INTRODUCTION TO IMAGE FILTERING, IMAGE DENOISING, IMAGE INTERPOLATION

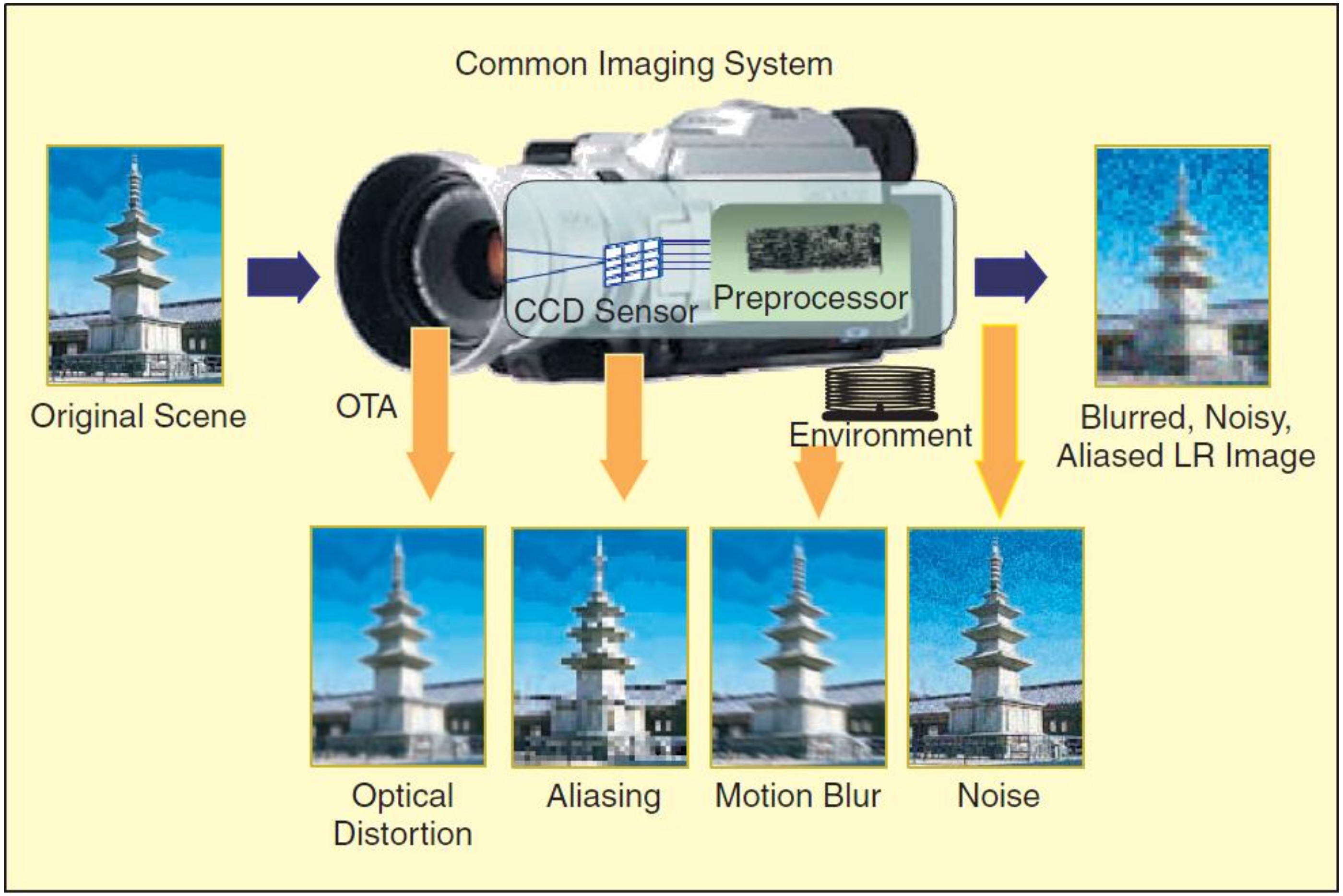
Speaker: Hsin-Hui Chen

Date: 2013.9.5 (Thurs)

Outline

- Introduction
- Image Filtering (Basic concepts)
- Image Denoising
- Image interpolation (super resolution)

Common Image Acquisition System



Computational Problems in Imaging

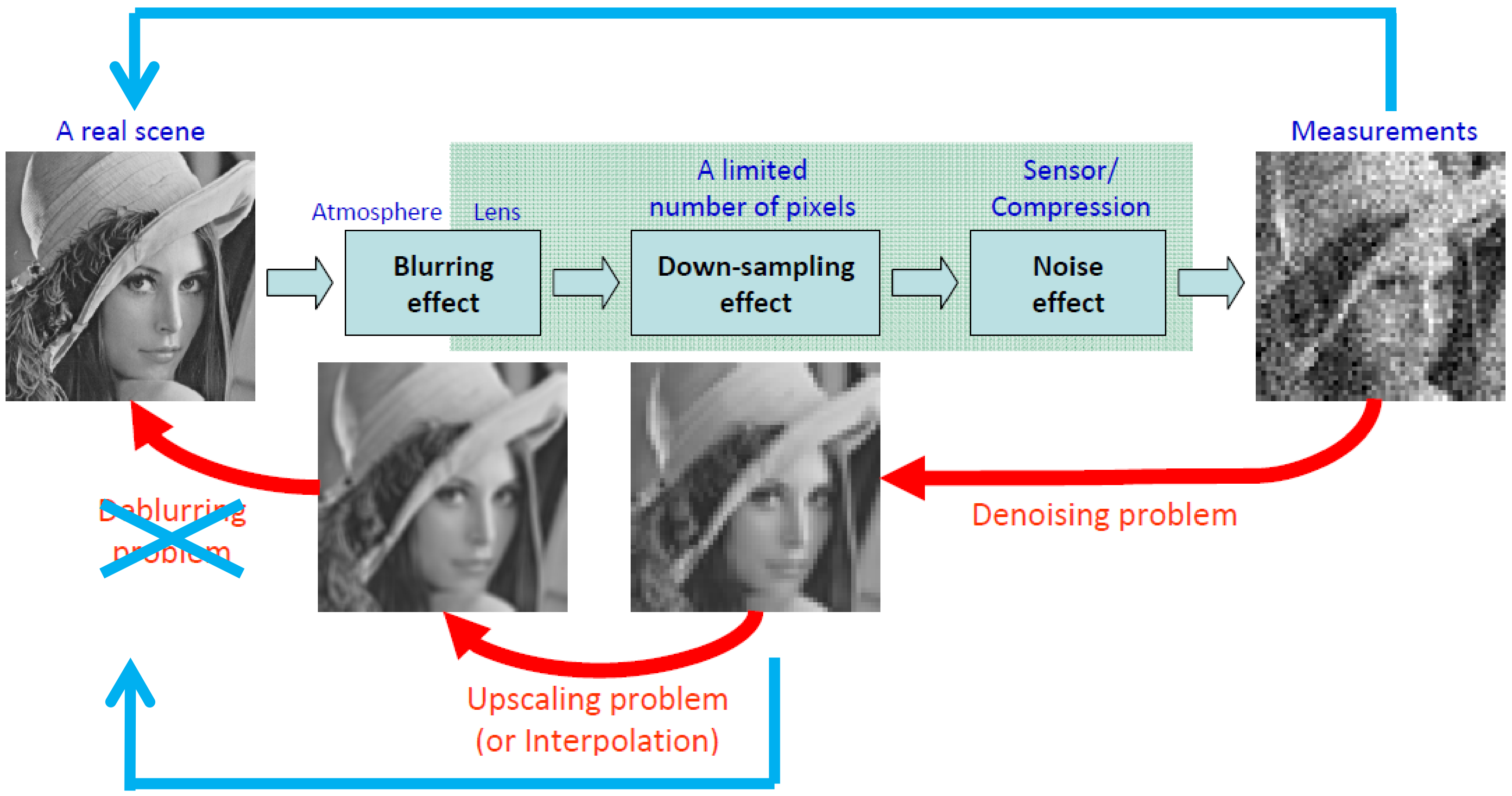


Image Denoising

- Gaussian noise with σ=15, 25, 35



Desired denoised image

Image Interpolation

Desired interpolated image

Evaluation Metrics (1/3)

- **PSNR**
- 30dB

$$\text{PSNR (dB)} = 10 \cdot \log_{10}\left[\frac{\sum_{x=1}^{N_x}\sum_{y=1}^{N_y}(255)^2}{\sum_{x=1}^{N_x}\sum_{y=1}^{N_y}\left(f(x,y)-\hat{f}(x,y)\right)^2}\right] = 10 \cdot \log_{10}\left[\frac{N_x N_y (255)^2}{\sum_{x=1}^{N_x}\sum_{y=1}^{N_y}\left(f(x,y)-\hat{f}(x,y)\right)^2}\right]$$

$$= 10 \cdot \log_{10}\left[\frac{255^2}{MSE}\right]$$

Mean Squared Error (MSE)

$$\text{MSE} = \frac{\sum_{x=1}^{N_x}\sum_{y=1}^{N_y}\left(f(x,y)-\hat{f}(x,y)\right)^2}{N_x N_y},$$

N_x is the number of pixels of the image in x direction

N_y is the number of pixels of the image in y direction

Evaluation Metrics (2/3)

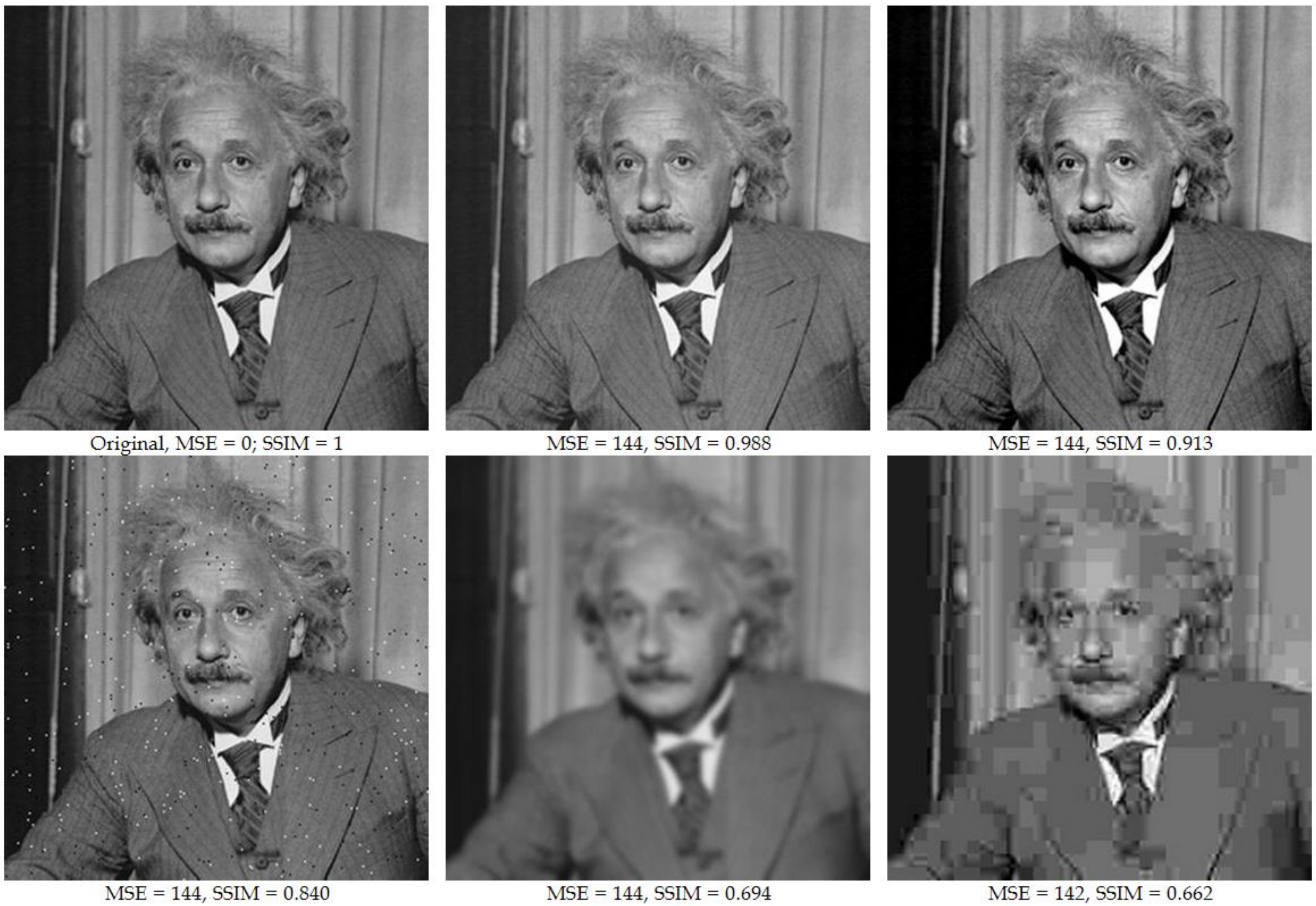
- **SSIM**

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

- μ_x the average of x;
- μ_y the average of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator;
- L the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

Evaluation Metrics (3/3)

- Why SSIM matters?



Example of Simulation Results

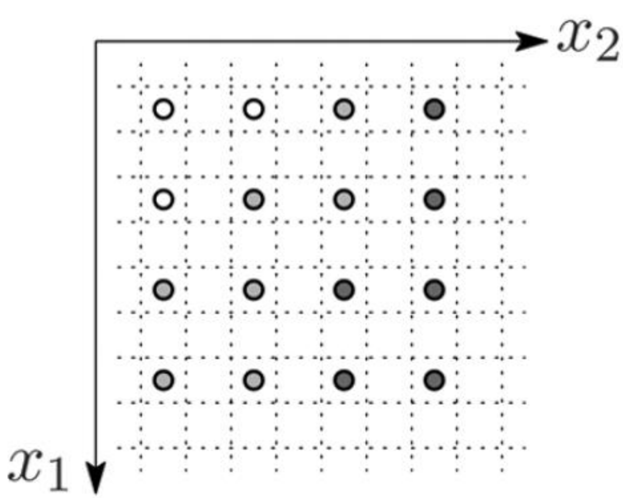
- Image denoising results

The PSNR (dB) and SSIM results of the denoised images at different noise levels and by different schemes.

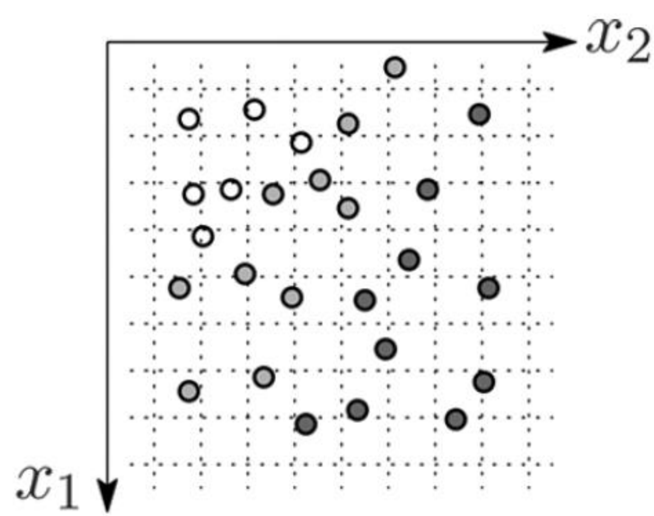
Methods	[10]	[8]	[14]	[20]	Proposed
Lena					
$\sigma = 10$	33.1(0.9154)	33.2(0.9160)	33.5(0.9203)	33.9(0.9272)	33.7(0.9243)
$\sigma = 20$	29.2(0.8455)	29.4(0.8514)	29.7(0.8571)	30.2(0.8699)	29.7(0.8605)
$\sigma = 30$	27.2(0.7878)	27.5(0.7964)	27.8(0.8055)	28.3(0.8231)	27.6(0.8066)
$\sigma = 40$	25.7(0.7315)	26.0(0.7466)	26.2(0.7504)	27.3(0.7727)	26.0(0.7578)
Cameraman					
$\sigma = 10$	33.2(0.9170)	33.7(0.9307)	33.9(0.9334)	34.4(0.9399)	34.1(0.9356)
$\sigma = 20$	29.1(0.8449)	29.6(0.8744)	29.9(0.8810)	30.6(0.8962)	30.1(0.8902)
$\sigma = 30$	26.8(0.7945)	27.5(0.8307)	27.9(0.8426)	28.5(0.8655)	27.8(0.8558)
$\sigma = 40$	25.3(0.7310)	26.0(0.7806)	26.5(0.8048)	27.1(0.8303)	26.2(0.8211)
House					
$\sigma = 10$	34.4(0.8791)	34.8(0.8809)	35.5(0.8960)	36.2(0.9143)	35.6(0.9012)
$\sigma = 20$	31.3(0.8199)	32.1(0.8374)	32.7(0.8458)	33.3(0.8553)	32.5(0.8471)
$\sigma = 30$	29.4(0.7829)	30.2(0.8066)	30.7(0.8137)	31.6(0.8319)	30.4(0.8185)
$\sigma = 40$	28.1(0.7409)	28.9(0.7708)	29.1(0.7771)	30.7(0.8065)	28.9(0.7891)
Paint					
$\sigma = 10$	33.0(0.9227)	33.5(0.9319)	33.5(0.9293)	33.7(0.9329)	33.6(0.9311)
$\sigma = 20$	29.0(0.8513)	29.6(0.8687)	29.6(0.8655)	29.9(0.8731)	29.5(0.8683)
$\sigma = 30$	26.9(0.7897)	27.5(0.8110)	27.5(0.8091)	27.7(0.8196)	27.2(0.8088)
$\sigma = 40$	25.6(0.7408)	26.0(0.7616)	26.0(0.7599)	26.6(0.7711)	25.6(0.7569)
Monarch					
$\sigma = 10$	33.1(0.9442)	33.6(0.9527)	33.5(0.9501)	33.9(0.9577)	34.2(0.9594)
$\sigma = 20$	28.8(0.8912)	29.5(0.9076)	29.6(0.9077)	30.1(0.9222)	30.0(0.9202)
$\sigma = 30$	26.5(0.8370)	27.1(0.8583)	27.4(0.8663)	28.0(0.8850)	27.4(0.8769)
$\sigma = 40$	25.0(0.7916)	25.7(0.8179)	25.9(0.8260)	26.6(0.8462)	25.9(0.8378)
Barbara					
$\sigma = 10$	31.6(0.9241)	31.6(0.9246)	32.3(0.9349)	32.7(0.9420)	32.5(0.9378)
$\sigma = 20$	27.4(0.8314)	27.2(0.8316)	28.4(0.8646)	28.9(0.8819)	28.5(0.8716)
$\sigma = 30$	25.1(0.7472)	25.0(0.7475)	26.3(0.7919)	26.8(0.8165)	26.2(0.8028)
$\sigma = 40$	23.5(0.6696)	23.5(0.6718)	24.7(0.7262)	25.0(0.7444)	24.5(0.7378)

Image Interpolation

- Regularly and irregularly sampled data



Interpolation of regularly sampled data



Reconstruction from irregularly sampled data

The Difference between Image Interpolation and Super Resolution (1/2)

- **Interpolation** only involves upsampling the low-resolution image, which is often assumed to be aliased due to direct down-sampling.
- **Super resolution** aims to address undesirable effects, including the resolution degradation, blur and noise effects. Super resolution usually involves three major processes which are upsampling (interpolation), deblurring, and denoising.

The Difference between Image Interpolation and Super Resolution (2/2)

- The formulation of an LR image (Image Interpolation):

$$\mathbf{I}^l = \mathbf{I}^h \downarrow_s$$

- The formulation of an LR image (Super Resolution):

$$\mathbf{I}^l = (\mathbf{I}^h * g) \downarrow_s$$

where D is the down-sampling matrix, and g is the point spread function (PSF) which is generally a smoothing kernel.

Image Filtering

- Example:1-D signal
- Mask, kernel, window…

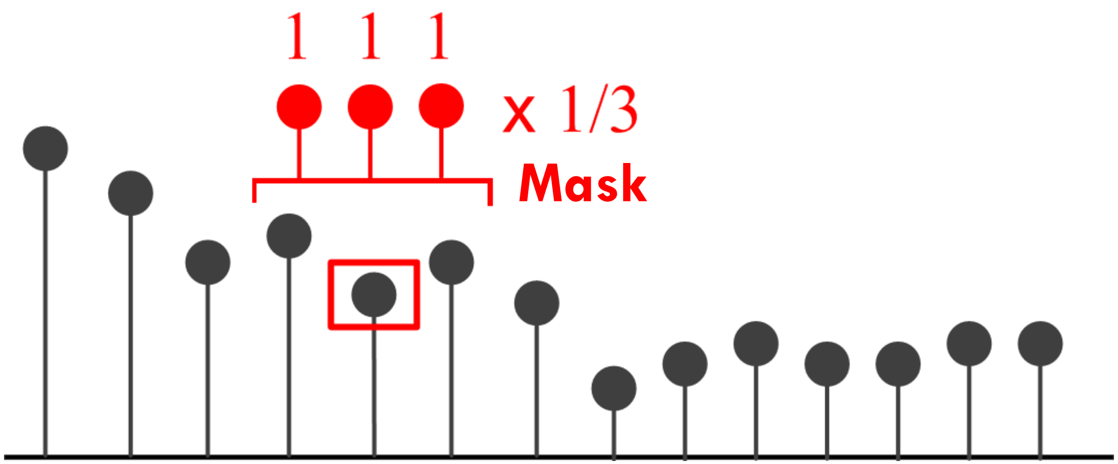


Image Filtering

- Example:1-D signal

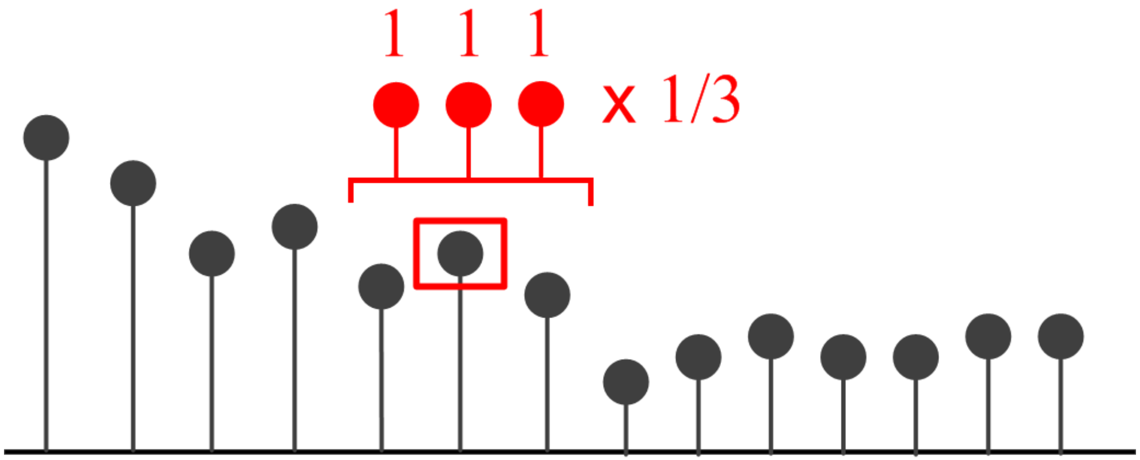


Image Filtering

- Example: 1-D signal

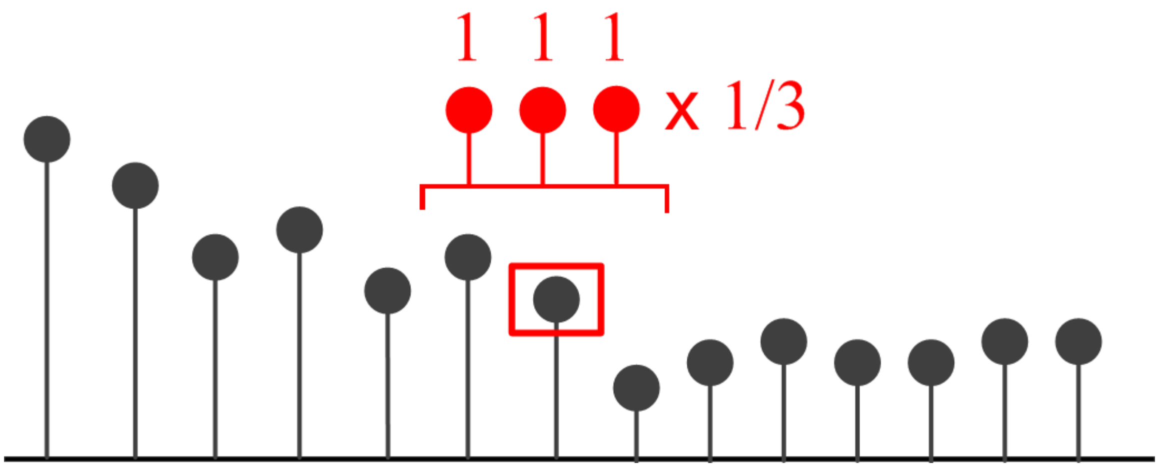


Image Filtering

- Example:1-D signal

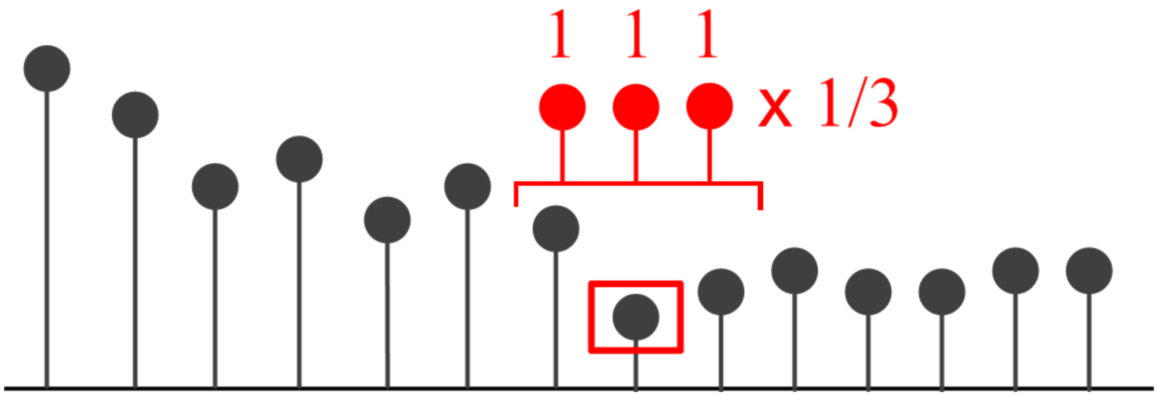


Image Filtering

- Example: 2-D image
- 3x3 mask

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\qquad\frac{1}{16}\times\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

100	100	100	100	100
100	200	205	203	100
100	195	200	200	100
100	200	205	195	100
100	100	100	100	100

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

100	100	100	100	100
100	144	167	145	100
100	167	200	168	100
100	144	166	144	100
100	100	100	100	100

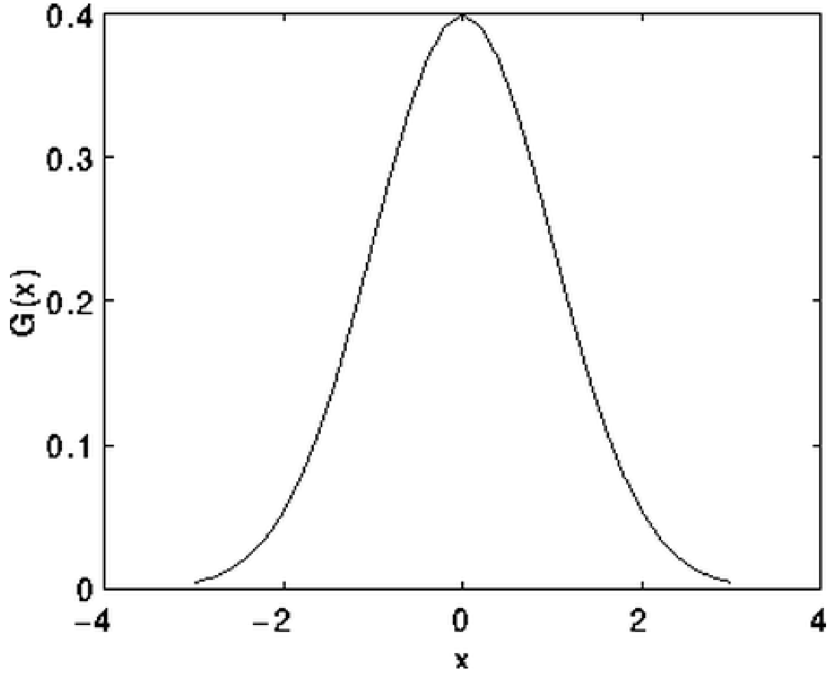
$$\frac{1}{16}\times\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

100	100	100	100	100
100	156	176	158	100
100	174	201	175	100
100	156	175	156	100
100	100	100	100	100

Gaussian Filter

- Example: 1-D case

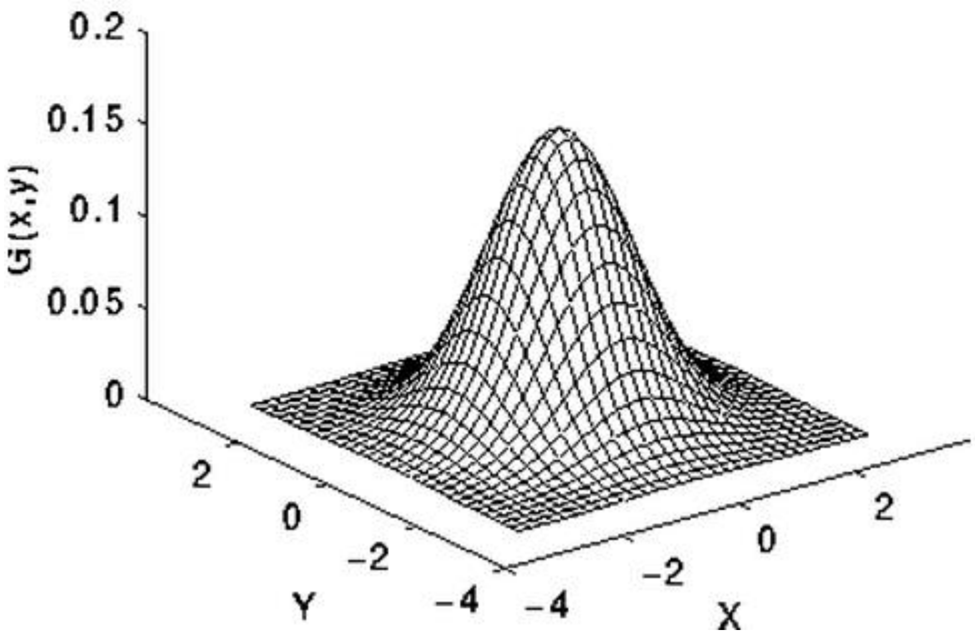
$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$



Gaussian Filter

- Example: 2-D case

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Gaussian Filter

- Discrete approximation
- Example

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

.006	.061	.242	.383	.242	.061	.006

Image Denoising

- Example: 3x3 mask

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$\sum_{i=1}^{9} w_i = 1$$

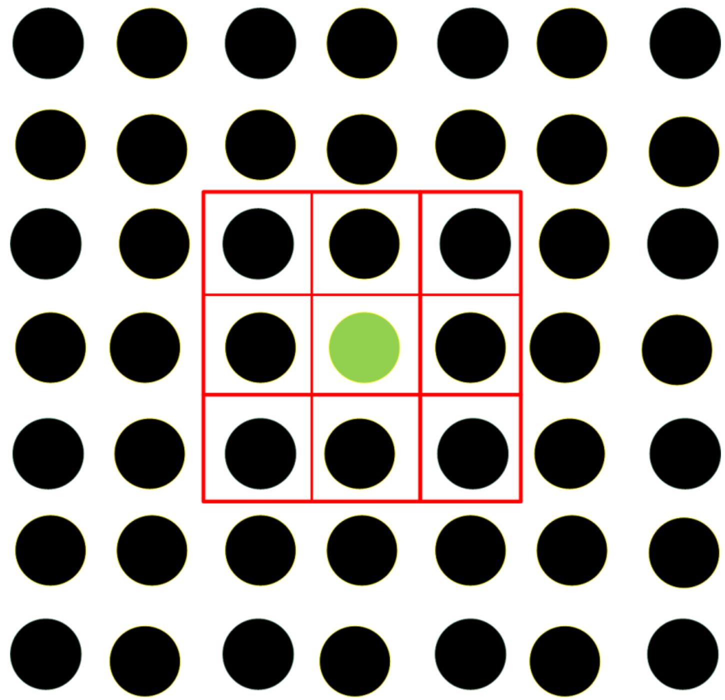
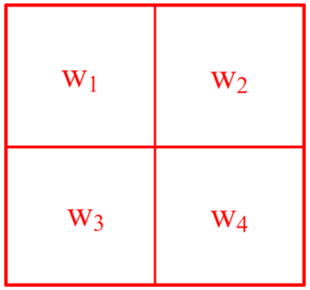


Image Interpolation

- Example: 2x2 mask



$$\sum_{i=1}^{4} w_i = 1$$

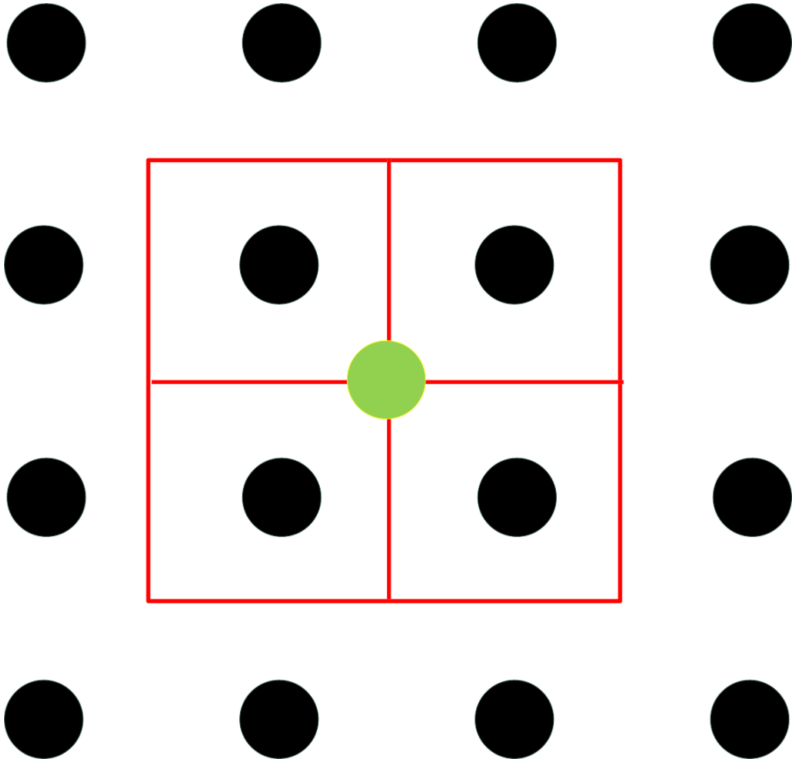
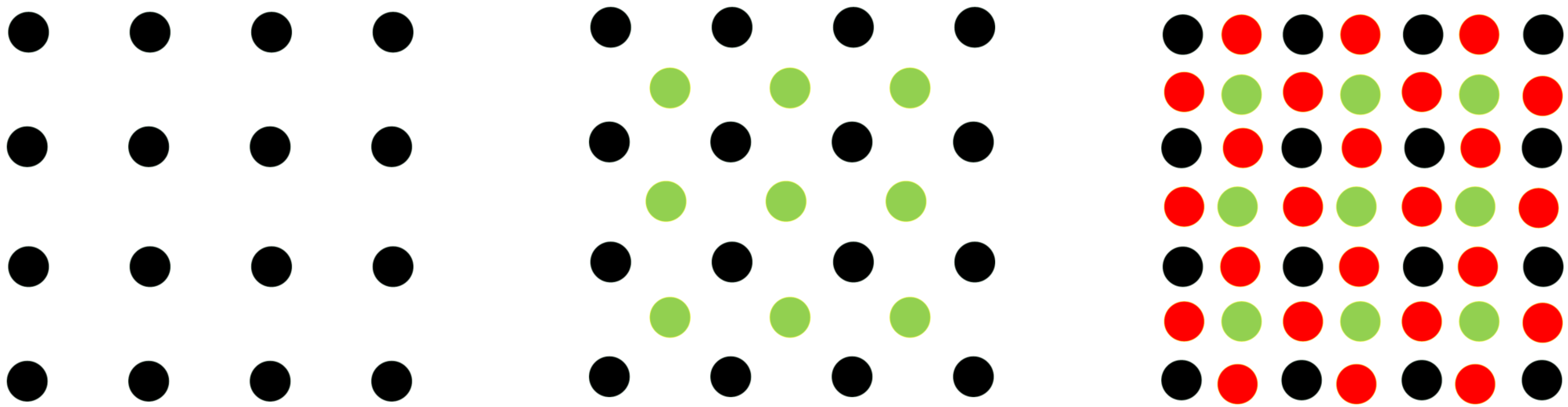
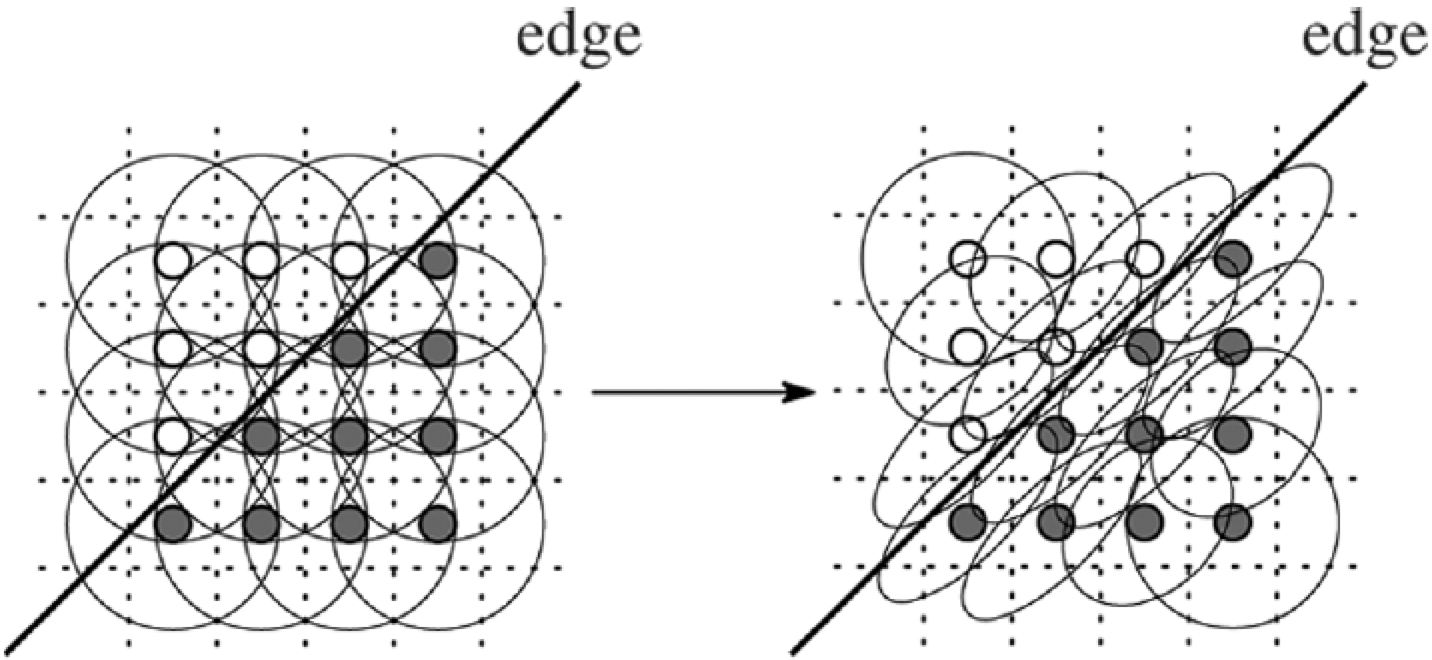


Image Interpolation

- Example: 2X



Classic and Data-adapted Kernels



$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

The Common Framework

- The data measurement model

Zero-mean, i.i.d noise (No other assump.)

$$y_i = z(x_i) + e_i, \qquad \text{for} \quad i = 1, \cdots, n,$$

Given samples

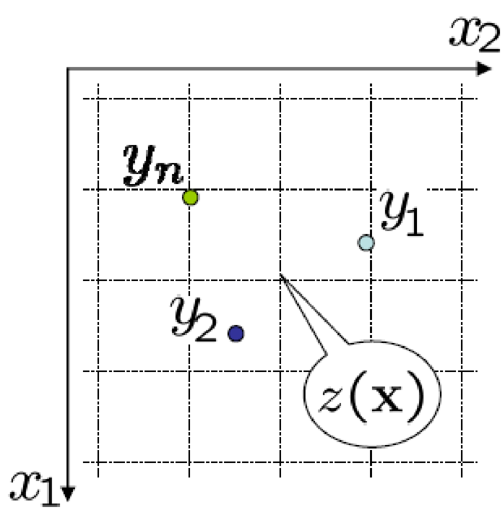
The regression function

The sample position

The number of samples

- For 2-D image

$$x_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \end{bmatrix}$$

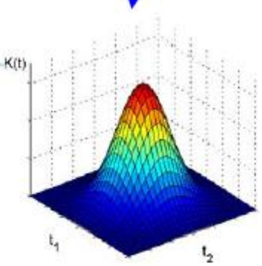


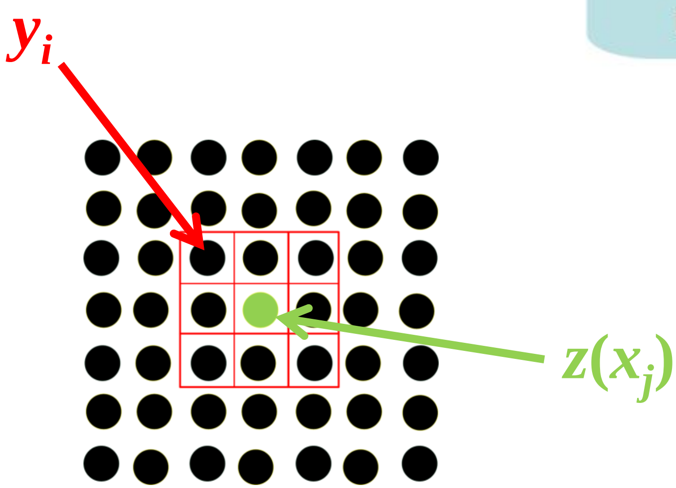
The Common Framework

- The (point) estimate:

$$\hat{z}(x_j) = \arg\min_{z(x_j)} \sum_{i=1}^{n} [y_i - z(x_j)]^2 \ K(x_i, x_j, y_i, y_j)$$

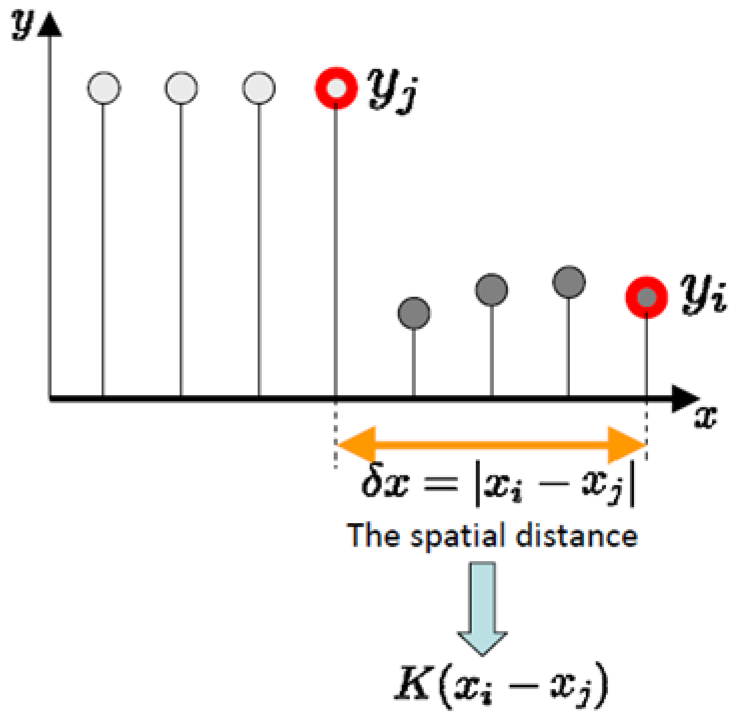
Measure of similarity between two data points i and j.

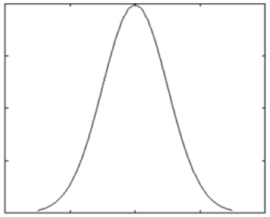




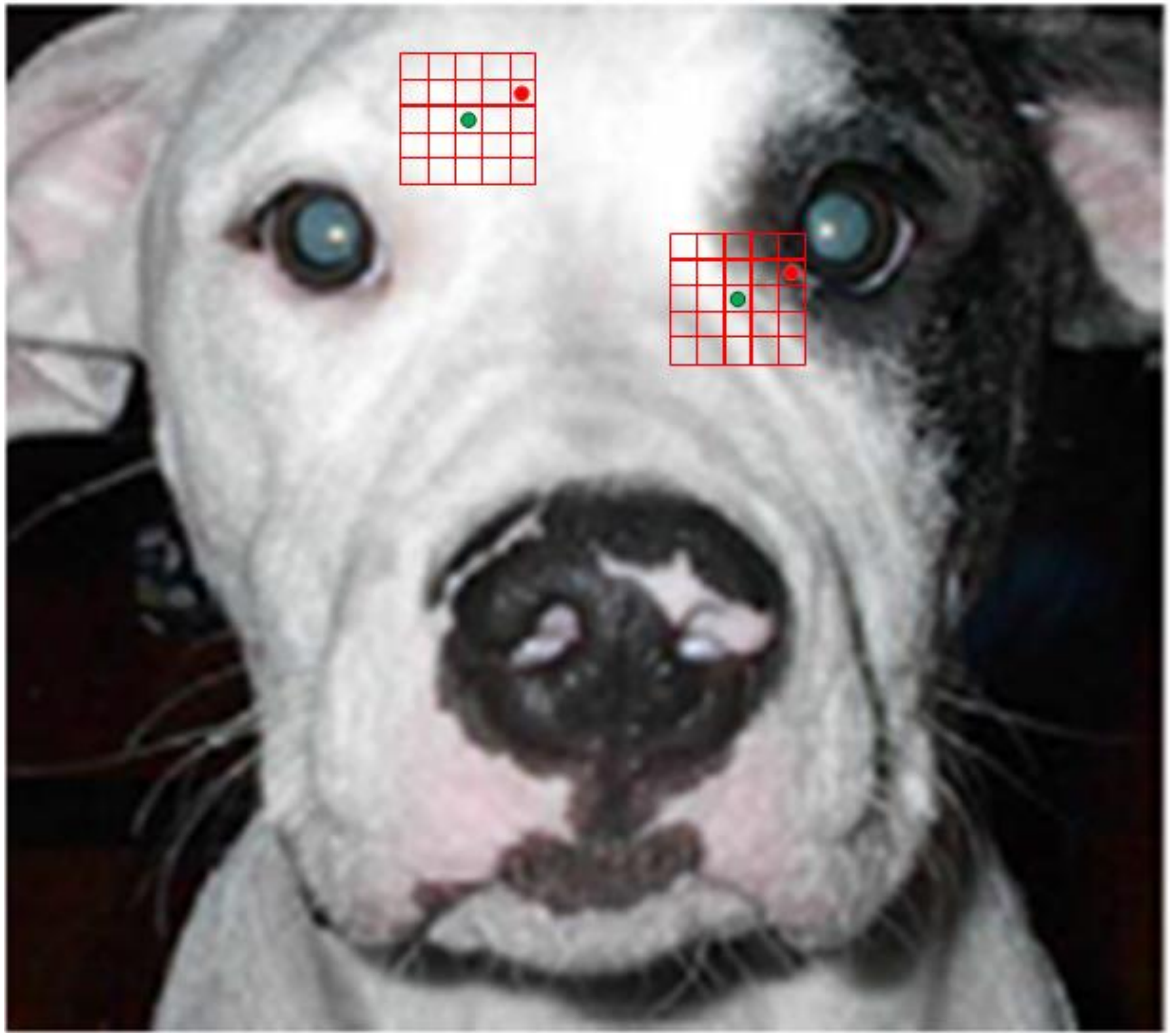
Classical Gaussian Linear Filter

$$K(x_i, x_j, y_i, y_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{h_x^2}\right)$$





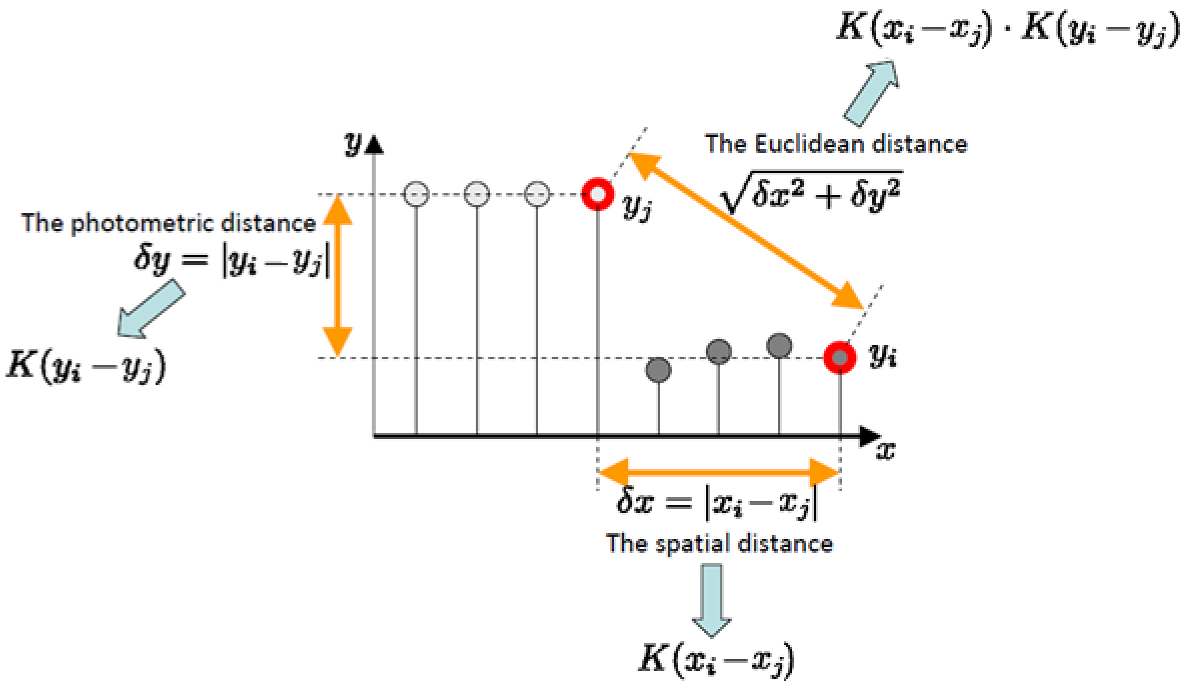
Classical Gaussian Linear Filter



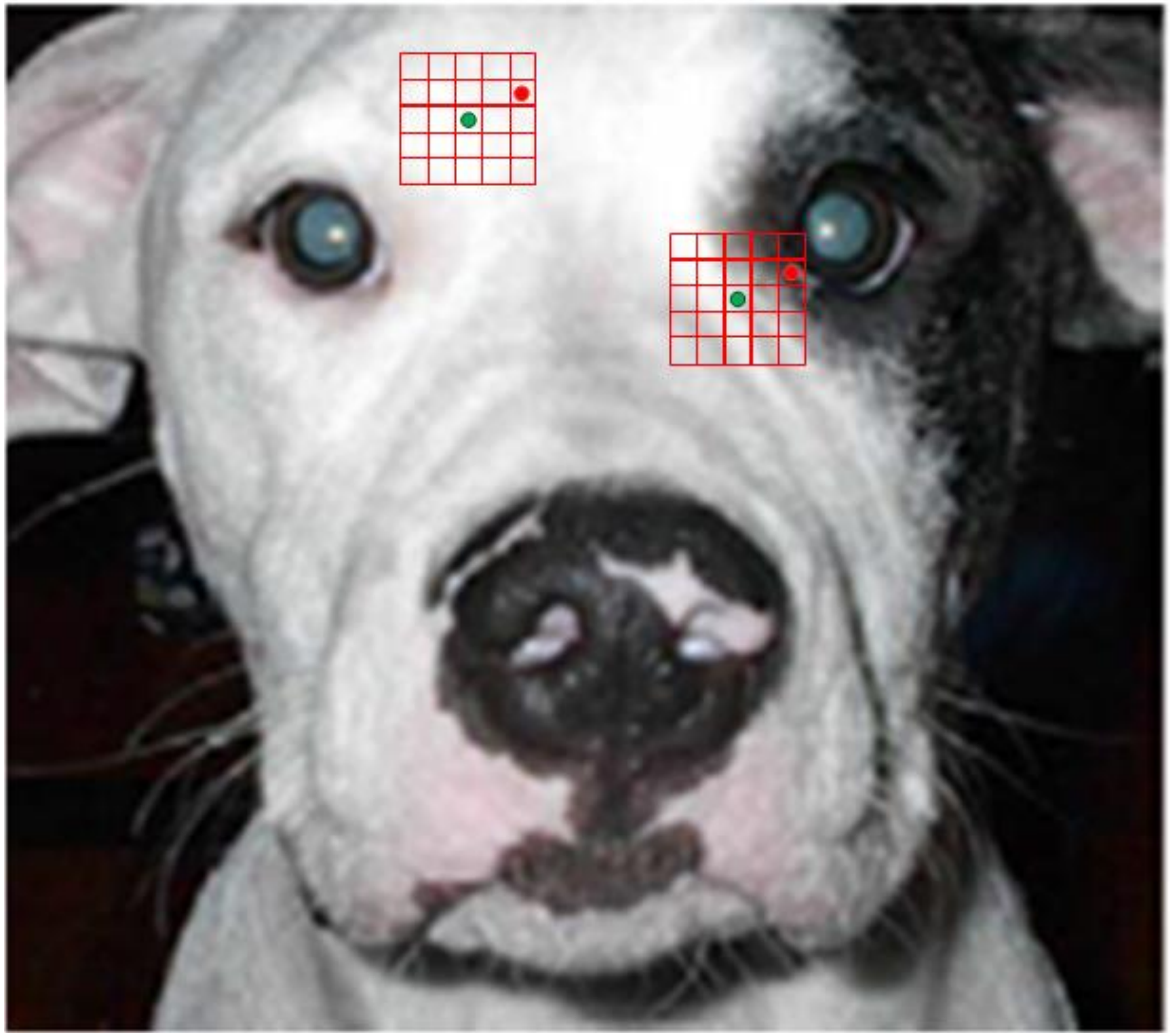
汪汪

Bilateral Filter

$$K(x_i, x_j, y_i, y_j) = \exp\left\{\frac{-\|x_i - x_j\|^2}{h_x^2} + \frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y^2}\right\}$$



Bilateral Filter



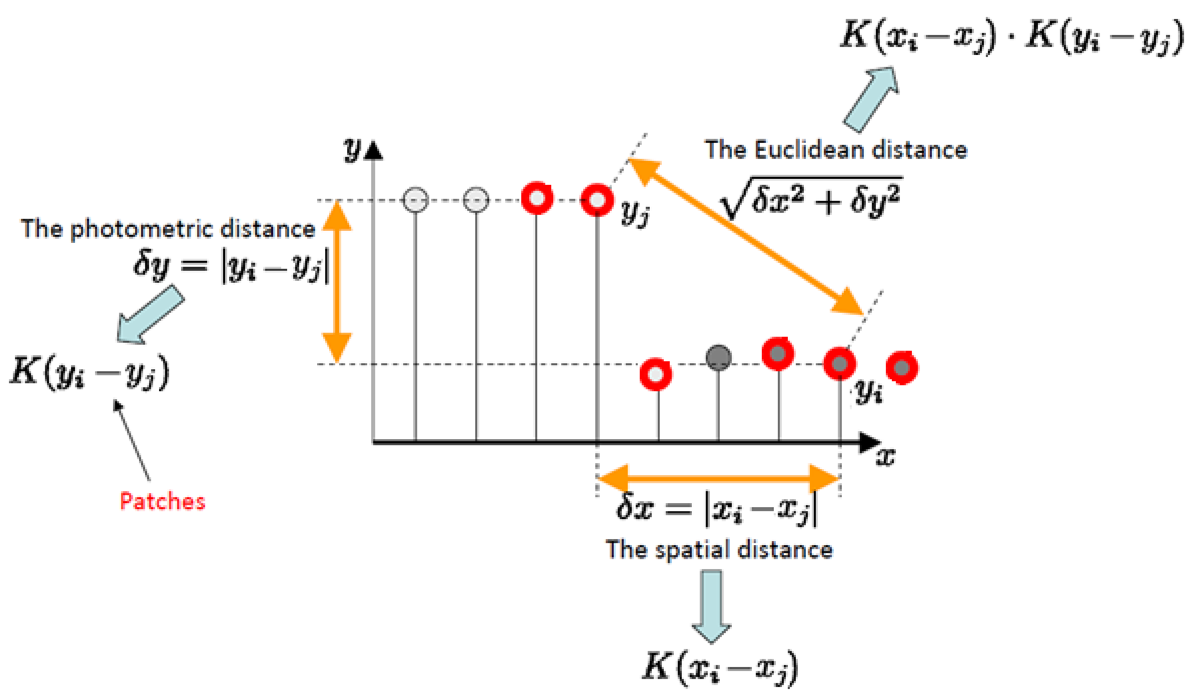
汪汪

Non-local Means

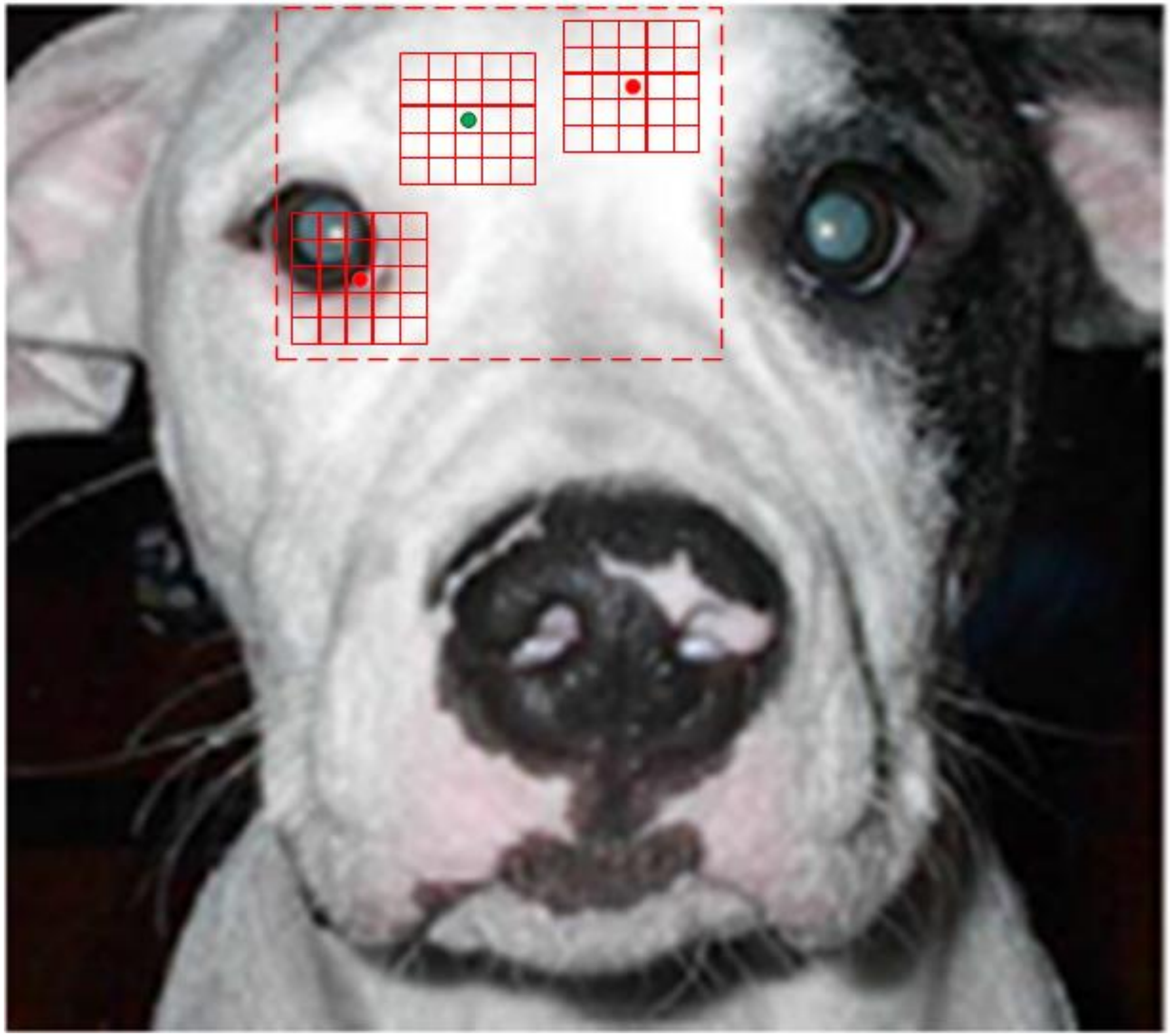
Patches

$$K(x_i, x_j, y_i, y_j) = \exp\left\{\frac{-\|x_i - x_j\|^2}{h_x^2} + \frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y^2}\right\}$$

∞



Non-local Means



汪汪

Locally Adaptive Regression Kernels (LARK)

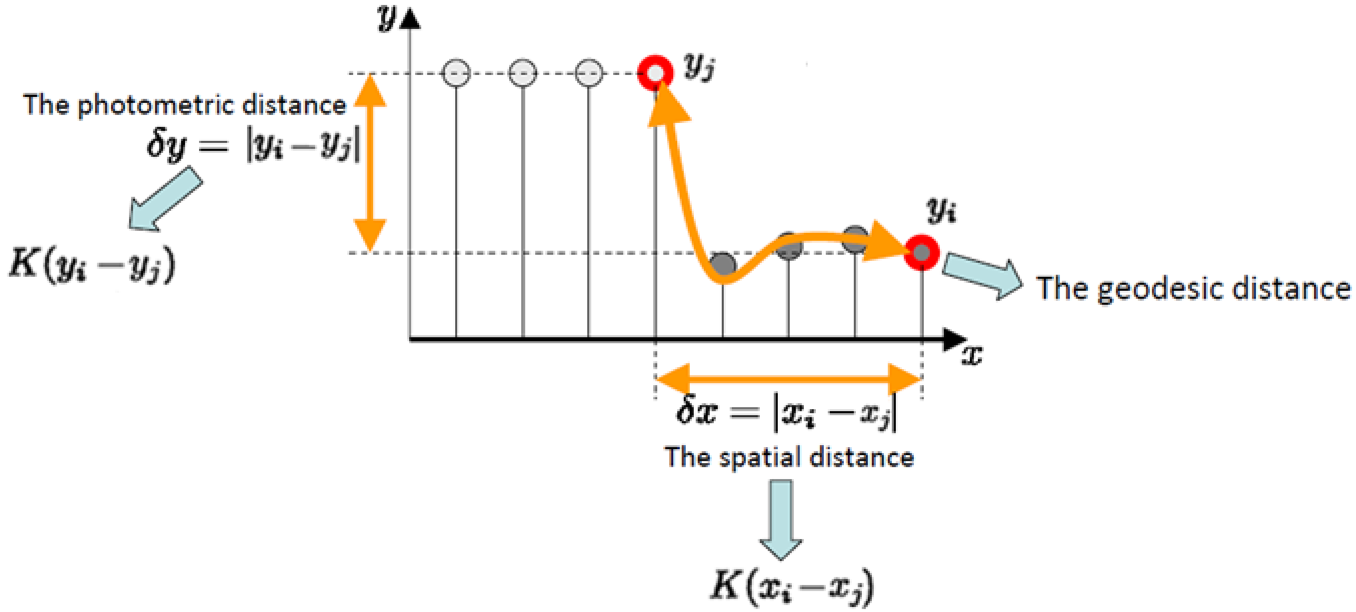
"Learned" (Geodesic) Distance Metric

$$K(x_i, x_j, y_i, y_j) = \exp\left\{-\boxed{(x_i - x_j)^T \widehat{\mathbf{C}}_{ij} (x_i - x_j)}\right\}$$

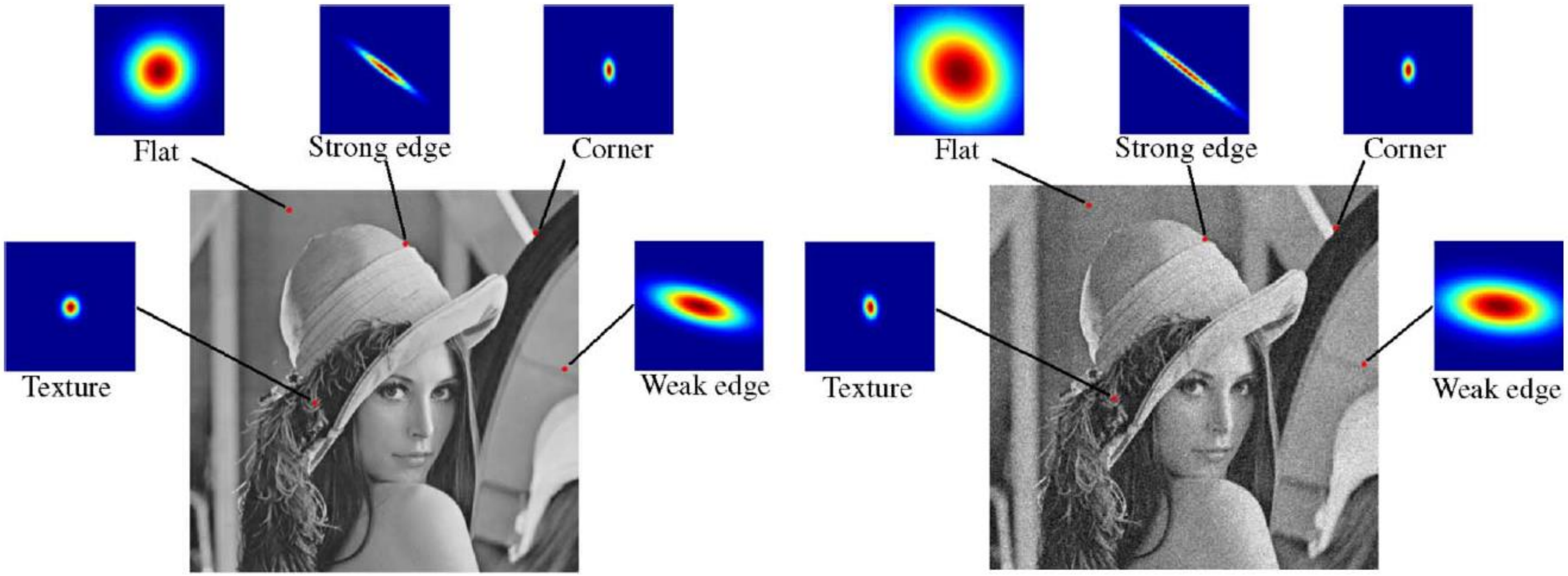
Estimated Local gradient covariance

- "Structure Tensor"
- "Metric Tensor"

$$\widehat{\mathbf{C}}_{ij} = \sum_j \begin{bmatrix} \widehat{z}_{i,1}^2(x_j) & \widehat{z}_{i,1}(x_j)\widehat{z}_{i,2}(x_j) \\ \widehat{z}_{i,1}(x_j)\widehat{z}_{i,2}(x_j) & \widehat{z}_{i,2}^2(x_j) \end{bmatrix}$$

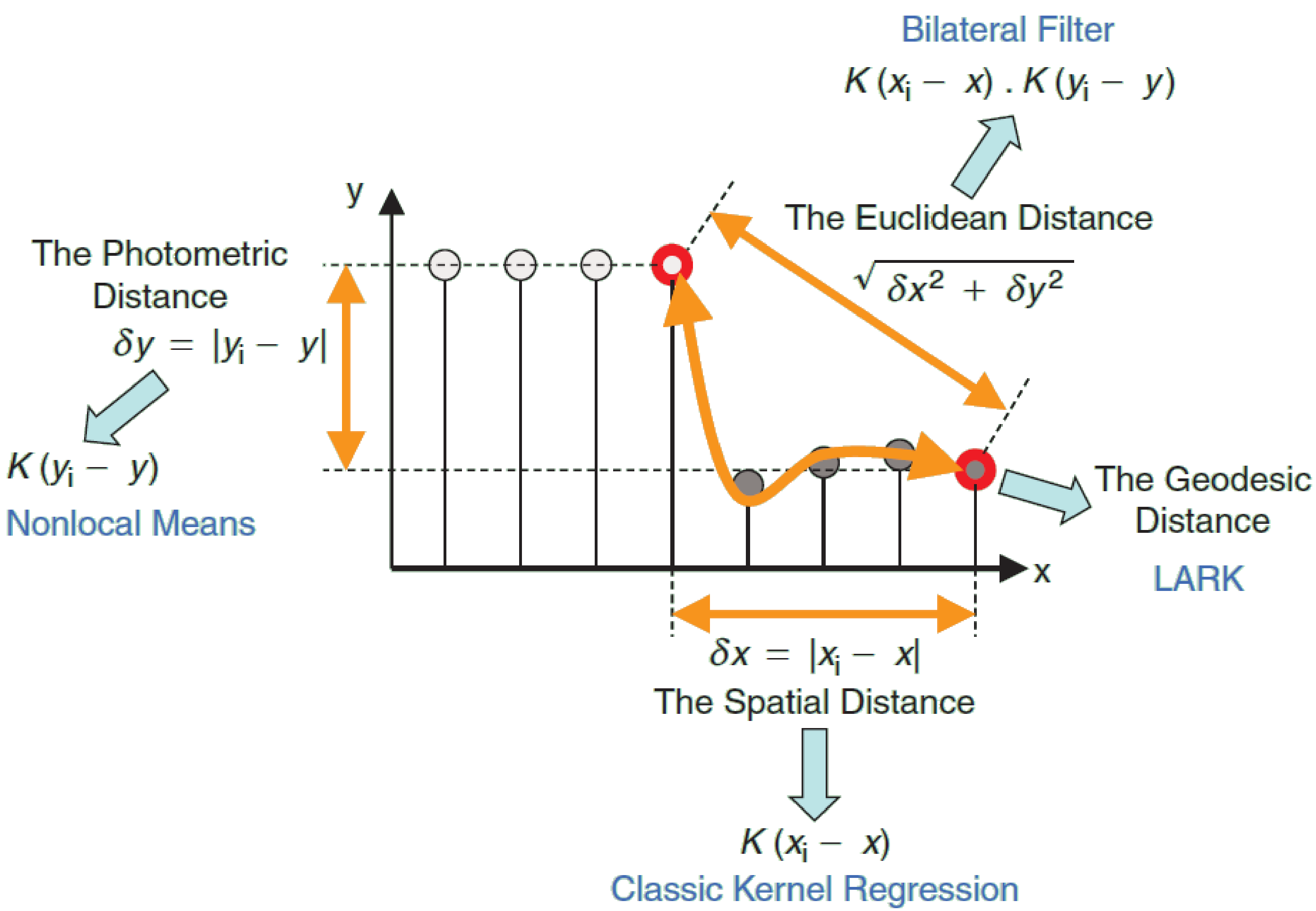


Locally Adaptive Regression (Steering) Kernels



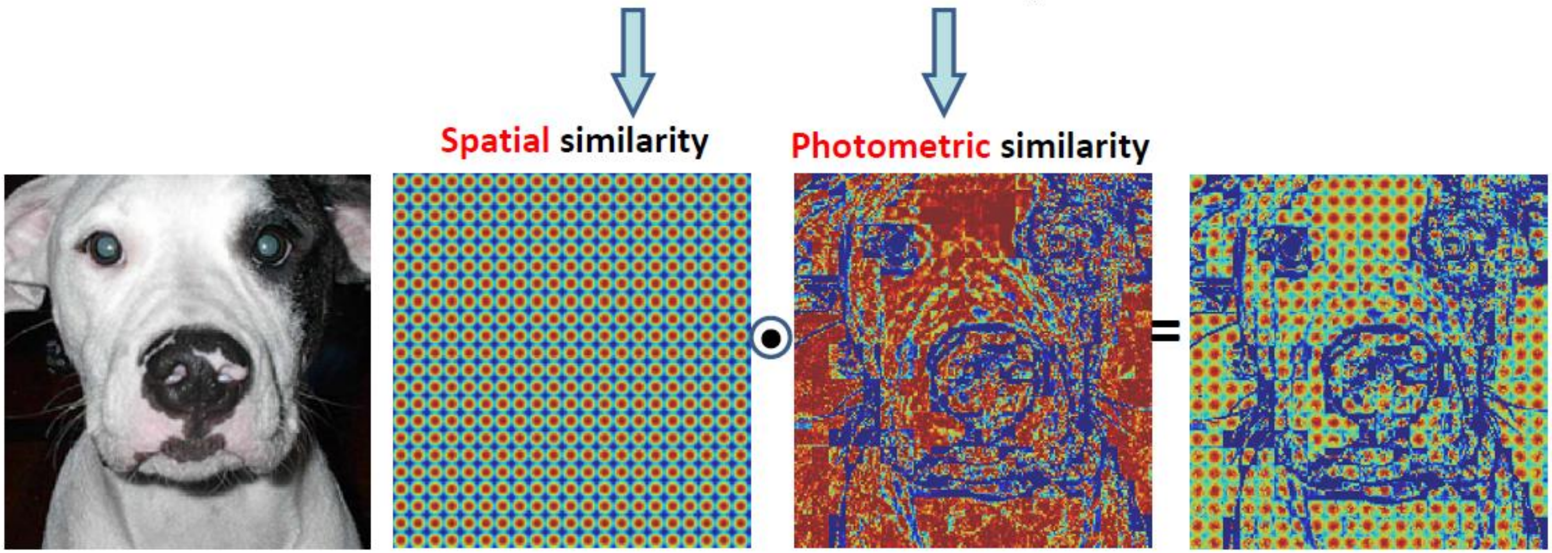
(a) Noiseless case (b) Noisy case (SNR = 5.64[dB])

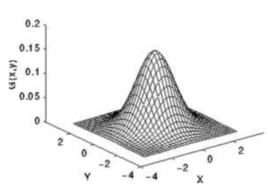
Some Special Cases



Kernels: Bilateral filter

$$K(x_i, x_j, y_i, y_j) = \exp\left\{ \frac{-\|x_i - x_j\|^2}{h_x^2} + \frac{-(y_i - y_j)^2}{h_y^2} \right\}$$





Shown in non-overlapping patches
(for convenience of illustration only)

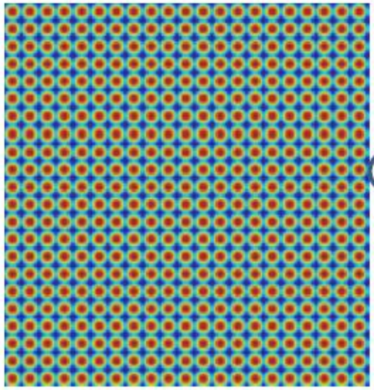
Kernels: Non-local means

Patches

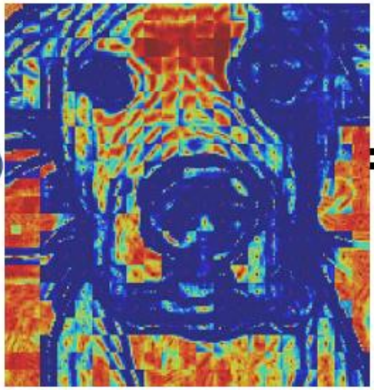
$$K(x_i, x_j, y_i, y_j) = \exp\left\{\frac{-\|x_i - x_j\|^2}{h_x^2} + \frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{h_y^2}\right\}$$

∞

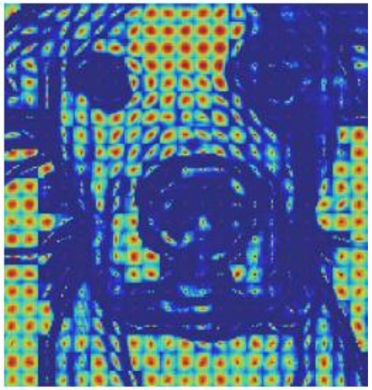
Spatial similarity | **Photometric similarity**

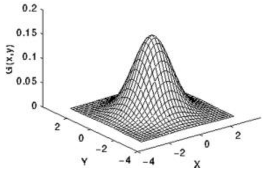


$\odot$



=





Shown in non-overlapping patches
(for convenience of illustration only)

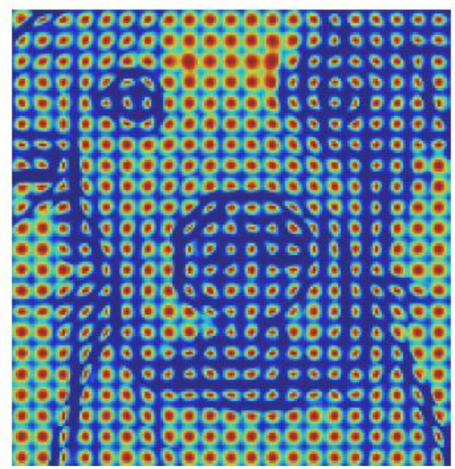
Kernels: LARK

"Learned" (Geodesic) Distance Metric

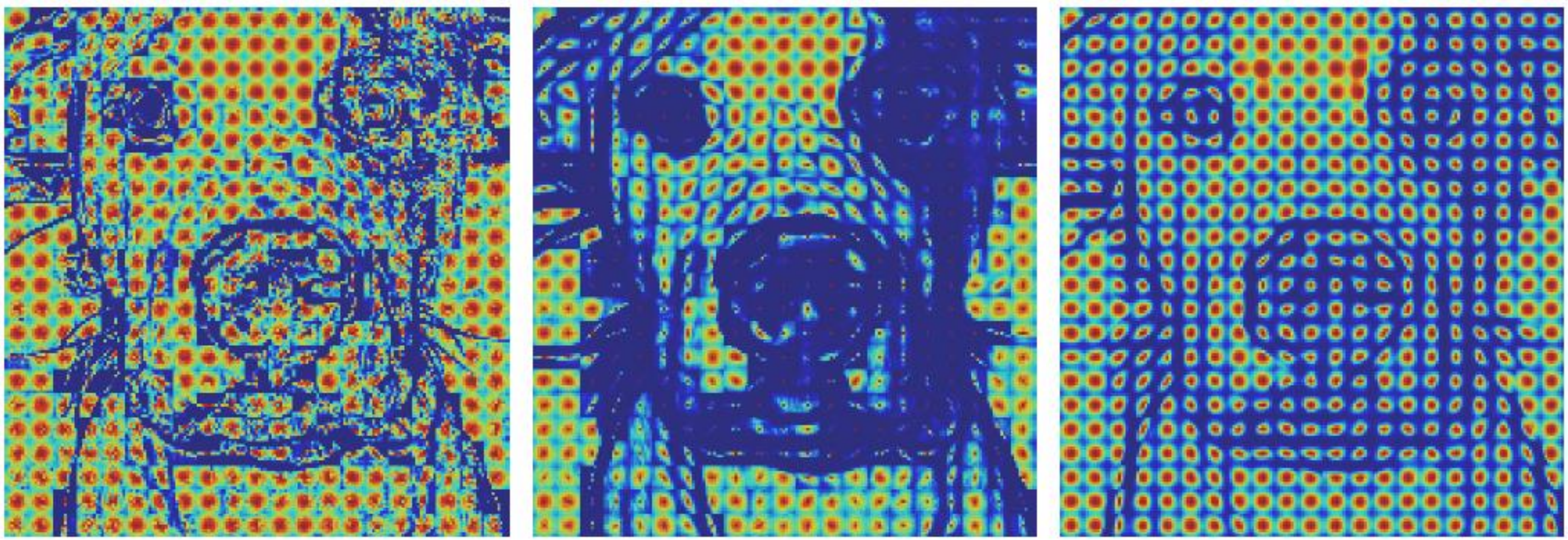
$$K(x_i, x_j, y_i, y_j) = \exp\left\{-\boxed{(x_i - x_j)^T \widehat{\mathbf{C}}_{ij}(x_i - x_j)}\right\}$$

Estimated Local gradient covariance
- "Structure Tensor"
- "Metric Tensor"

$$\widehat{\mathbf{C}}_{ij} = \sum_j \begin{bmatrix} \widehat{z}^2_{i,1}(x_j) & \widehat{z}_{i,1}(x_j)\widehat{z}_{i,2}(x_j) \\ \widehat{z}_{i,1}(x_j)\widehat{z}_{i,2}(x_j) & \widehat{z}^2_{i,2}(x_j) \end{bmatrix}$$



Comparisons



Bilateral

Non-local Means

LARK

Generalizations (1/2)

- General Gaussian Kernel with $\mathbf{t} = \begin{bmatrix} x \\ \mathbf{y} \end{bmatrix}$

$$
\begin{aligned}
K(\mathbf{t}_i, \mathbf{t}_j) &= \exp\left\{-(\mathbf{t}_i - \mathbf{t}_j)^T \mathbf{Q}_{i,j} (\mathbf{t}_i - \mathbf{t}_j)\right\} \\
\mathbf{Q}_{i,j} &= \begin{bmatrix} \mathbf{Q}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_y \end{bmatrix}
\end{aligned}
$$

Symmetric, positive-definite

- Classical: $\mathbf{Q}_x = \frac{1}{h_x^2}\mathbf{I}$ and $\mathbf{Q}_\mathbf{y} = \mathbf{0}$
- Bilateral: $\mathbf{Q}_x = \frac{1}{h_x^2}\mathbf{I}$ and $\mathbf{Q}_\mathbf{y} = \frac{1}{h_y^2}\mathrm{diag}[0, 0, \cdots, 1, \cdots, 0, 0]$
- Non-local Means: $\mathbf{Q}_\mathbf{x} = \mathbf{0}$ and $\mathbf{Q}_\mathbf{y} = \frac{1}{h_y^2}\mathbf{G}$
- LARK: $\mathbf{Q}_x = \mathbf{C}_{ij}$ and $\mathbf{Q}_\mathbf{y} = \mathbf{0}$.

Generalizations (2/2)

$$
\begin{aligned}
K(\mathbf{t}_i, \mathbf{t}_j) &= \exp\left\{-(\mathbf{t}_i - \mathbf{t}_j)^T \mathbf{Q}_{i,j} (\mathbf{t}_i - \mathbf{t}_j)\right\} \\
\mathbf{Q}_{i,j} &= \begin{bmatrix} \mathbf{Q}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_y \end{bmatrix}
\end{aligned}
$$

Symmetric, positive-definite

- Introduce off-diagonal blocks for Q
- Define the "feature" vector **t** more generally
- Reproducing Kernels

Simulations

Original image

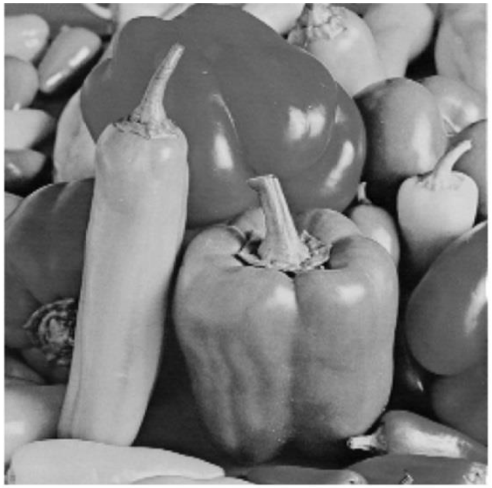
Noisy image

Bilateral filtering

LARK

Simulations

Original image



Compressed image by JPEG with quality of 10

Bilateral filtering

LARK

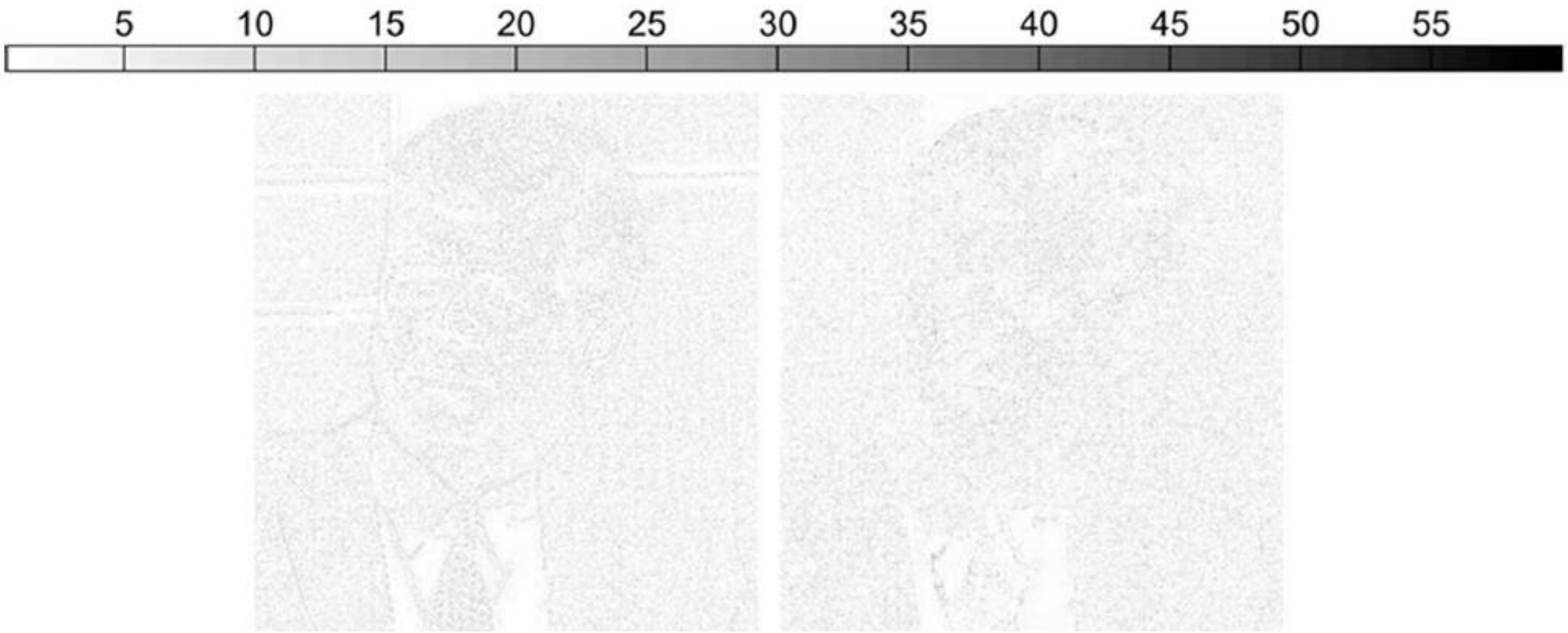
Simulations

Noisy image

Bilateral filtering

LARK

Simulations



Bilateral filtering **LARK**

Simulations

- Irregularly sampled data interpolation (85% of the pixels are omitted)



Reference

- Some papers
- Web

Thank you for your attention