

Image Filtering

Computer Vision: CS 566

Computer Science

University of Wisconsin-Madison

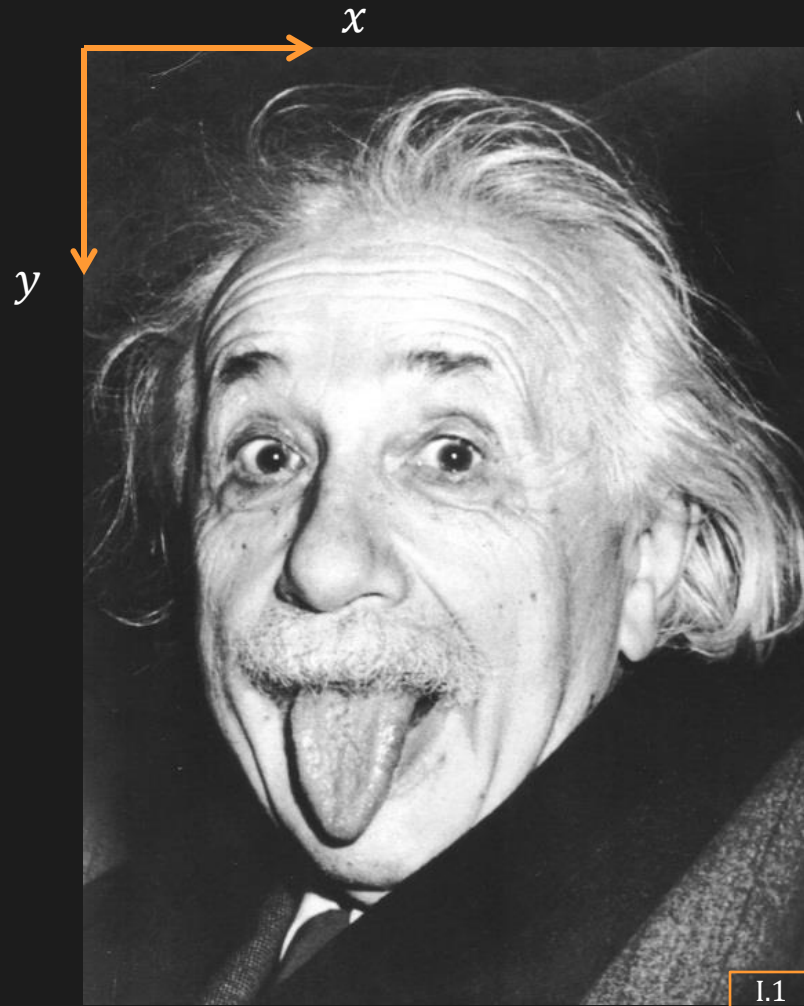
Image Processing

Transform image to new one that is easier to manipulate/analyze.

Topics:

- (1) Pixel Processing
- (2) Linear Filtering
- (3) Non-Linear Filtering
- (4) Correlation

Image as a Function



$f(x, y)$ is the image intensity at position (x, y)

Digital Images

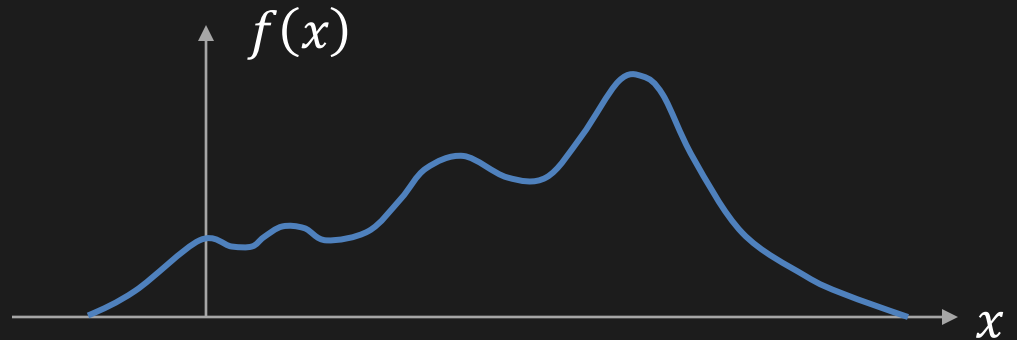
1. Sample the
2D Space on a
Discrete Grid



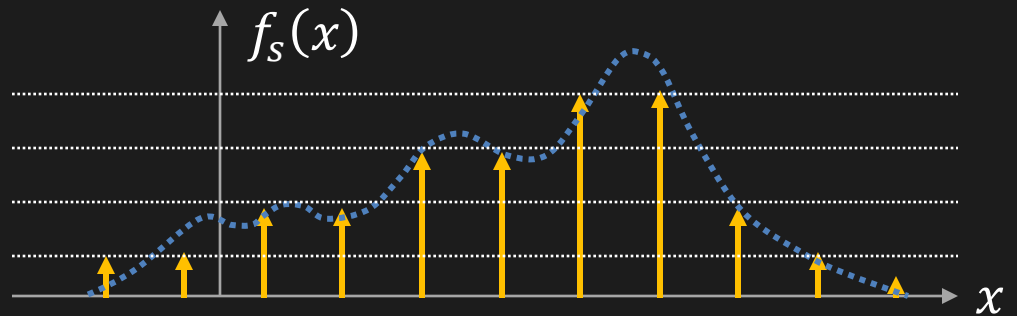
2. Quantize Each
Sample (Round to
Nearest Integer)

1D Example

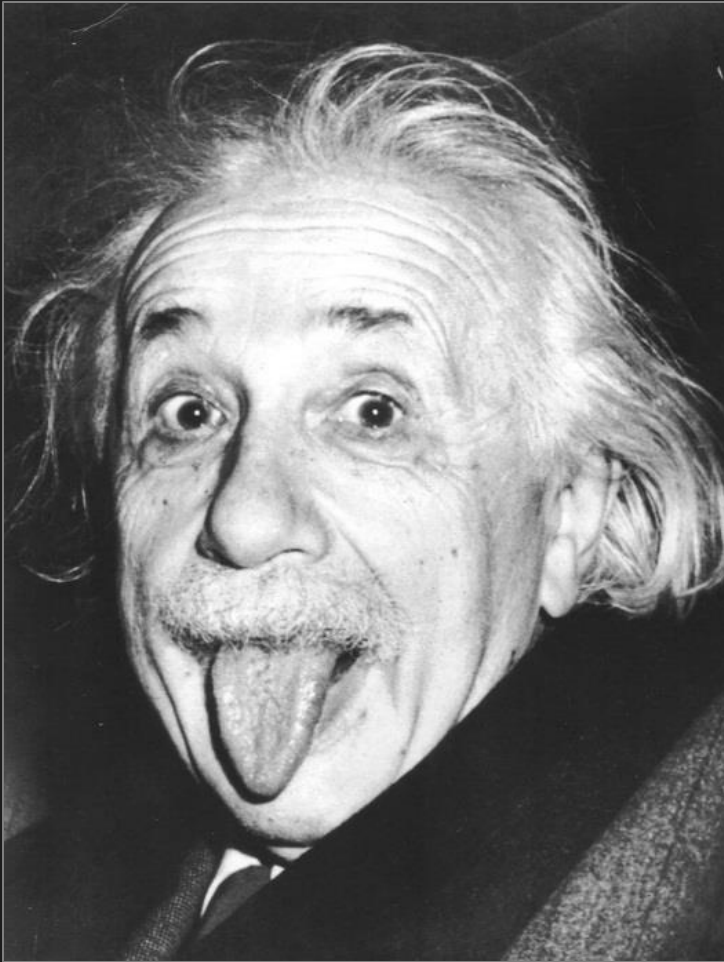
Continuous Signal:



Digital Signal:



Digital Images as Matrices



157	159	159	104	104	115	128	131	133	133	132	131
159	165	153	101	103	113	126	129	130	130	126	124
162	154	154	98	101	114	124	127	130	132	144	159
141	132	158	93	98	110	121	125	122	129	143	172
100	130	157	93	99	110	120	116	116	129	138	163
87	130	157	92	97	109	124	111	123	134	139	175
93	131	159	92	98	112	132	108	123	133	162	180
96	134	164	95	97	113	147	108	125	142	156	171
95	137	165	95	95	111	168	122	130	137	145	139
101	139	166	94	96	104	172	130	126	130	108	77
101	133	167	94	96	100	154	137	123	92	67	57
99	130	169	97	99	109	131	128	84	55	60	75
97	129	170	97	98	118	122	94	66	56	56	140
92	123	173	101	98	129	95	74	74	45	94	174
81	115	175	104	116	87	78	69	84	56	140	124
69	108	172	107	103	87	82	54	83	105	93	107
71	119	172	106	91	78	97	70	99	104	59	116
61	126	175	112	83	74	92	123	130	53	61	108
53	128	175	105	71	82	109	127	75	50	57	74
56	115	173	105	61	76	106	114	70	54	52	60
117	106	176	101	55	71	81	112	101	57	55	70
107	121	177	89	50	64	60	103	114	66	56	90

Image Processing

Transformation T of one image f to another image g

$$g(u, v) = T(f(x, y))$$

Point (Pixel) Processing



Darken ($f - 128$)



Original (f)

1.2



Lighten ($f + 128$)



Invert ($255 - f$)

Point (Pixel) Processing



Low Contrast ($f/2$)



Original (f)



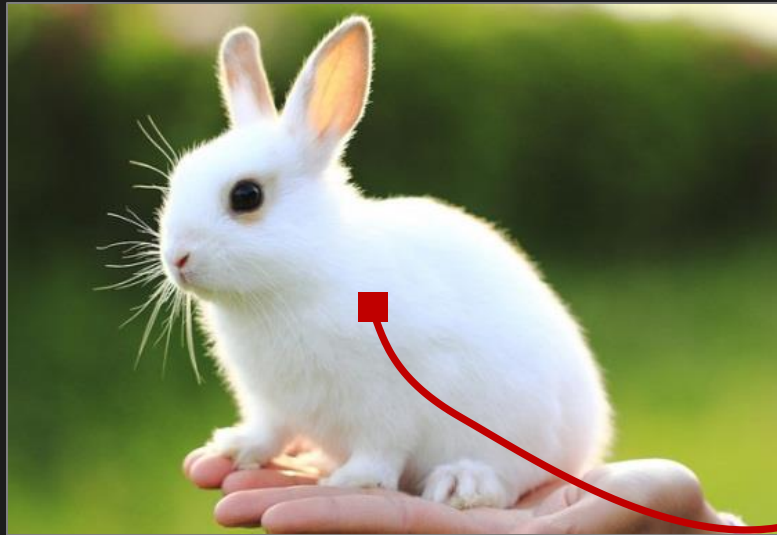
High Contrast ($f * 2$)



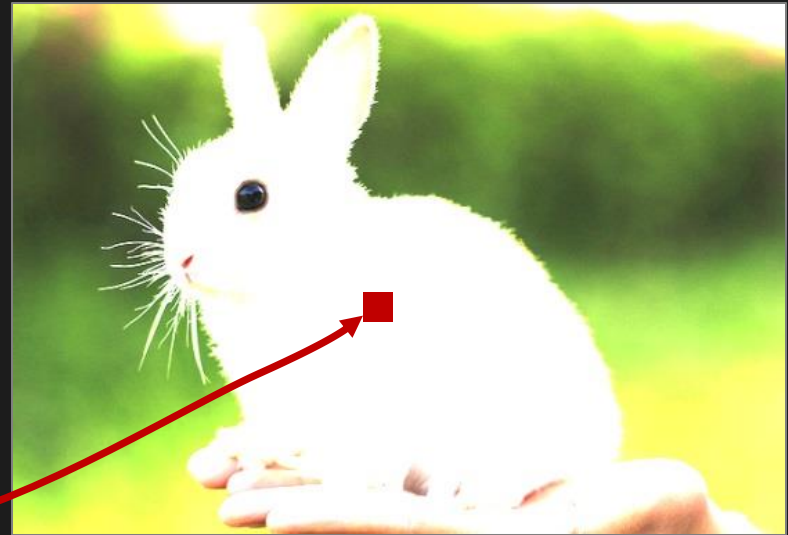
Gray ($0.3f_R + 0.6f_G + 0.1f_B$)

Point (Pixel) Processing

original image



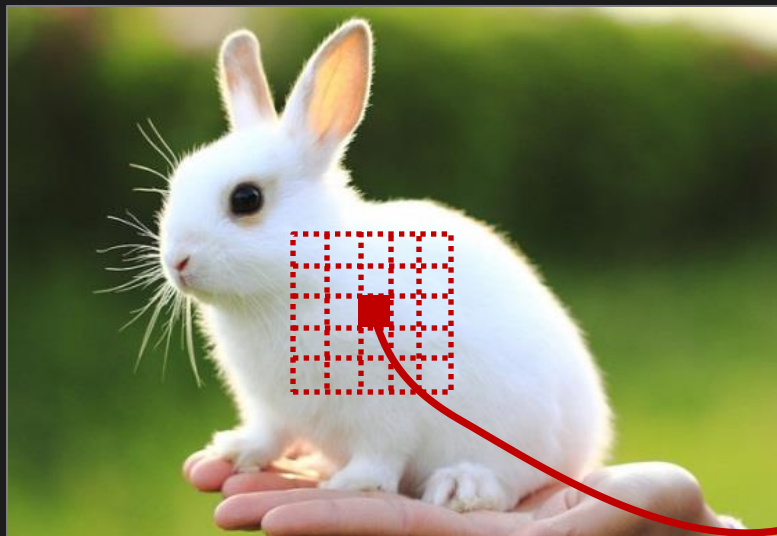
processed image



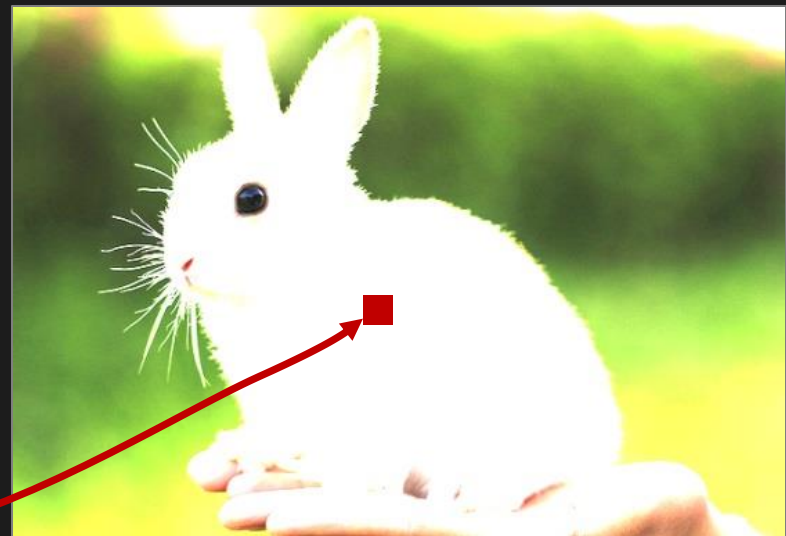
Pixel value in the processed image depends **ONLY** on the same pixel location in the original image

Image Filtering

original image



processed image



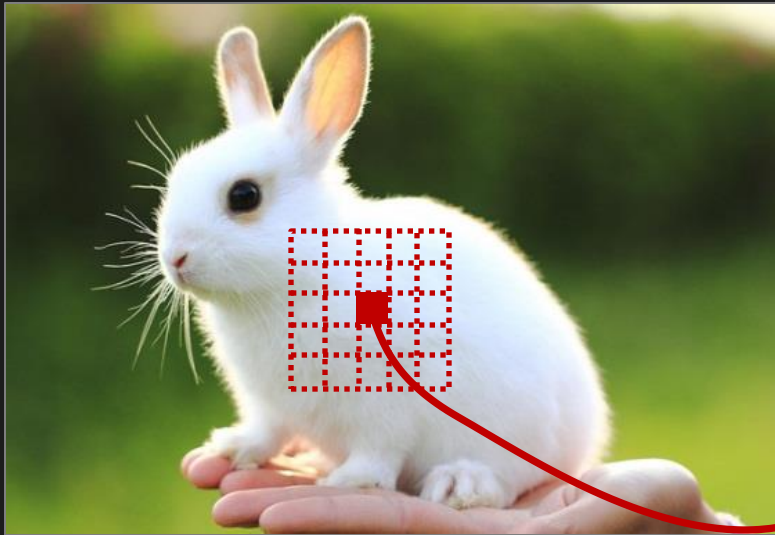
Pixel value in the processed image depends
on a local neighborhood in the original image

Image Filtering: Applications

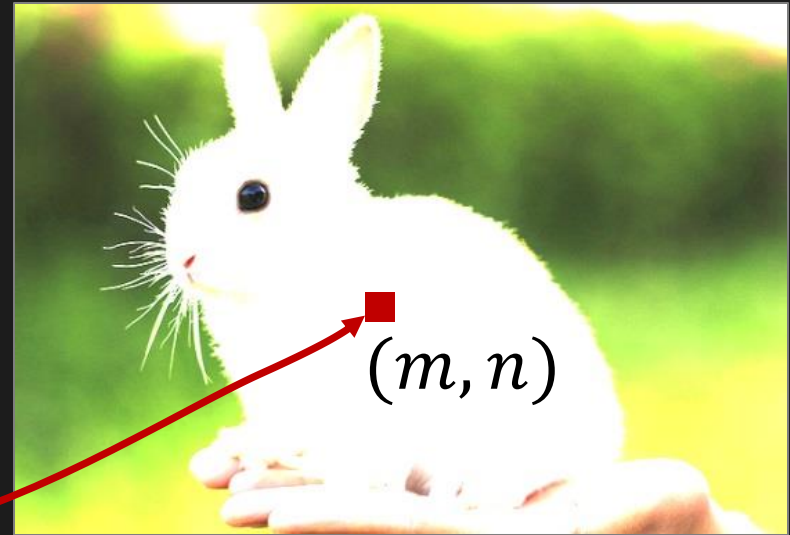
- Enhance Images (denoise, sharpen, resize, ...)
- Extract Information (edges, features, texture, ...)
- Detect Patterns (template matching)

Linear Image Filtering

original image (I)



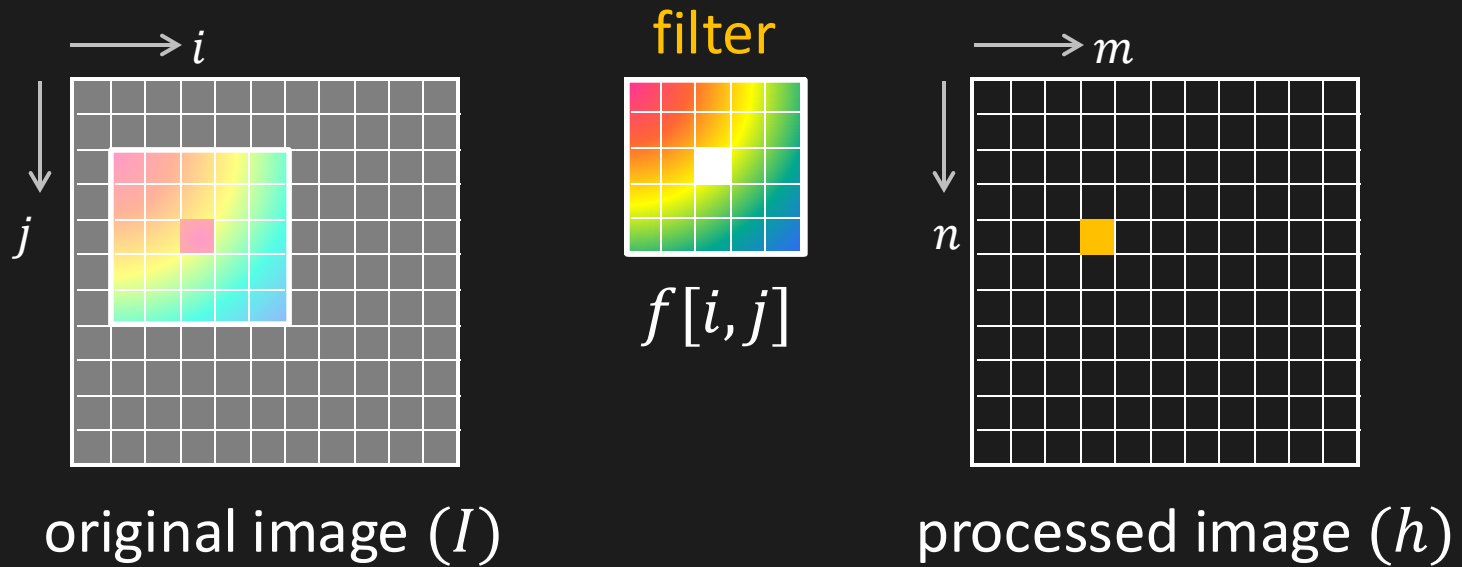
processed image (h)



$f = \text{filter}$

$$h[m, n] = \sum_k \sum_l f[k, l] I[m + k, n + l]$$

Linear Image Filtering



$$h[m, n] = \sum_k \sum_l f[k, l] I[m + k, n + l]$$

Example: Box Filter

1	1	1
1	1	1
1	1	1

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

$f[i, j]$

1	1	1
1	1	1
1	1	1

 $\times \frac{1}{9}$
 $f[i,j]$

Example: Box Filter

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

processed image (h)

1	1	1
1	1	1
1	1	1

 $\times \frac{1}{9}$
 $f[i,j]$

Example: Box Filter

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

	0	10							

processed image (h)

1	1	1
1	1	1
1	1	1

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

Example: Box Filter

$f[i,j]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

	0	10	20						

processed image (h)

1	1	1
1	1	1
1	1	1

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

Example: Box Filter

$f[i,j]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

	0	10	20	30					

processed image (h)

1	1	1
1	1	1
1	1	1

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

Example: Box Filter

$f[i,j]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

	0	10	20	30	30				

processed image (h)

1	1	1
1	1	1
1	1	1

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

Example: Box Filter

$f[i,j]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

	0	10	20	30	30				

processed image (h)

1	1	1
1	1	1
1	1	1

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

Example: Box Filter

$f[i,j]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

	0	10	20	30	30				
						?			
				50					

processed image (h)

1	1	1
1	1	1
1	1	1

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

Example: Box Filter

$f[i,j]$

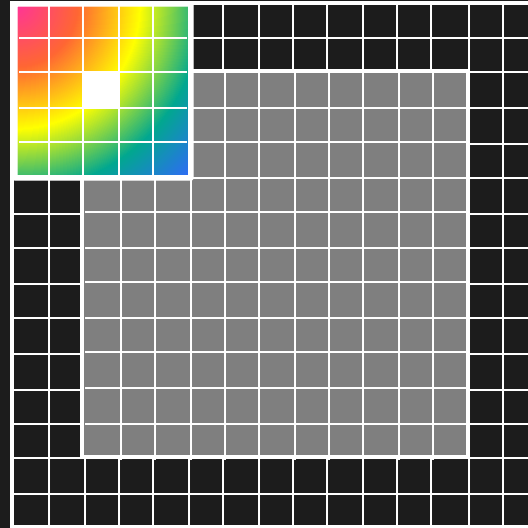
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

original image (I)

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

processed image (h)

Border Problem



Solution:

- Ignore Border
- Pad with Constant Value
- Pad with Reflection

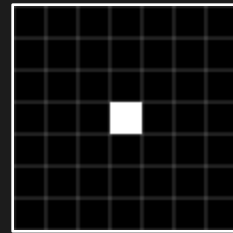
Example: Impulse Filter

Input



$I[i, j]$

*



$f[k, l]$

=

Output



$h[m, n]$

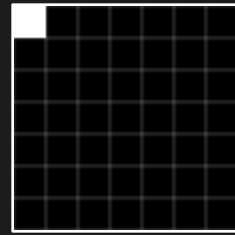
Example: Image Shift

Input



$I[i, j]$

*



$f[k, l]$

=

Output



$h[m, n]$

Example: Averaging

Input

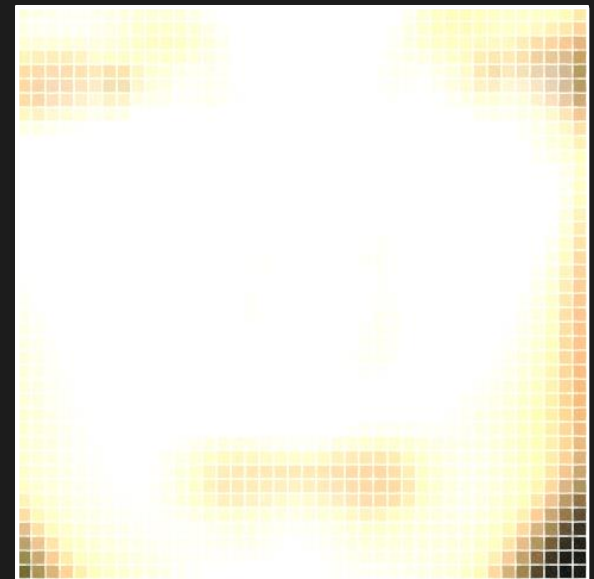


$I[i, j]$

“Box Filter”

$$* \begin{array}{c} \text{5x5 grid with a white center} \\ f[k, l] \\ 5 \times 5 \end{array} =$$

Output



$h[m, n]$

Result Image is saturated. Why?

Example: Averaging

Input

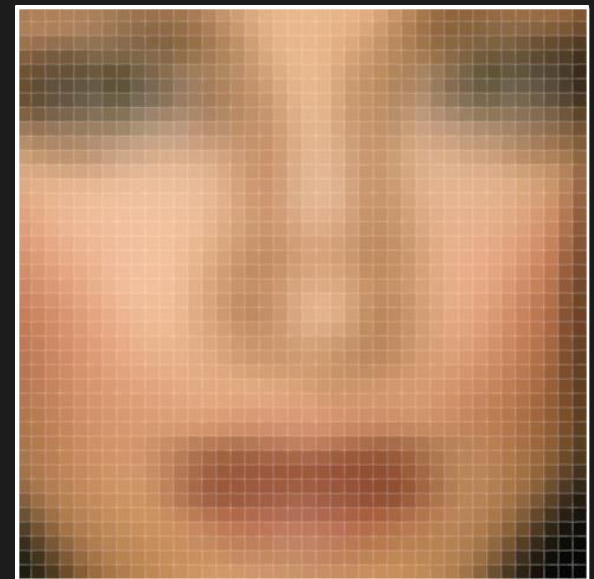


$I[i, j]$

“Box Filter”

$$* \begin{array}{c} \text{5x5 grid with a gray square in the center} \\ f[k, l] \\ \text{5} \times \text{5} \end{array} =$$

Output



$h[m, n]$

Sum of all the filter (kernel) weights should be 1.

Smoothing With Box Filter

Input

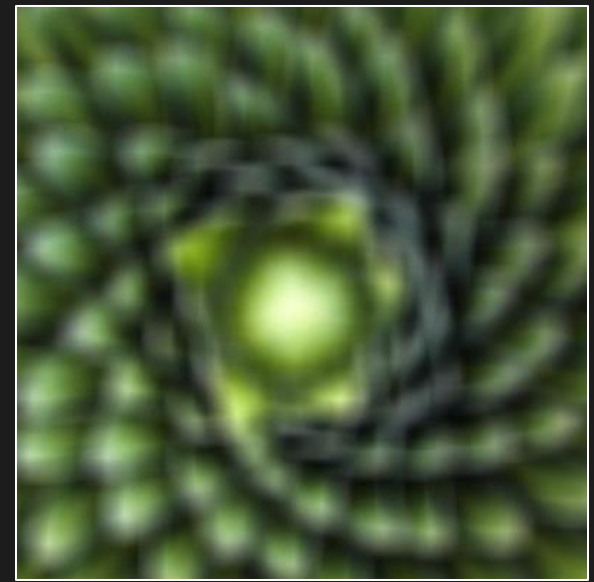


$I[i, j]$

“Box Filter”

$$* \begin{array}{c} \boxed{\text{Box Filter}} \\ f[k, l] \\ 21 \times 21 \end{array} =$$

Output



$h[m, n]$

Image smoothed with a box filter does not look “natural.”
Has blocky artifacts.

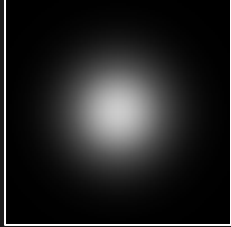
Smoothing With “Fuzzy” Filter

Input



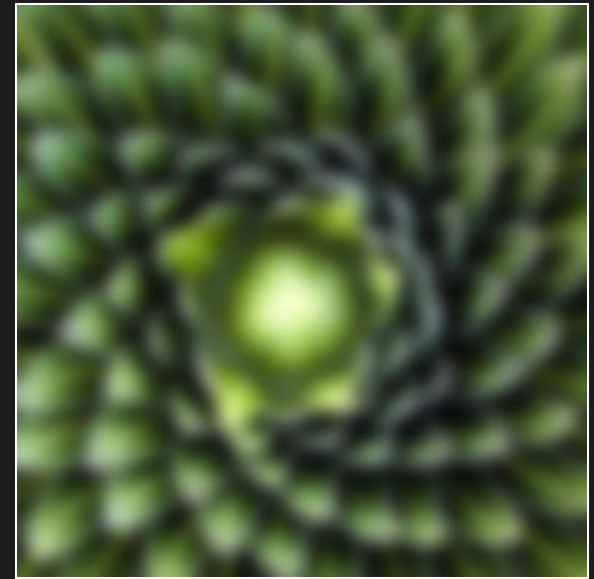
$I[i, j]$

“Fuzzy Filter”

$*$  $=$

$f[k, l]$
 21×21

Output

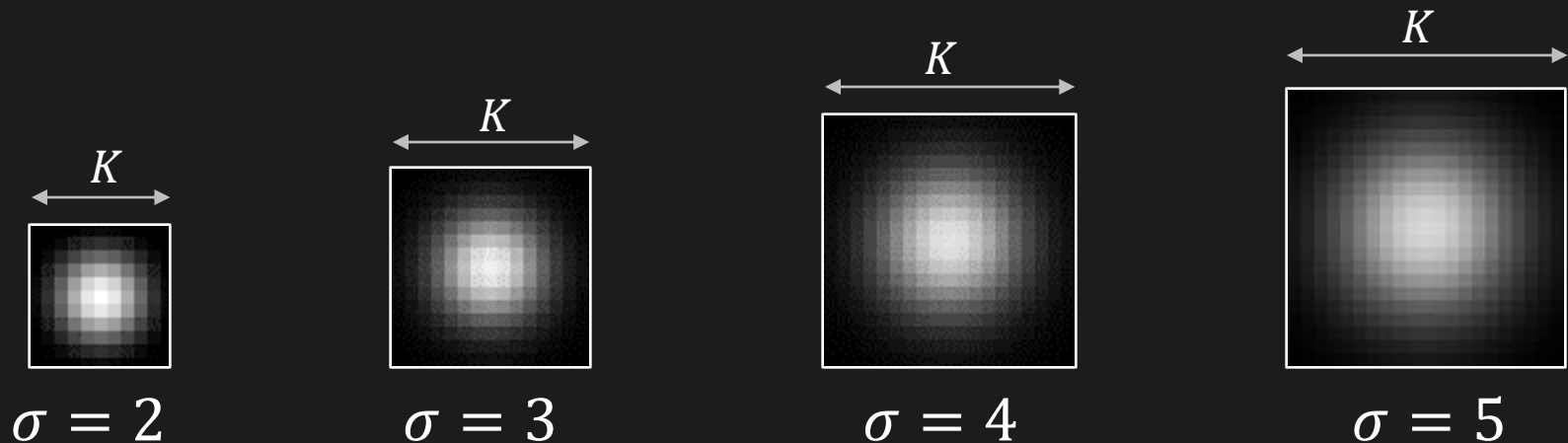
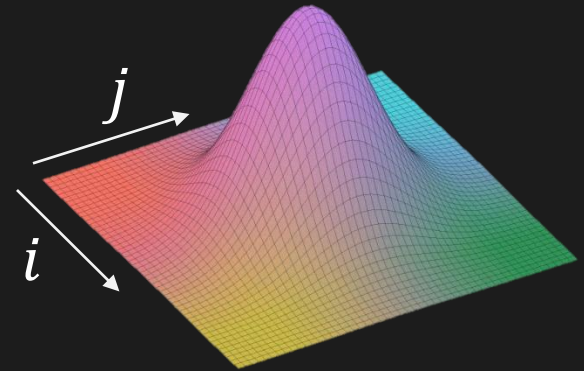


$h[m, n]$

Gaussian Kernel: A Fuzzy Filter

$$n_{\sigma}[i, j] = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$$

σ^2 : Variance



Rule of thumb: Set kernel size $K \approx 2\pi\sigma$

Gaussian Smoothing

Input

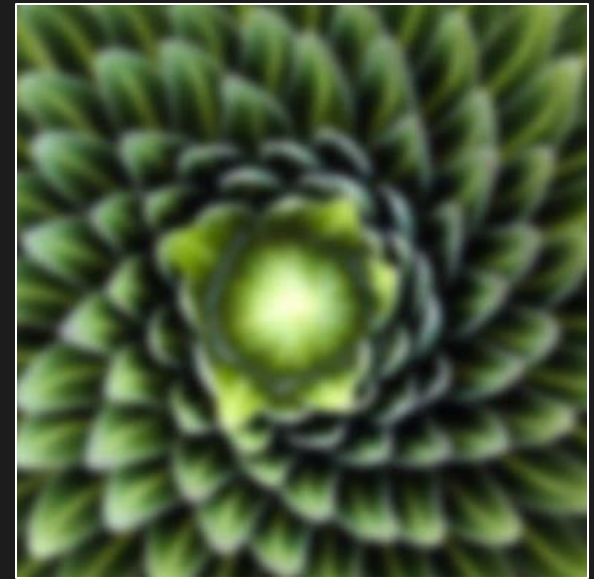


$I[i, j]$

“Fuzzy Filter”

$$* \begin{array}{c} \text{[Gaussian Kernel Image]} \\ f_4[k, l] \\ \sigma = 4 \end{array} =$$

Output



$h[m, n]$

Larger the kernel (or σ), more the blurring

Gaussian Smoothing

Input

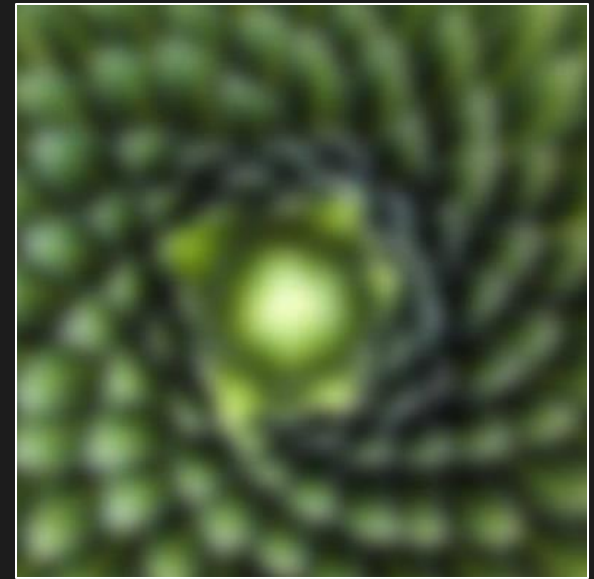


$I[i, j]$

“Fuzzy Filter”

$$* \begin{array}{c} \text{[Gaussian Kernel Image]} \\ f_8[k, l] \\ \sigma = 8 \end{array} =$$

Output



$h[m, n]$

Larger the kernel (or σ), more the blurring

Gaussian Smoothing

Input



$I[i, j]$

“Fuzzy Filter”

$$* \begin{array}{c} \text{[Gaussian Kernel Image]} \\ f_{16}[k, l] \\ \sigma = 16 \end{array} =$$

Output



$h[m, n]$

Larger the kernel (or σ), more the blurring

Gaussian Smoothing is Separable

$$f[i, j] = \frac{1}{2\pi\sigma^2} \sum_{m=1}^K \sum_{n=1}^K e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)} f[i-m, j-n]$$

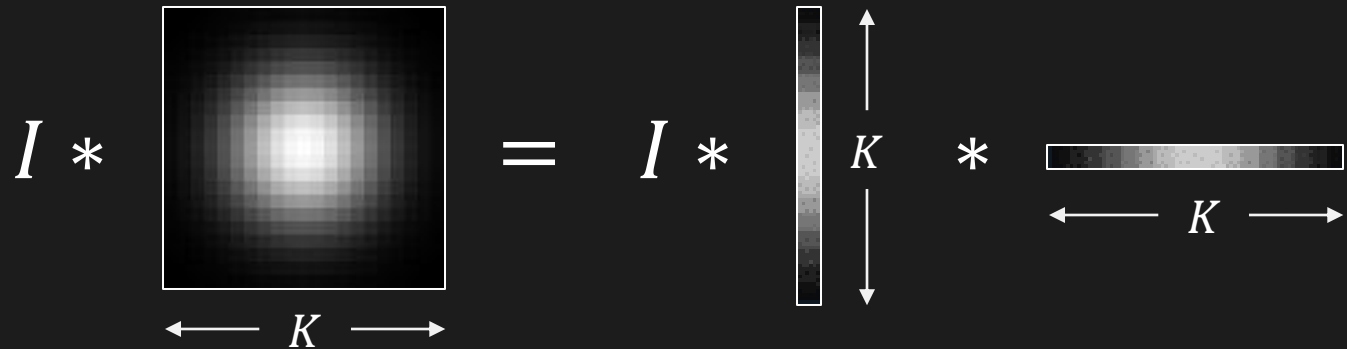
$$f[i, j] = \frac{1}{2\pi\sigma^2} \sum_{m=1}^K e^{-\frac{1}{2}\left(\frac{m^2}{\sigma^2}\right)} \cdot \sum_{n=1}^K e^{-\frac{1}{2}\left(\frac{n^2}{\sigma^2}\right)} f[i-m, j-n]$$

Using One 2D Gaussian Filter \equiv Using Two 1D Gaussian Filters

$$I * \begin{array}{c} \text{2D Gaussian Filter} \\ \leftarrow K \rightarrow \end{array} = I * \begin{array}{c} \text{1D Vertical Filter} \\ \uparrow K \\ \downarrow K \end{array} * \begin{array}{c} \text{1D Horizontal Filter} \\ \leftarrow K \rightarrow \end{array}$$

Gaussian Smoothing is Separable

Using One 2D Gaussian Filter \equiv Using Two 1D Gaussian Filters


$$I * \begin{array}{c} \text{2D Gaussian Filter} \\ \leftarrow K \rightarrow \end{array} = I * \begin{array}{c} \text{1D Gaussian Filter (vertical)} \\ \uparrow K \\ \downarrow K \end{array} * \begin{array}{c} \text{1D Gaussian Filter (horizontal)} \\ \leftarrow K \rightarrow \end{array}$$

Which one is faster? Why?

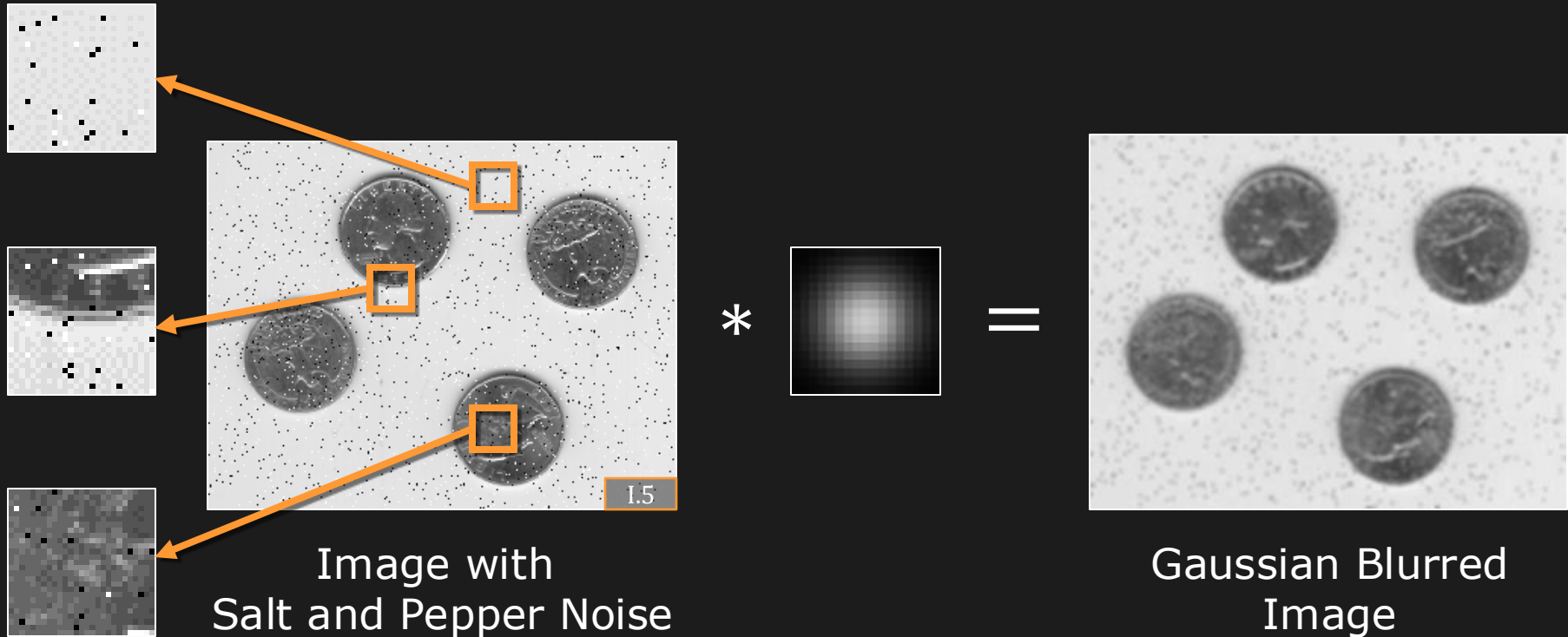
K^2 Multiplications

$K^2 - 1$ Additions

$2K$ Multiplications

$2(K - 1)$ Additions

Smoothing to Remove Image Noise



Problem with Smoothing:

- Sensitive to Outliers (**Noise**)
- Smooths Edges (**Blur**)

Median Filtering

1. Sort the K^2 values in window centered at the pixel
2. Assign the Middle Value (**Median**) to pixel

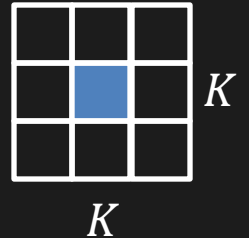


Image with
Salt and Pepper Noise



Median Filtered
Image ($K = 3$)

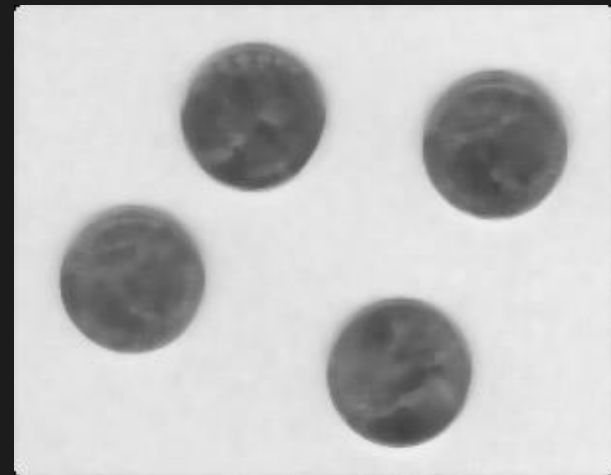
Non-linear Operation
(Cannot be implemented using convolution)

Median Filtering

Not Effective when Image Noise is not a Simple Salt and Pepper Noise.



Image with Noise



Median Filtered
Image ($K = 7$)

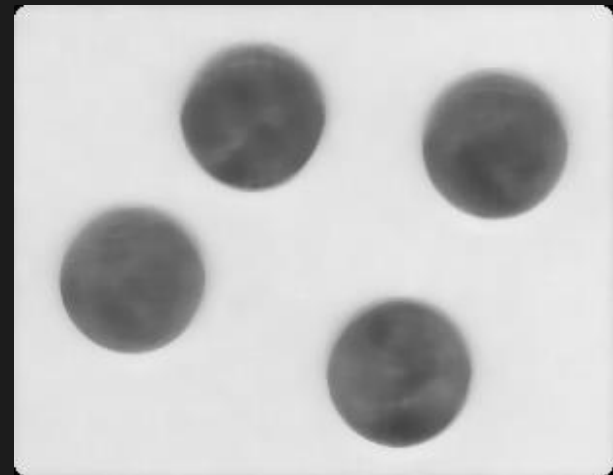
Larger K causes blurring of image detail

Median Filtering

Not Effective when Image Noise is not a Simple Salt and Pepper Noise.



Image with Noise



Median Filtered
Image ($K = 11$)

Larger K causes blurring of image detail

What Does this Do?

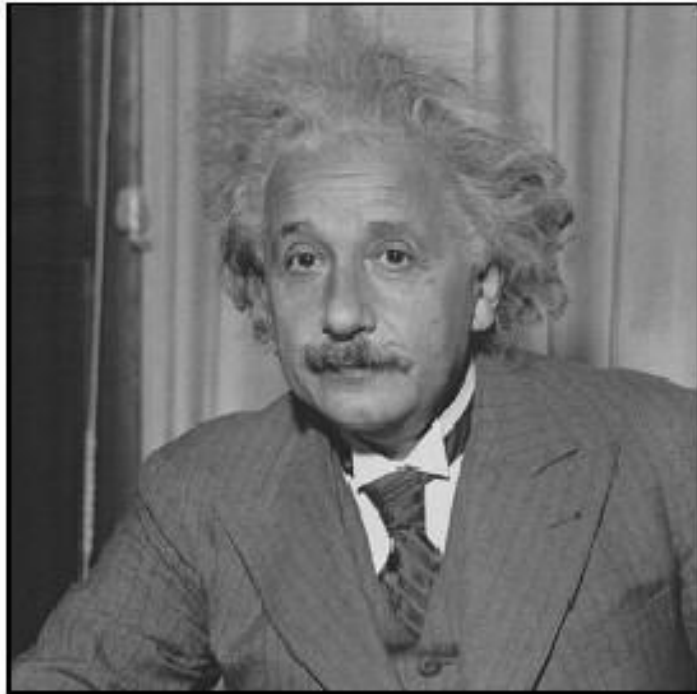
0	0	0
0	2	0
0	0	0

 $-$ $\frac{1}{9}$

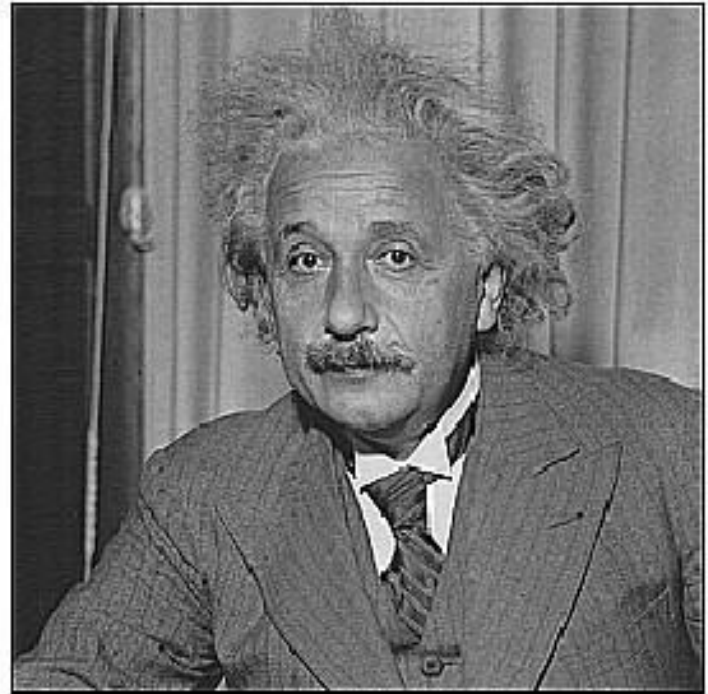
1	1	1
1	1	1
1	1	1

$f[i, j]$

Sharpening Filter



before



after

Template Matching



Template

How do we locate the template in the image?

Minimize:

$$E[i, j] = \sum_m \sum_n (f[m, n] - t[m - i, n - j])^2$$

$$E[i, j] = \sum_m \sum_n (f^2[m, n] + t^2[m - i, n - j] - \underbrace{2f[m, n]t[m - i, n - j]})$$

Maximize

Template Matching



Template

How do we locate the template in the image?

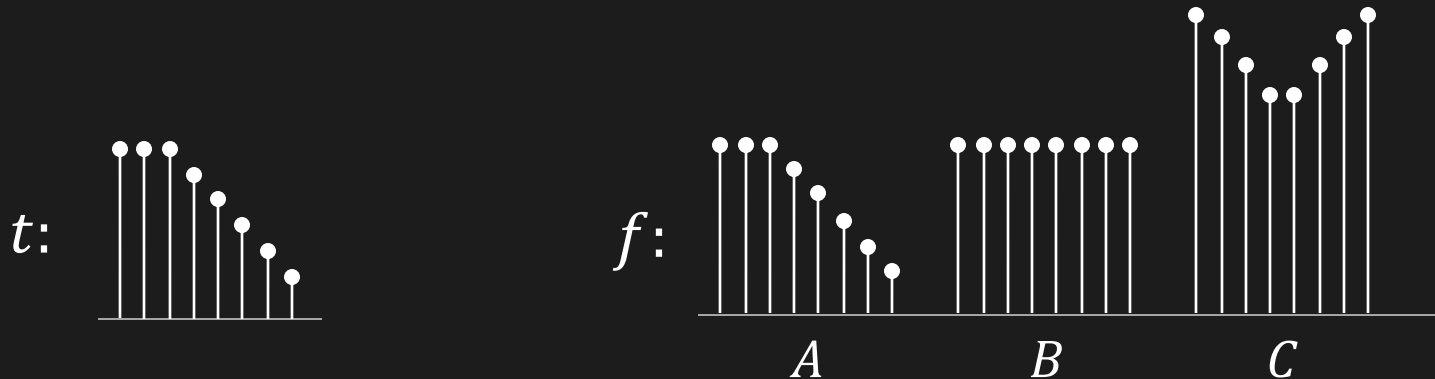
Maximize:

$$R_{tf}[i,j] = \sum_m \sum_n f[m,n]t[m-i,n-j] = t \otimes f$$

(Cross-Correlation)

Problem with Cross-Correlation

$$R_{tf}[i, j] = \sum_m \sum_n f[m, n] t[m - i, n - j] = t \otimes f$$



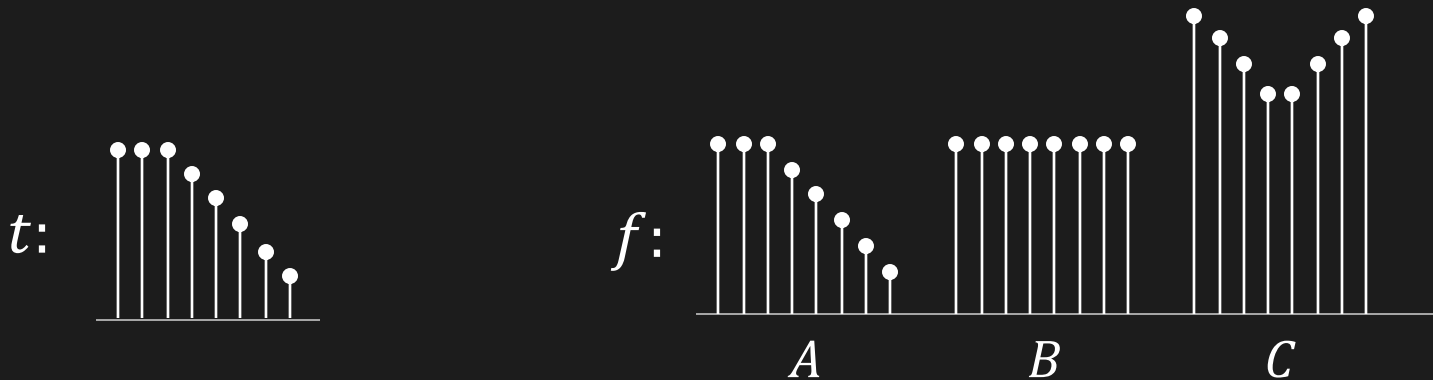
$$R_{tf}(C) > R_{tf}(B) > R_{tf}(A)$$

We need $R_{tf}(A)$ to be the maximum!

Normalized Cross-Correlation

Account for energy differences

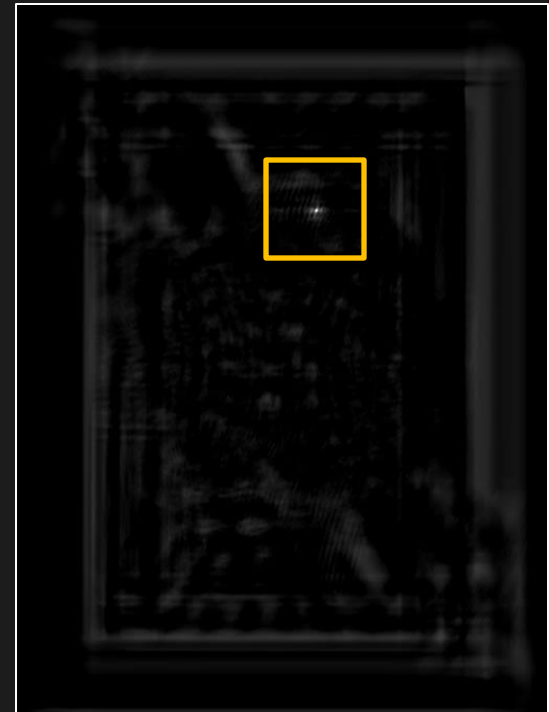
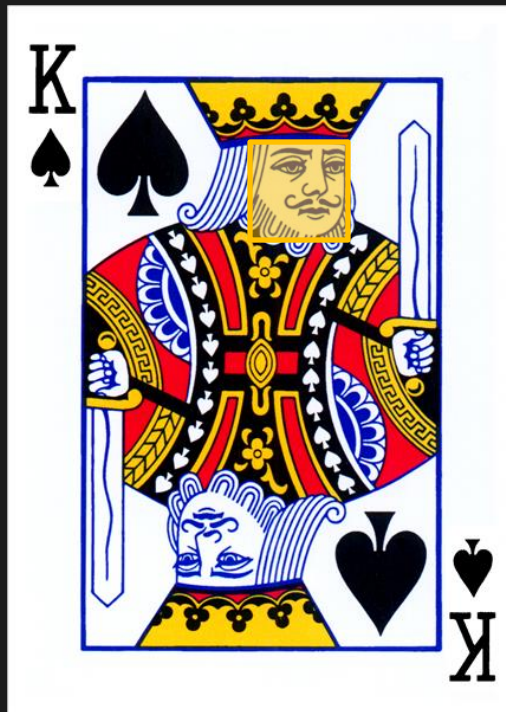
$$N_{tf}[i, j] = \frac{\sum_m \sum_n f[m, n] t[m - i, n - j]}{\sqrt{\sum_m \sum_n f^2[m, n]} \sqrt{\sum_m \sum_n t^2[m - i, n - j]}}$$



Normalized Cross-Correlation

Account for energy differences

$$N_{tf}[i, j] = \frac{\sum_m \sum_n f[m, n] t[m - i, n - j]}{\sqrt{\sum_m \sum_n f^2[m, n]} \sqrt{\sum_m \sum_n t^2[m - i, n - j]}}$$



Correlation: Issues

- Problem at borders
- Sensitive to object pose, scale and rotation
- Not good for general object recognition
- Good for feature detection

References: Textbooks

Digital Image Processing (Chapter 3)

González, R and Woods, R., Prentice Hall

Computer Vision: Algorithms and Applications (Chapter 3)

Szeliski, R., Springer

Robot Vision (Chapter 6 and 7)

Horn, B. K. P., MIT Press

Computer Vision: A Modern Approach (Chapter 7)

Forsyth, D and Ponce, J., Prentice Hall

References: Papers

[Tomasí 1998] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in Proceedings of the IEEE International Conference on Computer Vision, 1998.

Image Credits

- I.1 Sasse, Arthur. Einstein © Ullstein Bilderdienst, Berlin.
- I.2 <http://www.flickr.com/photos/byspice/4577634277>
- I.3 <http://www.toptenz.net/wp-content/uploads/2008/06/kristin-kreuk.jpg>
- I.4 http://images.nationalgeographic.com/wpf/media-live/photos/000/093/cache/giant-lobelia-plant_9372_600x450.jpg
- I.5 Matlab Demo Image
- I.6 Adapted from <http://1x.com/photo/24072>
- I.7 <http://cdn04.cdn.socialitelife.com/wp-content/uploads/2009/11/stars-without-makeup-photos-11062009-07.jpg>