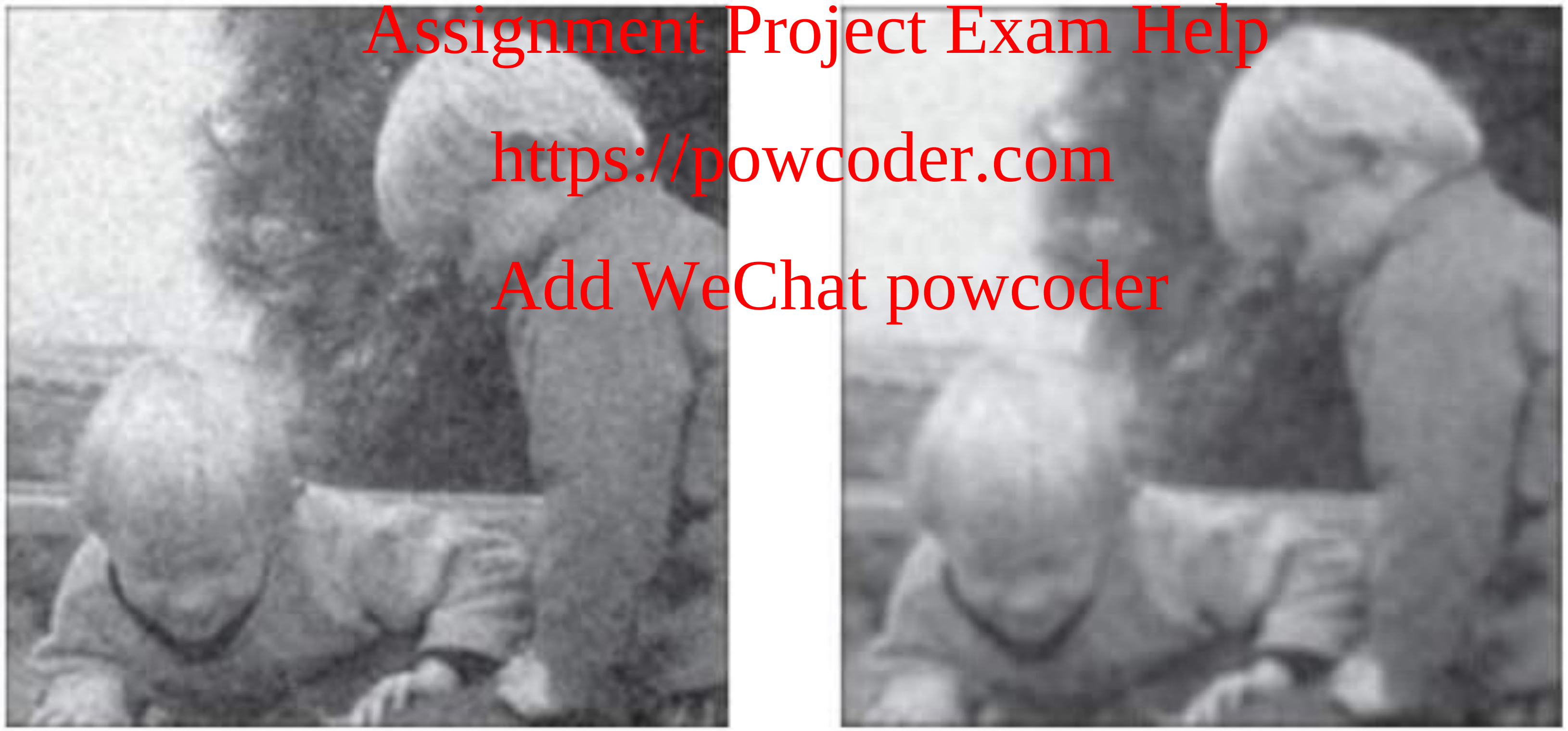


Image averaging to remove Gaussian noise. (a) 10 images (b) 100 images

Gaussian Noise Removal: Simple Average Filtering

- **Simple Average Filtering: Uniform mean filter or Gaussian filter**
  - Using a small window: not so effective in noise removal.
  - Using a large window: effective in noise removal, but the output is over-smoothed.

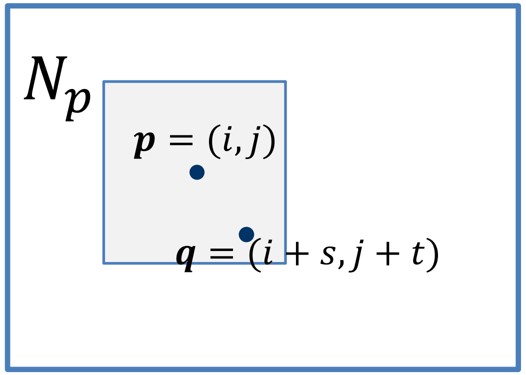


Uniform mean filter to remove Gaussian noise. (a) 3x3 averaging (b) 5x5 averaging

Gaussian Noise Removal: Bilateral Filtering

- **Bilateral filter for grayscale image**
  - One of the most popular filters with various applications
  - Considers both spatial and intensity distances

$$O(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(i+s, j+t)$$



$$w(s,t) = \frac{1}{W(i,j)} \exp\left(-\frac{s^2}{2\sigma_s{}^2} - \frac{t^2}{2\sigma_t{}^2}\right) \exp\left(-\frac{(I(i,j) - I(i+s,j+t))^2}{2\sigma_r{}^2}\right)$$

$$W(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} \exp\left(-\frac{m^2}{2\sigma_s{}^2} - \frac{n^2}{2\sigma_t{}^2}\right) \exp\left(-\frac{(I(i,j) - I(i+m,j+n))^2}{2\sigma_r{}^2}\right)$$

  - This can be rewritten as:

$$O_{\boldsymbol{p}} = \frac{1}{W_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}\left(\left|I_{\boldsymbol{p}} - I_{\boldsymbol{q}}\right|\right) I_{\boldsymbol{q}}$$

$$W_{\boldsymbol{p}} = \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}\left(\left|I_{\boldsymbol{p}} - I_{\boldsymbol{q}}\right|\right)$$

Gaussian Noise Removal: Bilateral Filtering

- **Bilateral filter for color image**
  - Applying filter to each channel

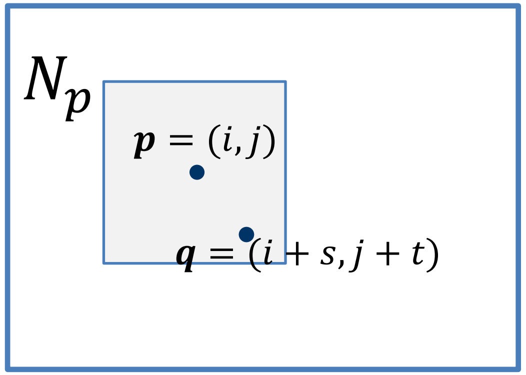
$$C_{\boldsymbol{p}} = (R_{\boldsymbol{p}}, G_{\boldsymbol{p}}, B_{\boldsymbol{p}})^{\mathrm{T}}$$

$$R_{\boldsymbol{p}} = \frac{1}{W_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}(|C_{\boldsymbol{p}} - C_{\boldsymbol{q}}|) R_{\boldsymbol{q}}$$

$$G_{\boldsymbol{p}} = \frac{1}{W_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}(|C_{\boldsymbol{p}} - C_{\boldsymbol{q}}|) G_{\boldsymbol{q}}$$

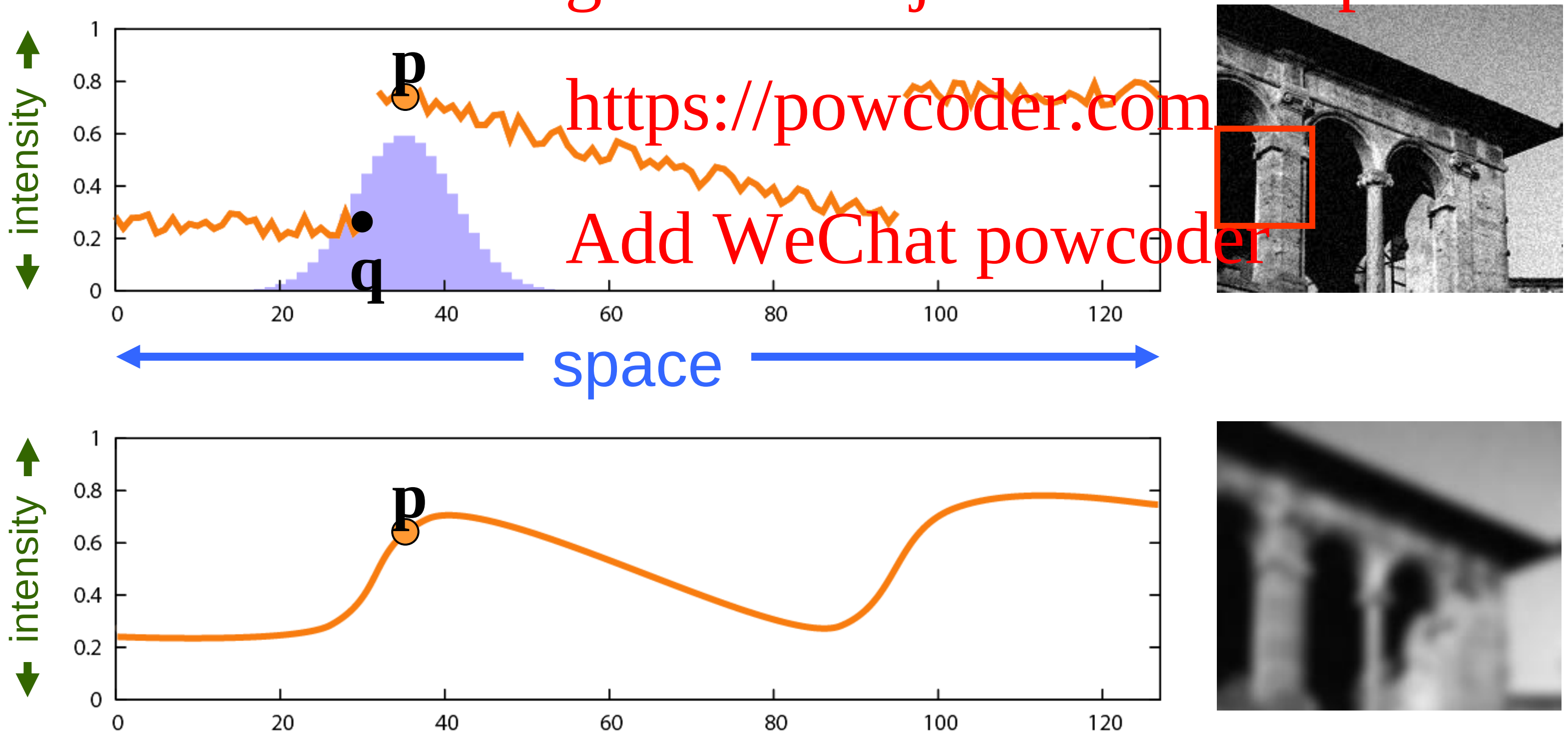
$$B_{\boldsymbol{p}} = \frac{1}{W_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}(|C_{\boldsymbol{p}} - C_{\boldsymbol{q}}|) B_{\boldsymbol{q}}$$

$$W_{\boldsymbol{p}} = \sum_{\boldsymbol{q} \in N_p} G_{\sigma_s}(|\boldsymbol{p} - \boldsymbol{q}|) G_{\sigma_r}(|C_{\boldsymbol{p}} - C_{\boldsymbol{q}}|)$$



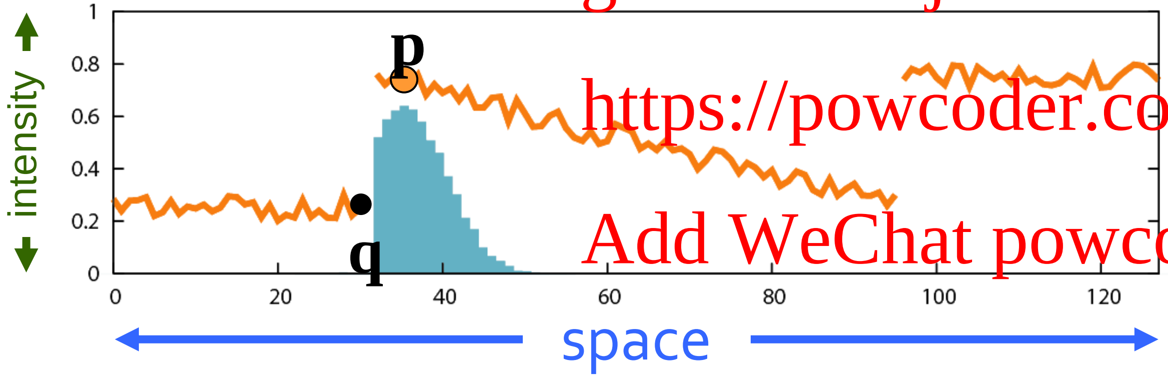
Gaussian filter vs. Bilateral filter

- **Gaussian filter**
  - Weighted average of neighbors
  - Depends only on spatial distance
  - No edge term

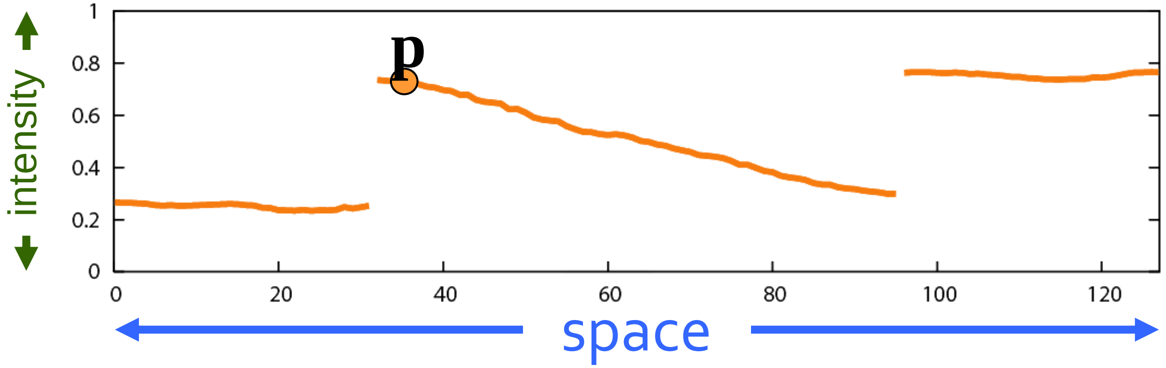


Gaussian filter vs. Bilateral filter

- **Bilateral filter**
  - Weighted average of neighbors
  - Depends on spatial and range difference



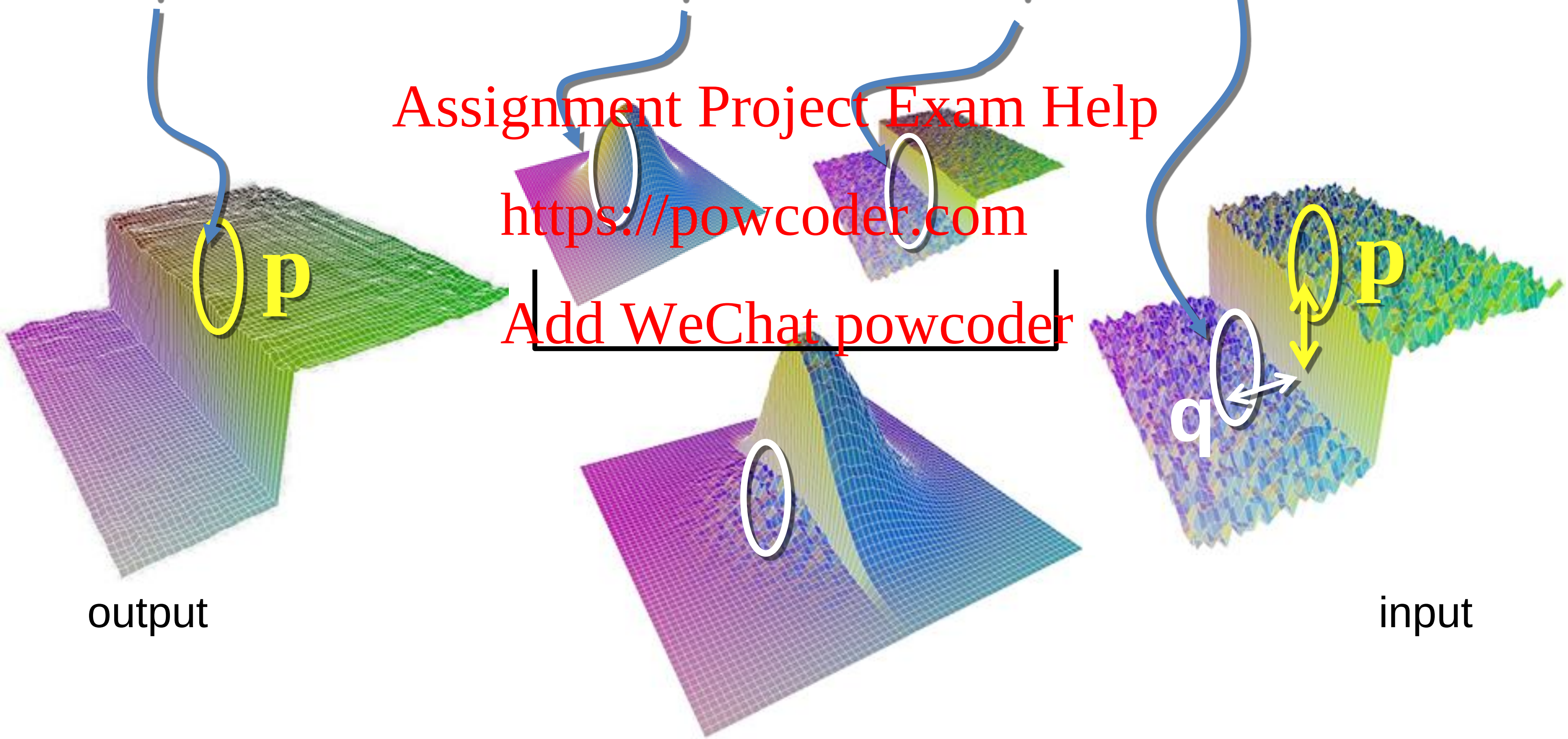
Input



Output

Bilateral filter on a height field

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(||\mathbf{p}-\mathbf{q}||)\; G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\; I_{\mathbf{q}}$$



reproduced from [Durand 02]

Gaussian Noise Removal: Non-local Means Filtering

- **Same goals:**
  - Smooth within Similar Regions
- **KEY INSIGHT:**
  - Generalize, extend 'Similarity'

- **Bilateral:**
  - Averages neighbors with similar intensities;
- **NL-Means:**
  - Averages neighbors with similar neighborhoods!

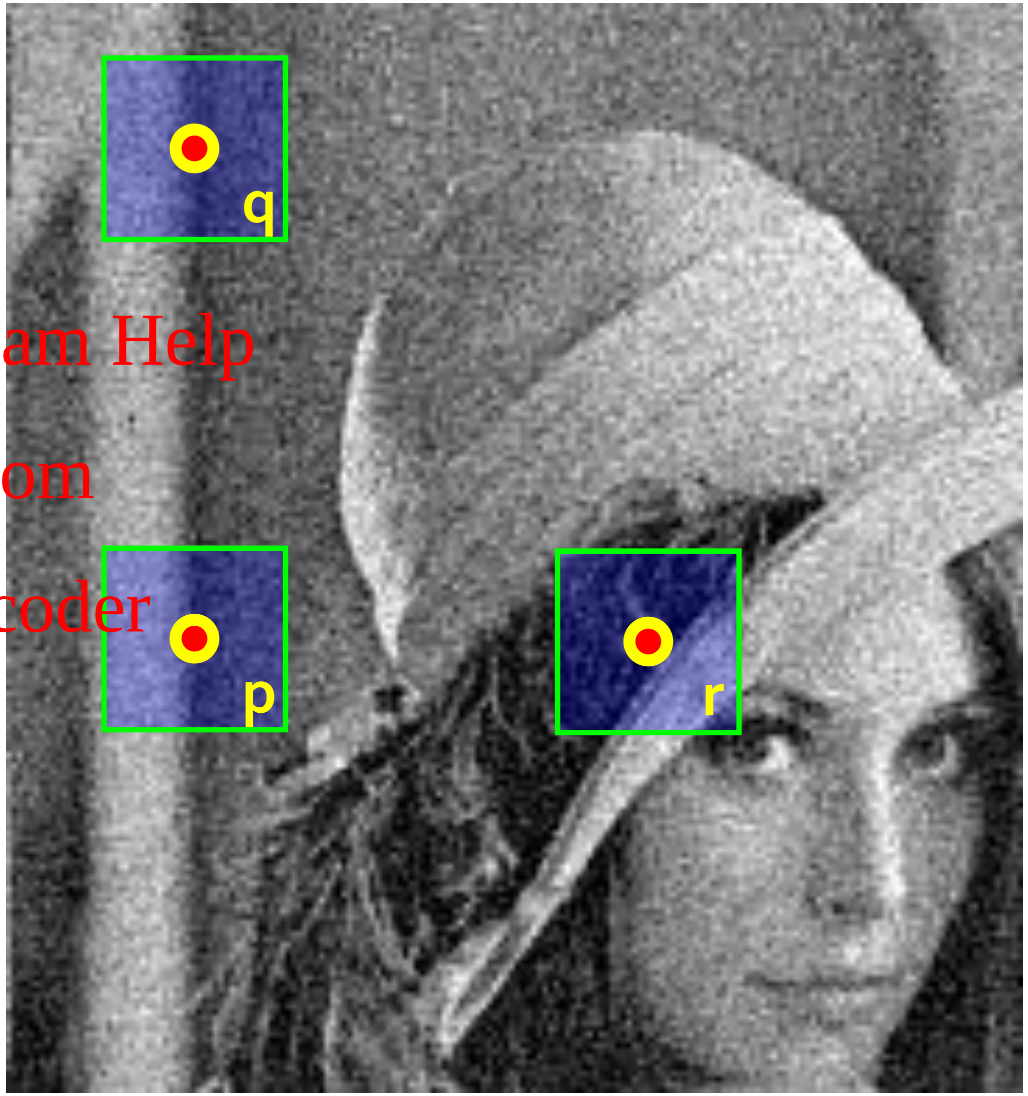
Gaussian Noise Removal: Non-local Means Filtering

- **For each pixel p:**
  - Define a small, simple fixed size neighborhood;
  - Define vector $\mathbf{V}_p$: a list of neighboring pixel values.

$$\mathbf{V}_p = \begin{bmatrix} 0.74 \\ 0.32 \\ 0.41 \\ 0.55 \\ \ldots \\ \ldots \\ \ldots \end{bmatrix}$$

Gaussian Noise Removal: Non-local Means Filtering

- **For each pixel p:**
  - Define a small, simple fixed size neighborhood;
  - Define vector V_p: a list of neighboring pixel values.
  - 'Similar' pixels **p, q** → SMALL distance $\|V_p - V_q\|^2$
  - 'Dissimilar' pixels **p, r** → LARGE distance $\|V_p - V_r\|^2$

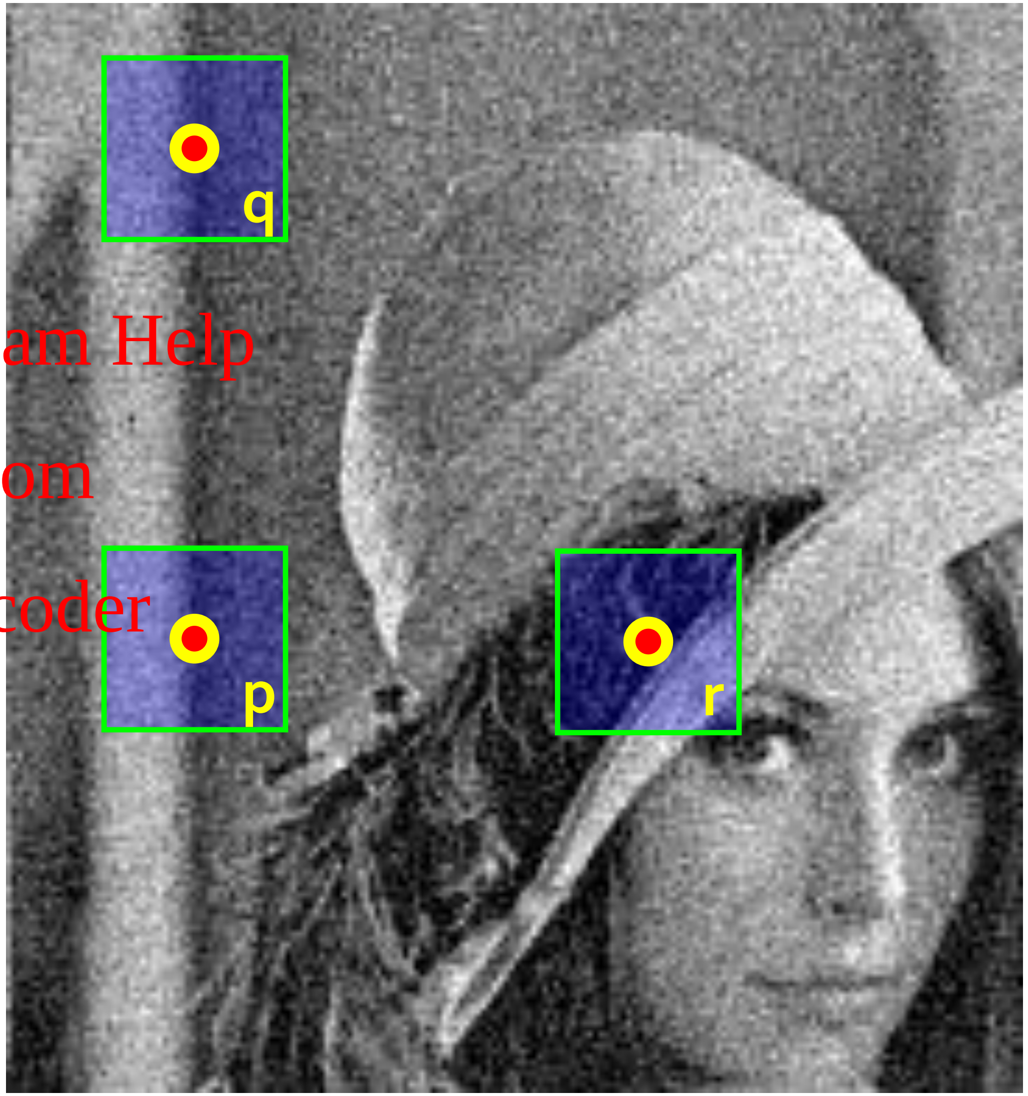


Gaussian Noise Removal: Non-local Means Filtering

- **For each pixel p:**
  - Define a small, simple fixed size neighborhood;
  - Define vector $\mathbf{V}_\mathbf{p}$: a list of neighboring pixel values.
  - 'Similar' pixels **p, q** → SMALL distance $\|\mathbf{V}_\mathbf{p} - \mathbf{V}_\mathbf{q}\|^2$
  - 'Dissimilar' pixels **p, r** → LARGE distance $\|\mathbf{V}_\mathbf{p} - \mathbf{V}_\mathbf{r}\|^2$

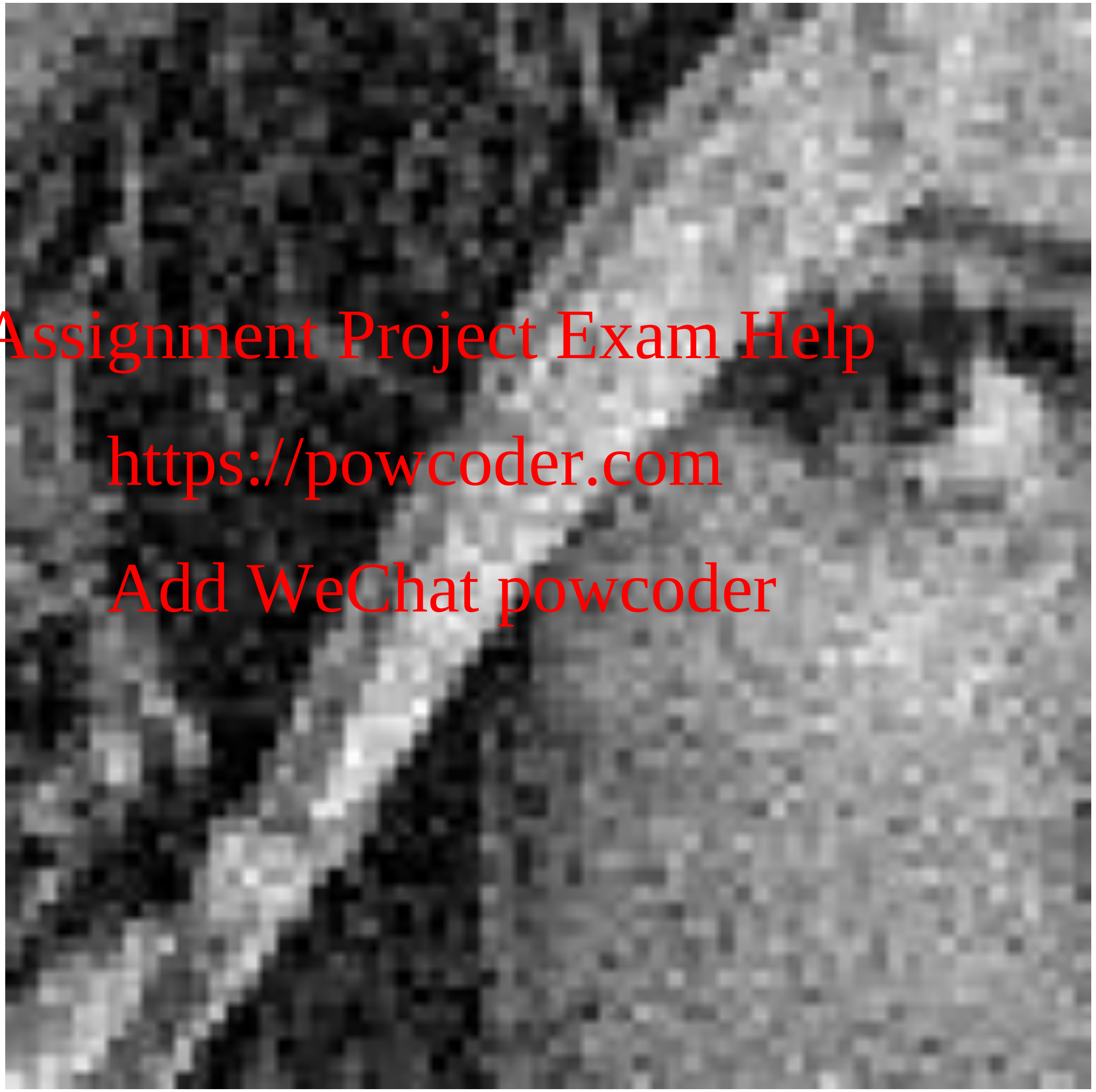
- **Filtering with this neighboring pixels!**
  - No spatial terms,
  - Measures the distance between patches (neighbor pixels)

$$NLMF[I]_\mathbf{p} = \frac{1}{W_\mathbf{p}} \sum_{\mathbf{q}\in S} \cancel{G_{\sigma_s}(||\mathbf{p}-\mathbf{q}||)} \boxed{G_{\sigma_r}\left(||\vec{V}_\mathbf{p} - \vec{V}_\mathbf{q}||^2\right)} I_\mathbf{q}$$



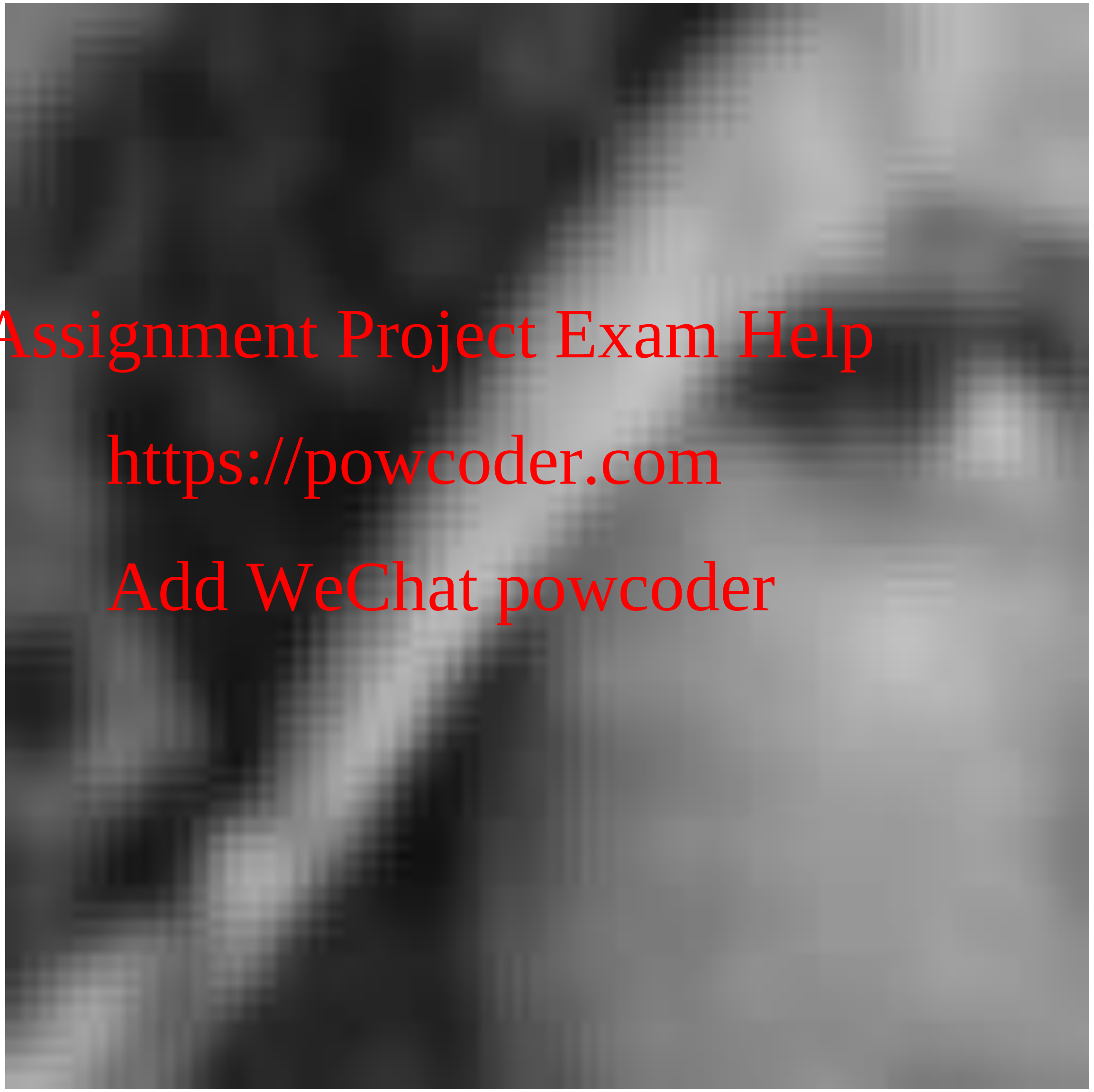
Gaussian Noise Removal: Non-local Means Filtering

- Noisy source image:



Gaussian Noise Removal: Non-local Means Filtering

- **Gaussian Filter**
  - Low noise
  - Low detail

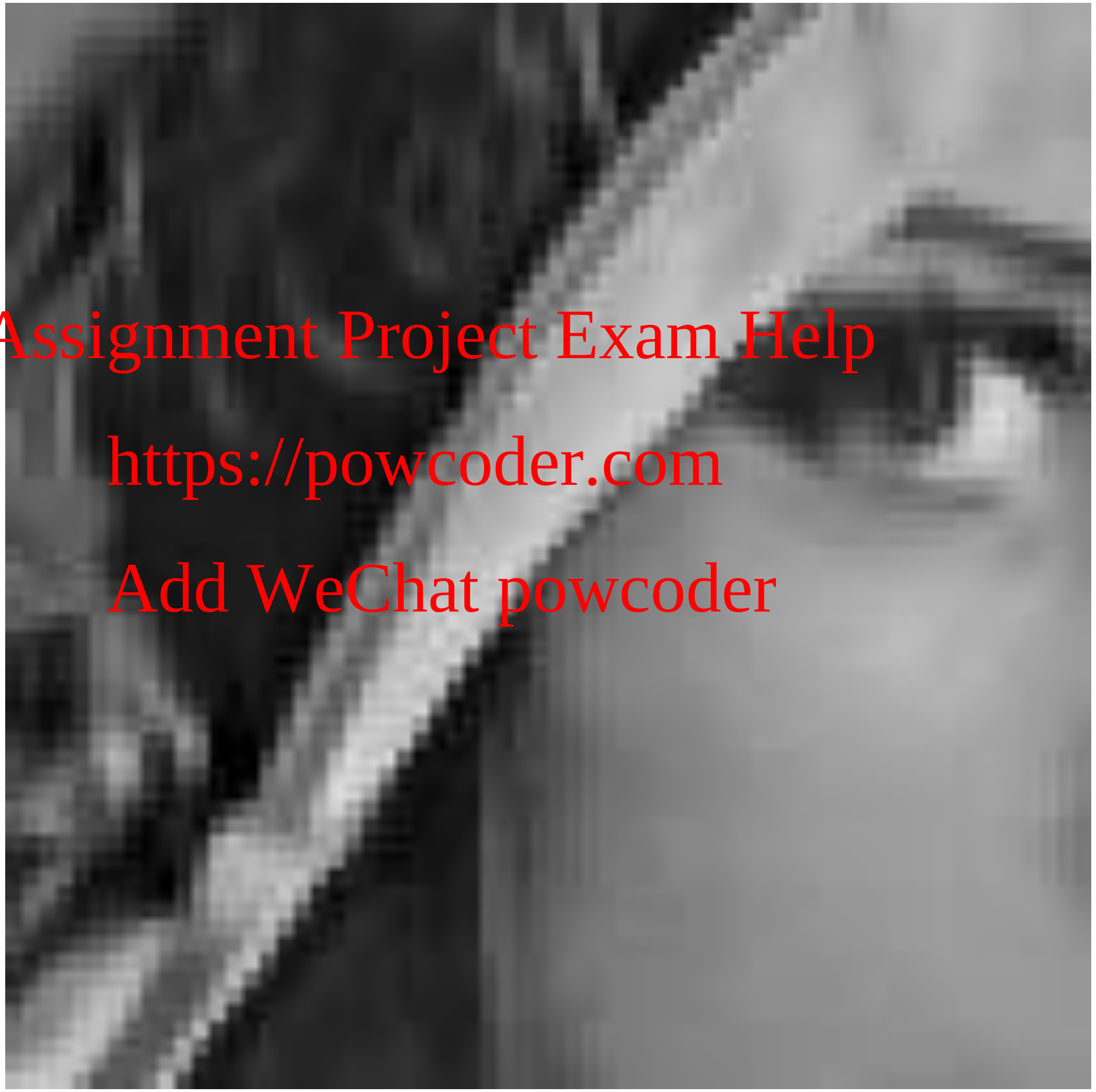


Gaussian Noise Removal: Non-local Means Filtering

- **Bilateral Filter**
  - Better at noise removal
  - but 'stairsteps'

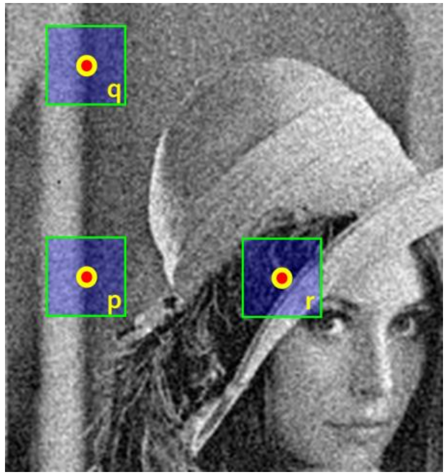
Gaussian Noise Removal: Non-local Means Filtering

- **NL-Means**
  - Sharp
  - Low noise
  - Few artifacts

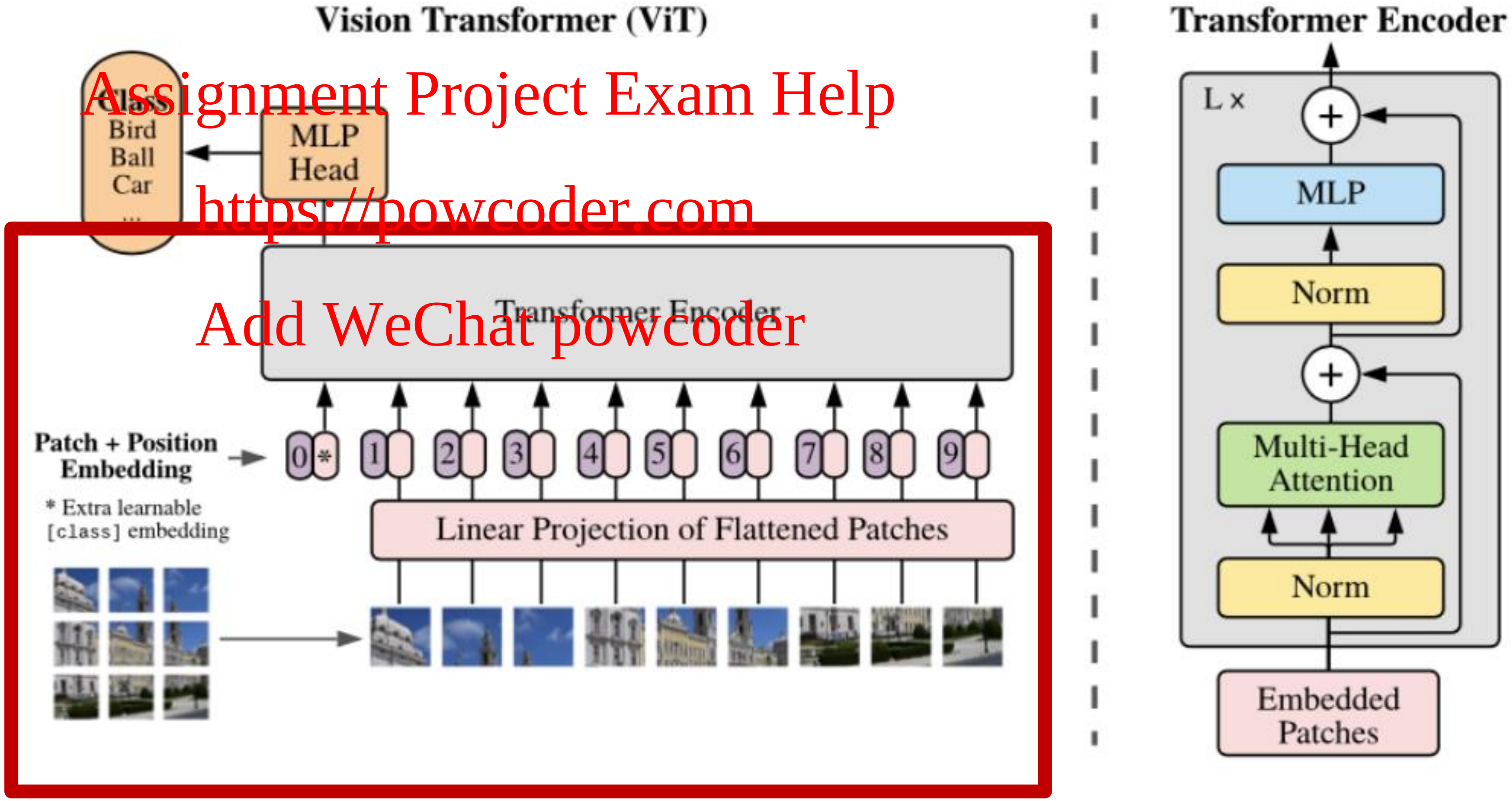


Non-local Means Filtering: Old-fashioned?

- **Vision Transformer (in deep learning)**
  - A new architecture that shows state-of-the-art performance in computer vision tasks

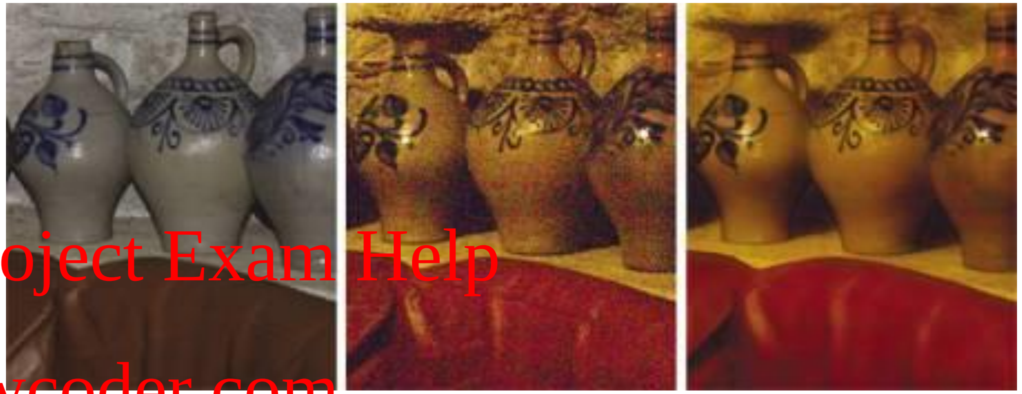


$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_r}(||\vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}}||^2)\, I_{\mathbf{q}}$$

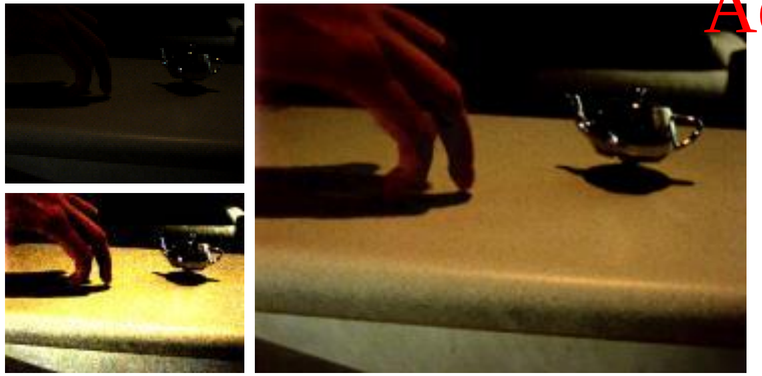


Many applications, not limited to denoising

Tone Mapping [Durand 02]



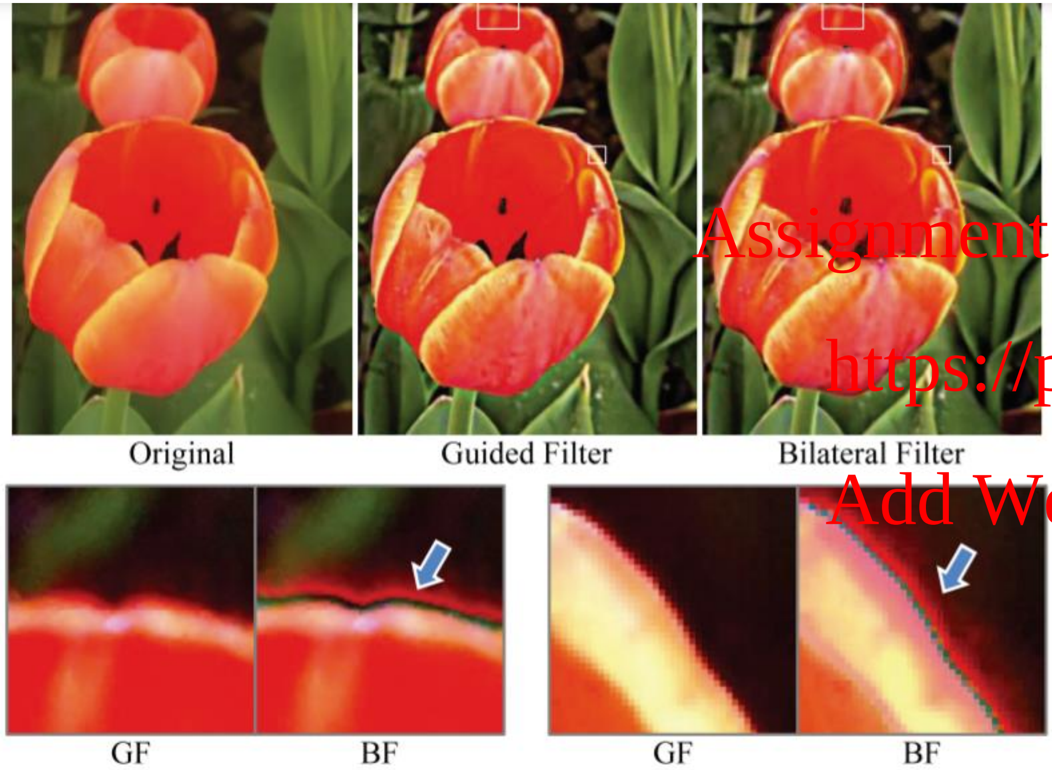
Flash / No-Flash [Eisemann 04, Petschnigg 04]



Virtual Video Exposure [Bennett 05]

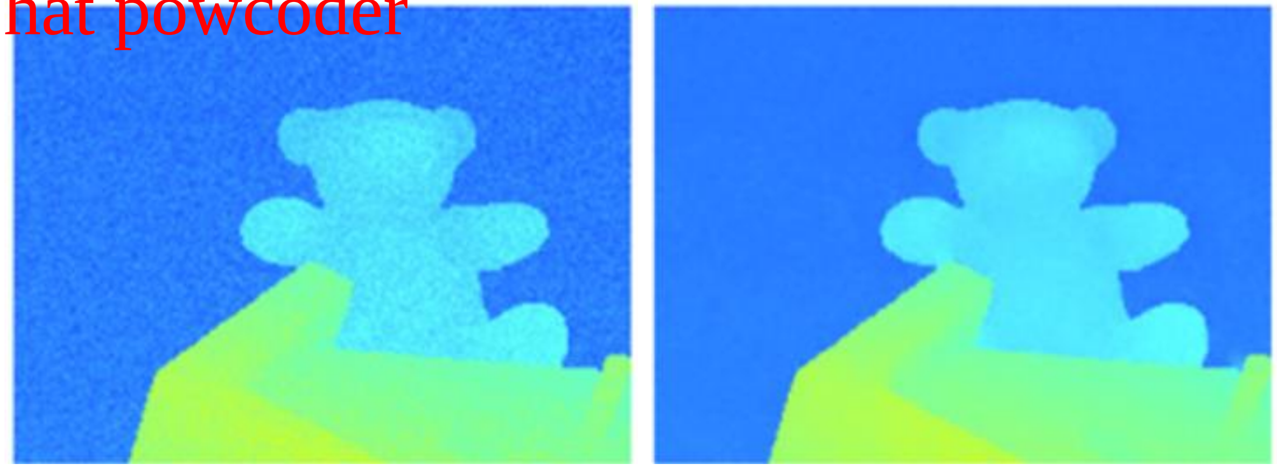
Tone Management [Bae 06]

Edge-aware smoothing filters



[He 2010]

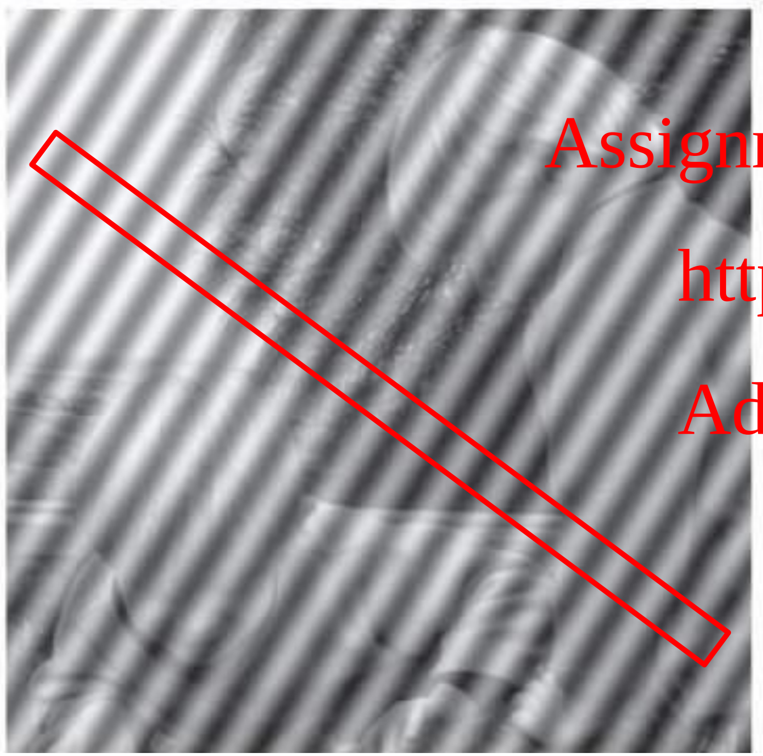
[Li 2011]

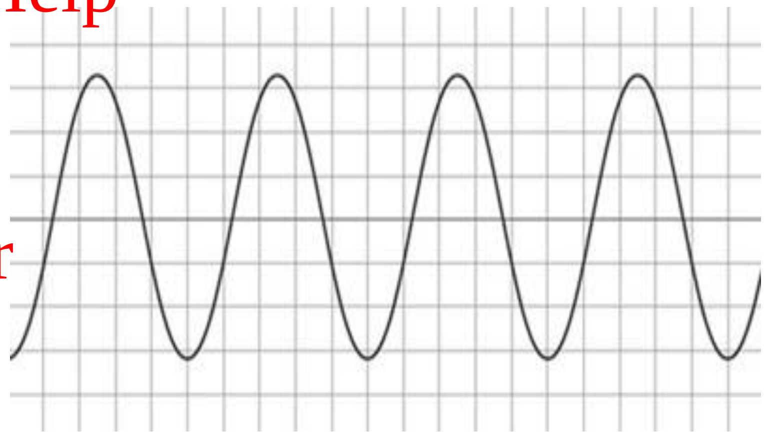


[Shen 2015]

Periodic Noise Removal: Frequency Domain Filtering

- **Periodic fluctuation can be modeled as sinusoid waves**





Periodic Noise Removal: Frequency Domain Filtering

- **Periodic fluctuation can be modeled as sinusoid waves**

- Spatial domain

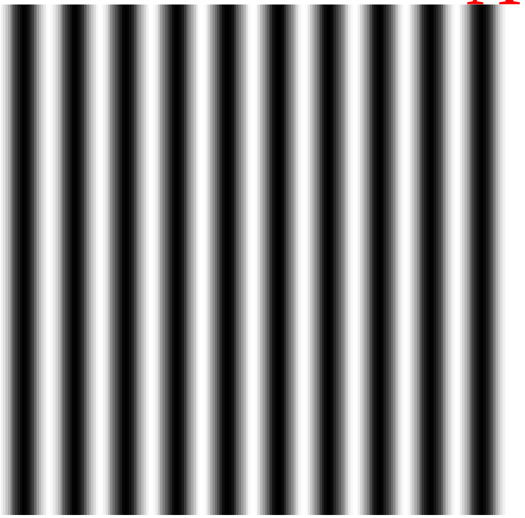
$$r(x,y) = A\sin\left[\frac{2\pi u_0(x+B_x)}{M} + \frac{2\pi v_0(y+B_y)}{N}\right]$$

A : amplitude
u_0, v_0 : sinusoidal frequencies
B_x, B_y : phase displacement w.r.t. the origin

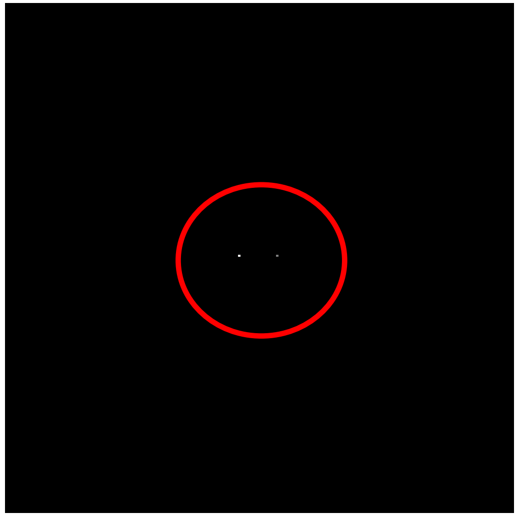
- Frequency domain

$$R(u,v) = j\frac{AMN}{2}\left[e^{-j2\pi\left(\frac{u_0B_x}{M}+\frac{v_0B_y}{N}\right)}\delta(u+u_0,v+v_0) - e^{j2\pi\left(\frac{u_0B_x}{M}+\frac{v_0B_y}{N}\right)}\delta(u-u_0,v-v_0)\right]$$

Sinusoidal wave in the spatial domain



Two symmetric spikes in the Fourier domain



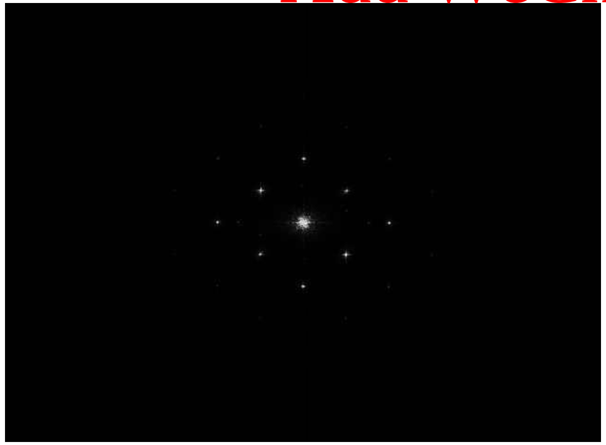
Periodic Noise Removal: Frequency Domain Filtering

- **Periodic noise reduction by selective filters**
  - Notch filters can efficiently remove the periodic noise
  - Step 1: analyze the Fourier spectrum F of the image

https://docs.opencv.org/3.4/d2/d0b/tutorial_periodic_noise_removing_filter.html

Periodic Noise Removal: Frequency Domain Filtering

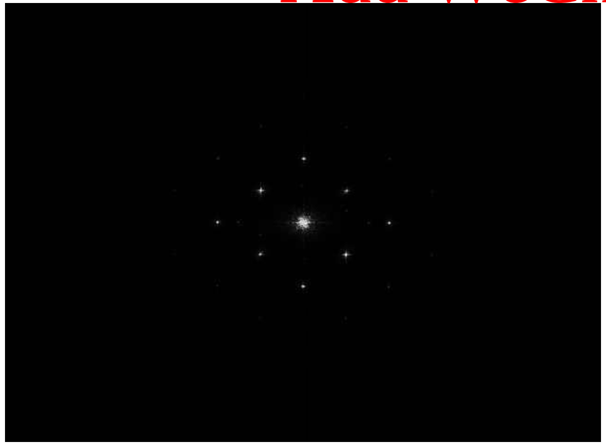
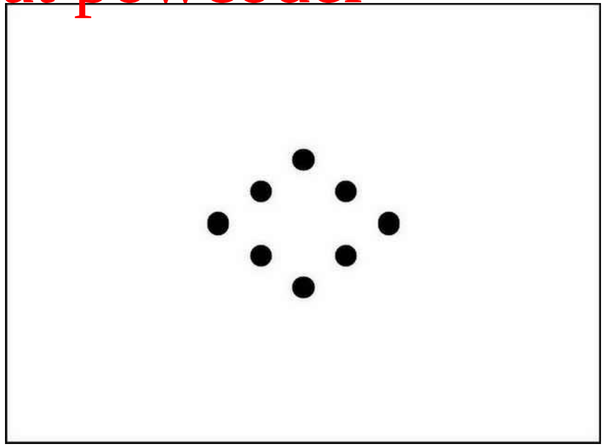
- **Periodic noise reduction by selective filters**
  - Notch filters can efficiently remove the periodic noise
  - Step 1: analyze the Fourier spectrum F of the image
  - Step 2: identify the locations of the peaks in F



https://docs.opencv.org/3.4/d2/d0b/tutorial_periodic_noise_removing_filter.html

Periodic Noise Removal: Frequency Domain Filtering

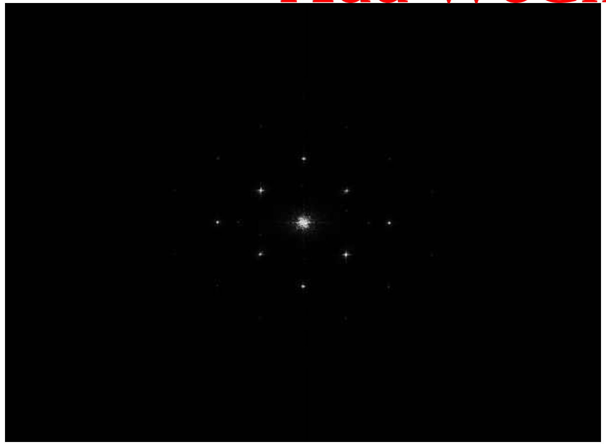
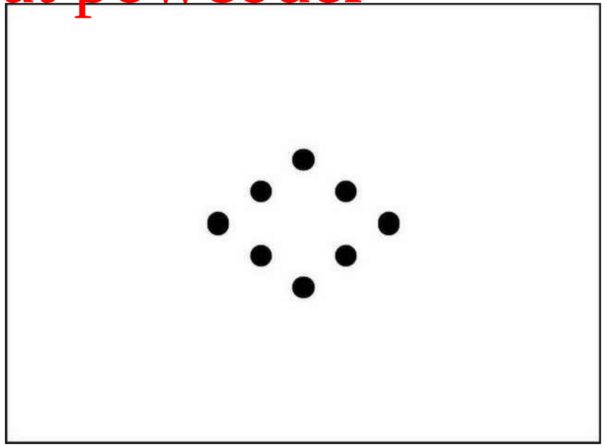
- **Periodic noise reduction by selective filters**
  - Notch filters can efficiently remove the periodic noise
  - Step 1: analyze the Fourier spectrum F of the image
  - Step 2: identify the locations of the peaks in F
  - Step 3: construct a notch reject filter H in Fourier domain, whose centers are at peaks

https://docs.opencv.org/3.4/d2/d0b/tutorial_periodic_noise_removing_filter.html

Periodic Noise Removal: Frequency Domain Filtering

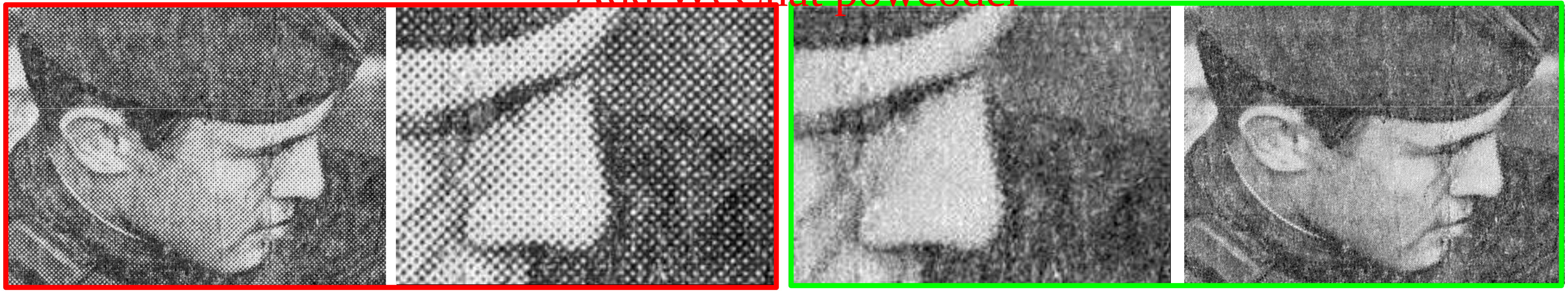
- **Periodic noise reduction by selective filters**
  - Notch filters can efficiently remove the periodic noise
  - Step 1: analyze the Fourier spectrum F of the image
  - Step 2: identify the locations of the peaks in F
  - Step 3: construct a notch reject filter H in Fourier domain, whose centers are at peaks
  - Step 4: use H to filter F to get the result


https://docs.opencv.org/3.4/d2/d0b/tutorial_periodic_noise_removing_filter.html

Periodic Noise Removal: Frequency Domain Filtering

- **Periodic noise reduction by selective filters**
  - Notch filters can efficiently remove the periodic noise
  - Step 1: analyze the Fourier spectrum F of the image
  - Step 2: identify the locations of the peaks in F
  - Step 3: construct a notch reject filter H in Fourier domain, whose centers are at peaks
  - Step 4: use H to filter F to get the result



https://docs.opencv.org/3.4/d2/d0b/tutorial_periodic_noise_removing_filter.html

Summary

- **4 different type of noises**
  - Salt and Pepper noise, Gaussian noise, Speckle noise, Periodic noise

- **Salt-and-Pepper Noise Removal**
  - Median filtering produces the best results.
- **Gaussian Noise Removal**
  - Additive White Gaussian Noise (AWGN): the most widely used noise model
  - Adaptive filtering produces satisfactory results.
- **Periodic Noise Removal**
  - Filtering in Frequency domain

Next topic

- **How can we represent images more effectively?**
  - Prerequisite
    - Review EBU6230 Image/Video Processing – Week3: Edges
    - Review EBU6230 Image/Video Processing – Week3: Interest Points