Non-Linear Image filtering

https://liaoux.github.io/uir-cv-2026/

Introduction to Computer Vision
Spring 2026, Lecture 2.1

Smoothing to Remove Image Noise

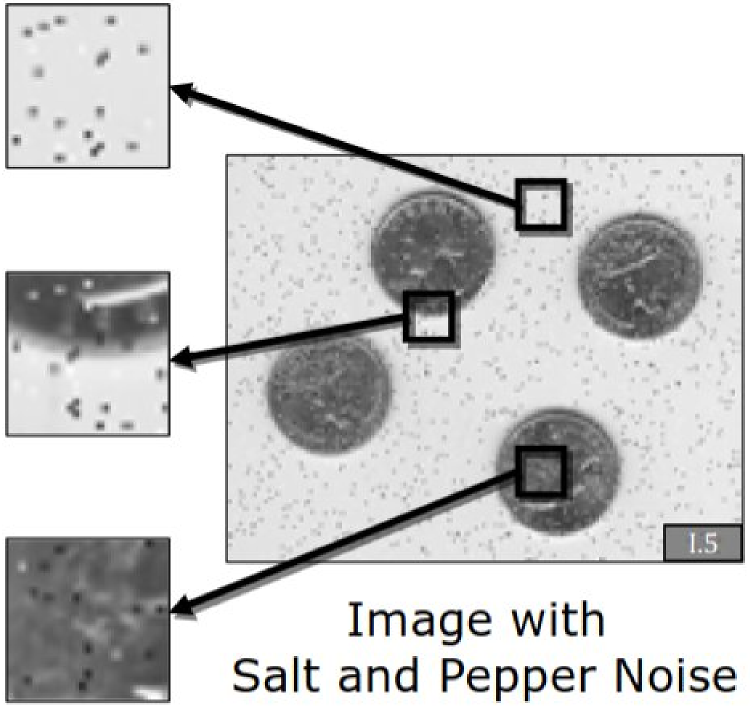
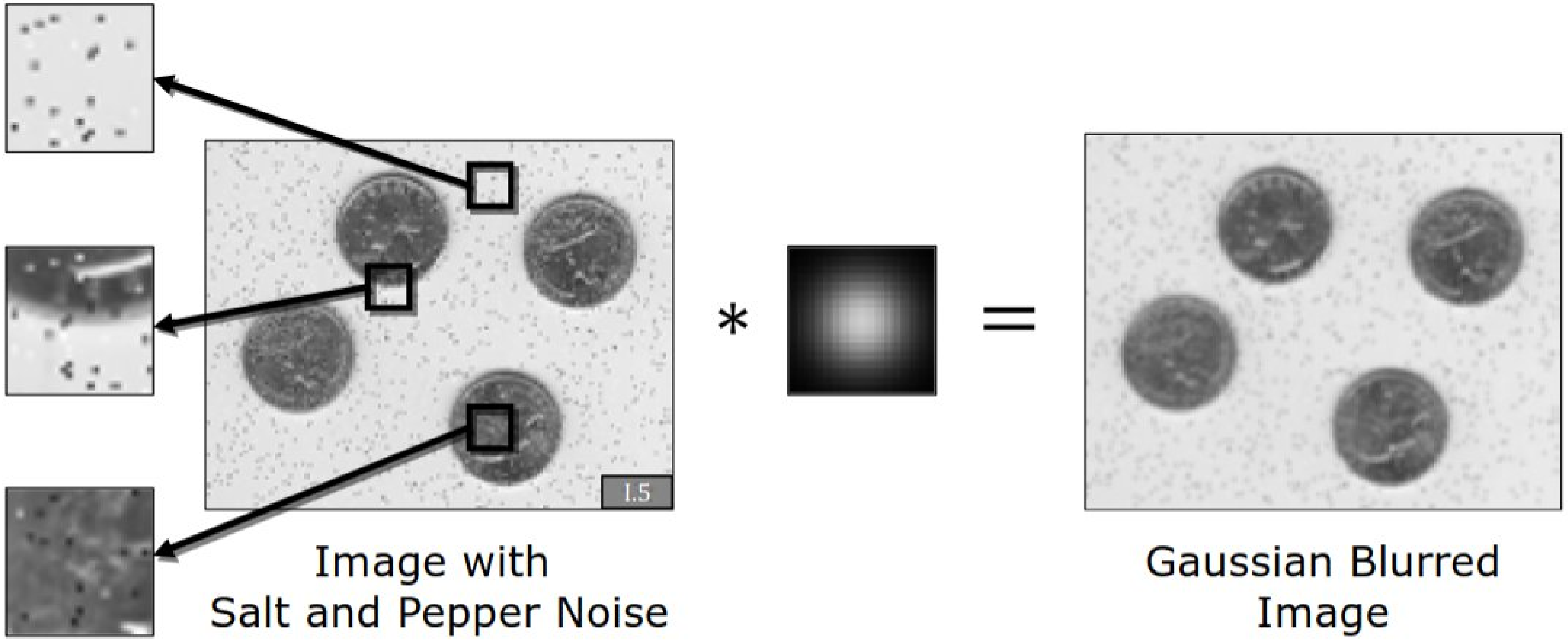
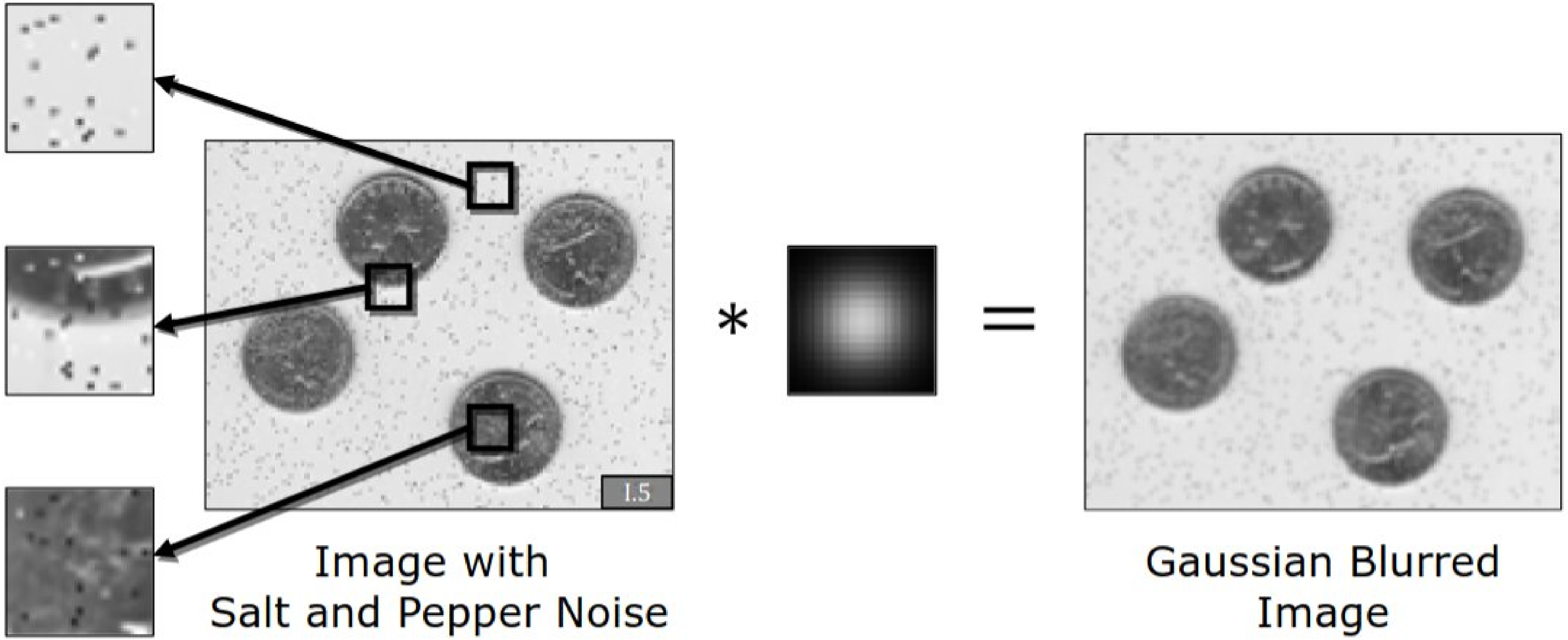


Image with
Salt and Pepper Noise

Smoothing to Remove Image Noise



Smoothing to Remove Image Noise

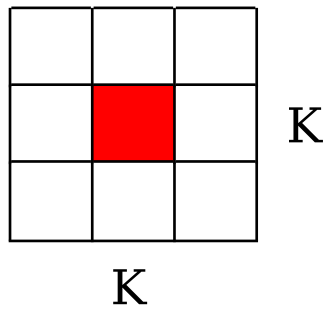


Problem with Smoothing:

- Does not remove 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



Median Filtering

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

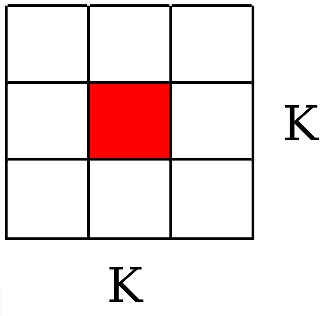
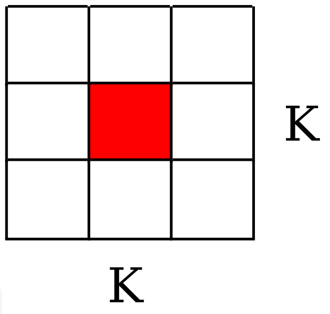


Image with
Salt and Pepper Noise

Median Filtered
Image ($K = 3$)

Median Filtering

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



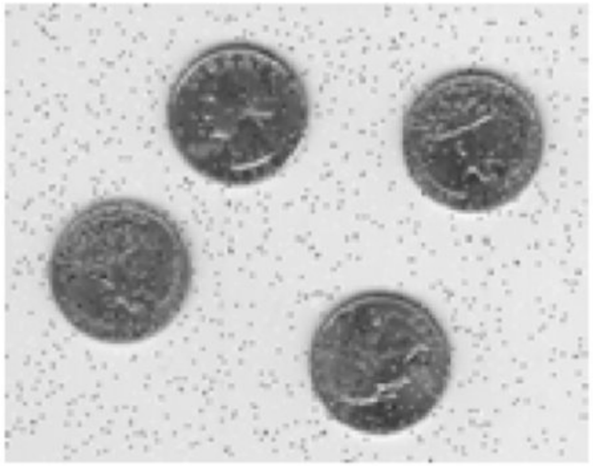


Image with
Salt and Pepper Noise

Median Filtered
Image ($K = 3$)

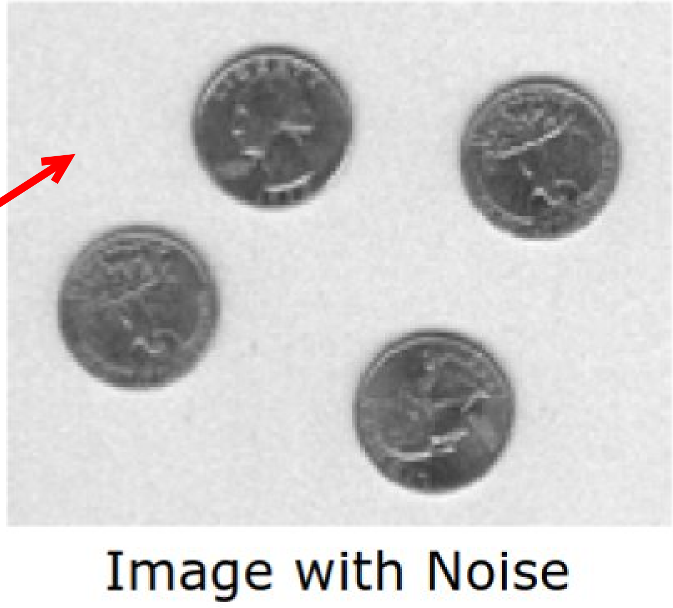
Non-Linear Operation

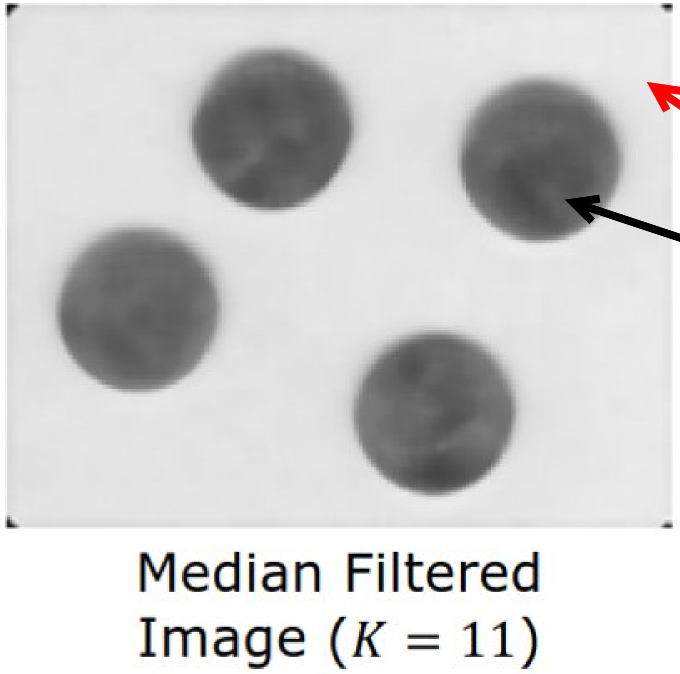
(Cannot be implemented using convolution)

Median Filtering

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

Similar to the noise under low light conditions





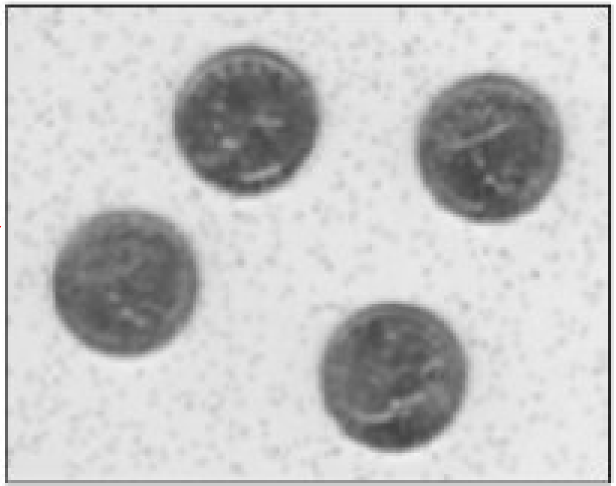
The noise is removed, but the details on the coins are almost lost

Larger K causes blurring of image detail

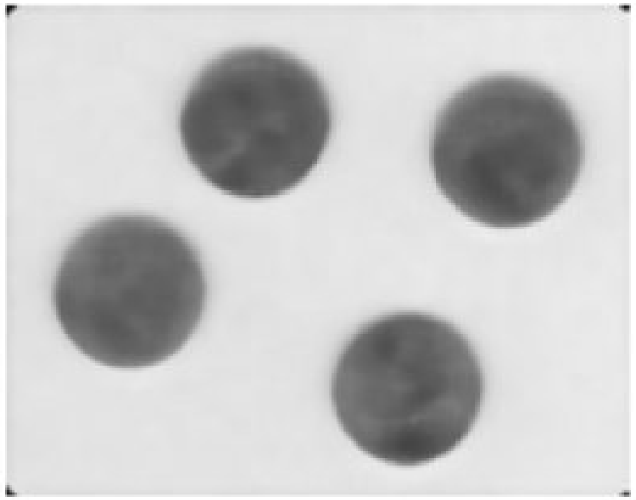
Linx Linear Filtering Non-Linear Filtering

Can we come up with a filter for removing noise that does better than both Gaussian smoothing (linear) and median filtering (non-linear) ?

Linear Filter



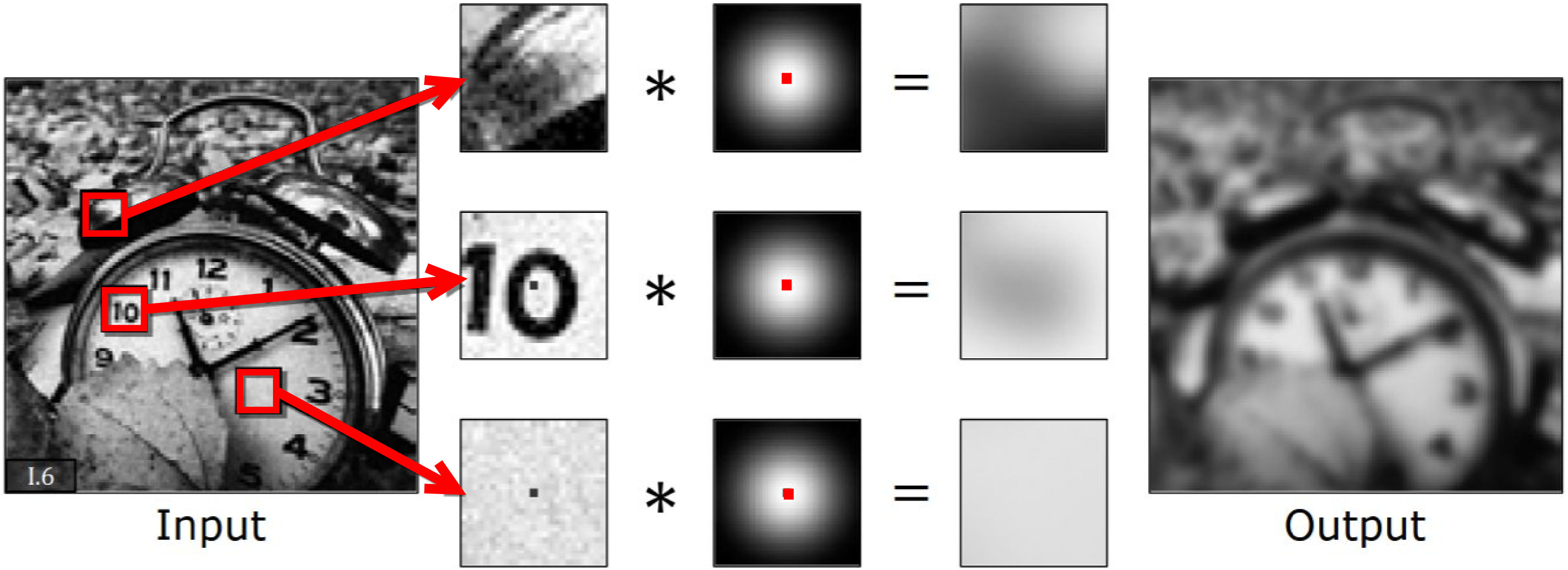
Gaussian Blurred Image



Median Filtered Image ($K = 11$)

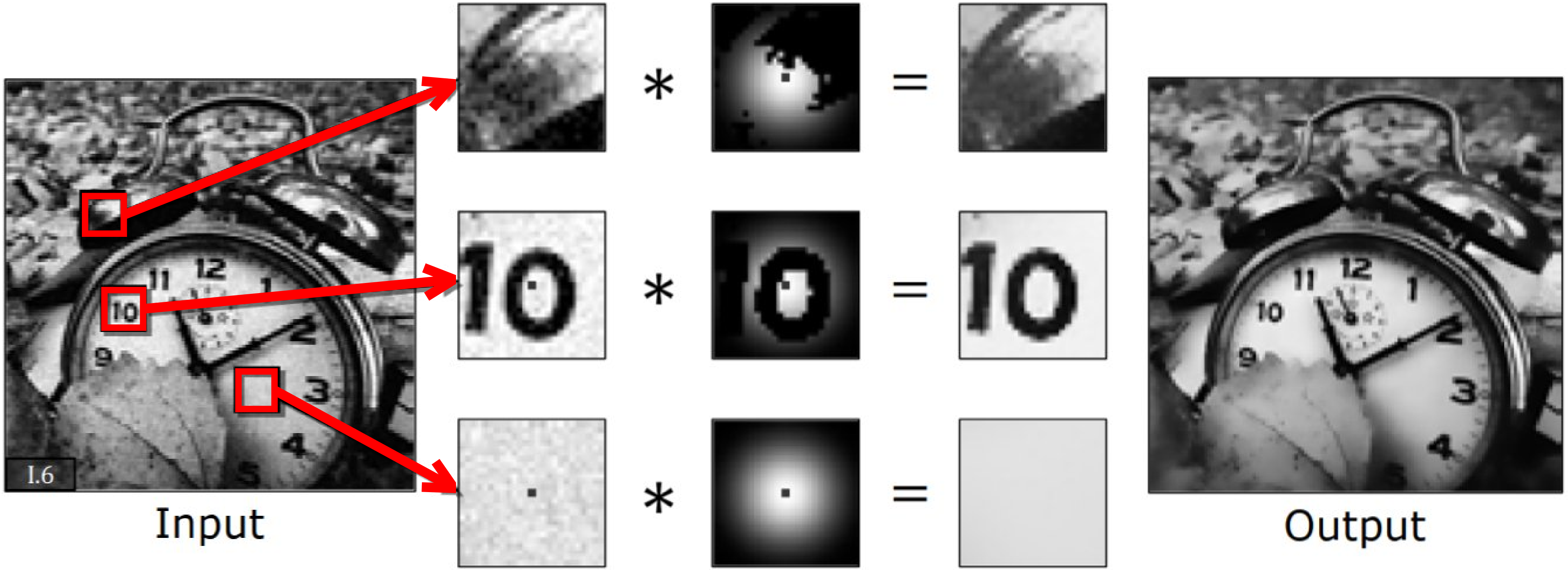
Non-linear Filter

Revisiting Gaussian Smoothing



Same Gaussian kernel is used everywhere.
Blurs across edges.

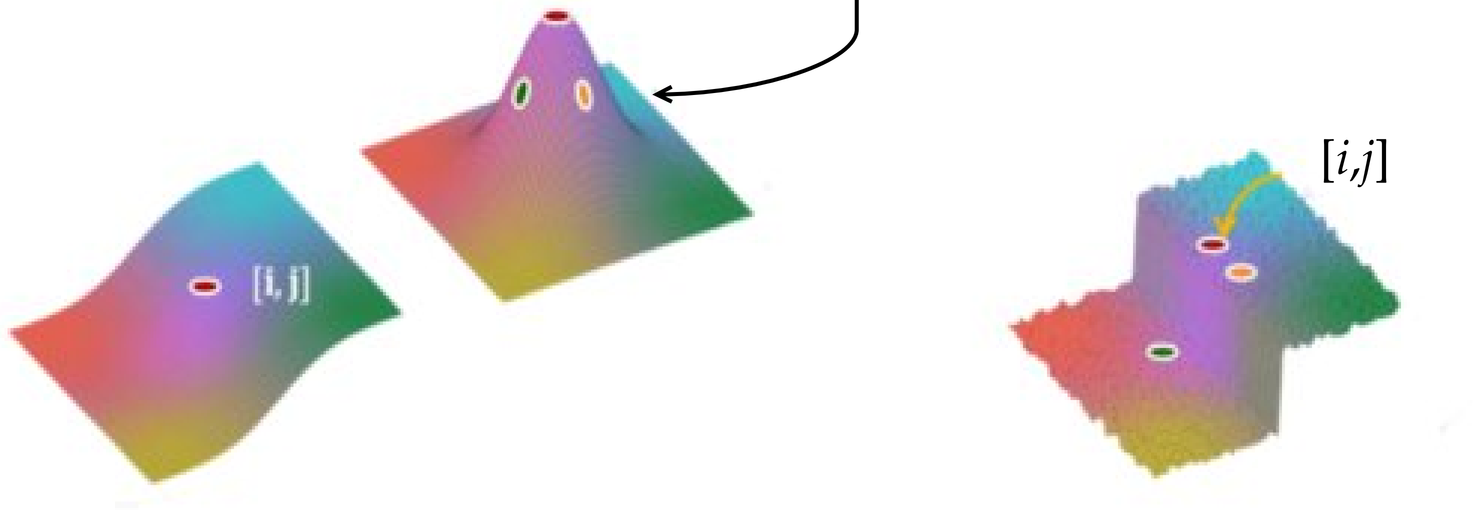
Blur Similar Pixels Only



"Bias" Gaussian Kernel such that pixels not similar in intensity to the center pixel receive a lower weight.

Bilateral Filter: Start With Gaussian

$$g[i,j] = \frac{1}{W_s}\sum_m \sum_n f[m,n]\underbrace{n_{\sigma_s}[i-m,j-n]}_{\text{Spatial Gaussian}}$$



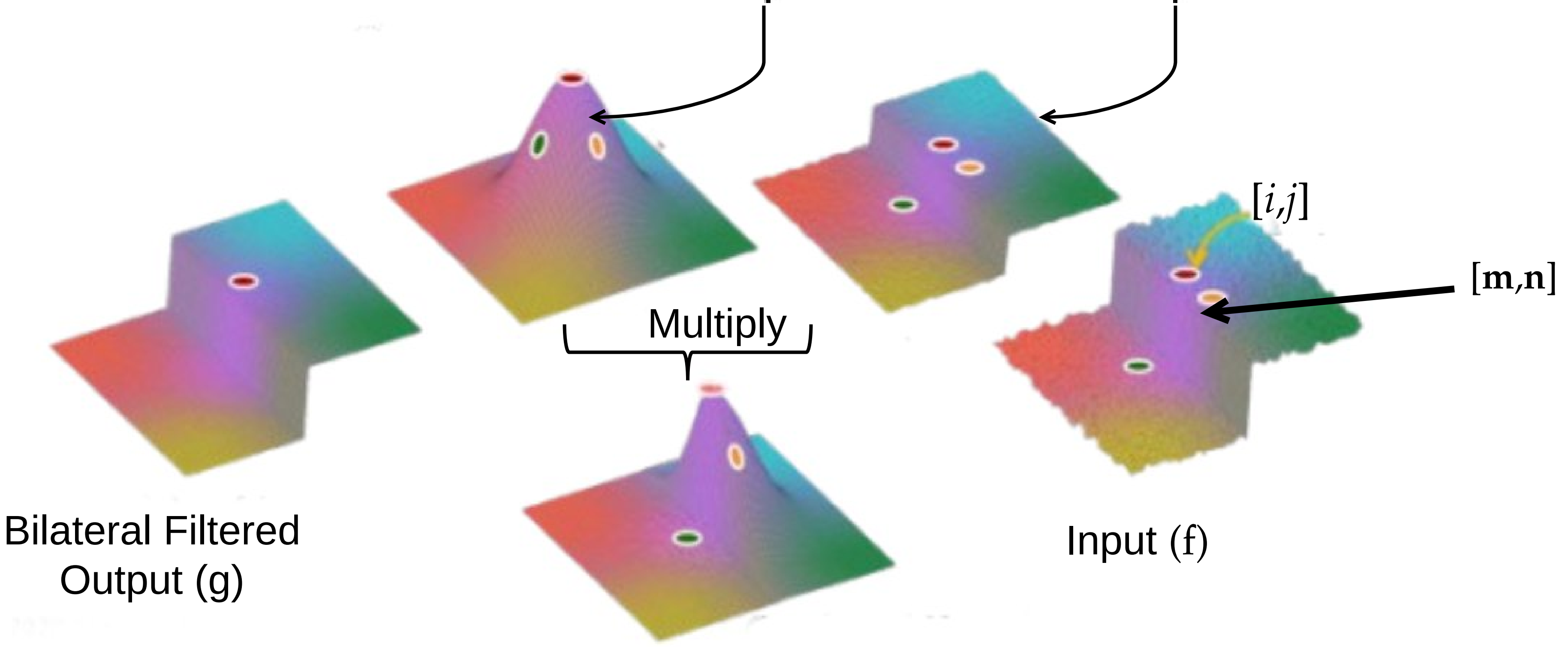
Gaussian Smoothed Output (g)

Input image (f)

Gaussian blurs across edges

Bilateral Filter: Add Bias to Gaussian

$$g[i,j] = \frac{1}{W_{sb}} \sum_m \sum_n f[m,n] \underbrace{n_{\sigma_s}[i-m,j-n]}_{\text{Spatial Gaussian}} \underbrace{n_{\sigma_b}(f[m,n]-f[i,j])}_{\text{Brightness Gaussian}}$$



Bilateral Filter: Summary

Spatial Gaussian Brightness Gaussian

$$g[i,j] = \frac{1}{W_{sb}} \sum_m \sum_n f[m,n] n_{\sigma_s}[i-m, j-n] n_{\sigma_b}(f[m,n] - f[i,j])$$

Where:

$$n_{\sigma_s}[m,n] = \frac{1}{2\pi{\sigma_s}^2} e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma_s^2}\right)} \qquad n_{\sigma_b}(k) = \frac{1}{\sqrt{2\pi}\sigma_b} e^{-\frac{1}{2}\left(\frac{k^2}{\sigma_b^2}\right)}$$

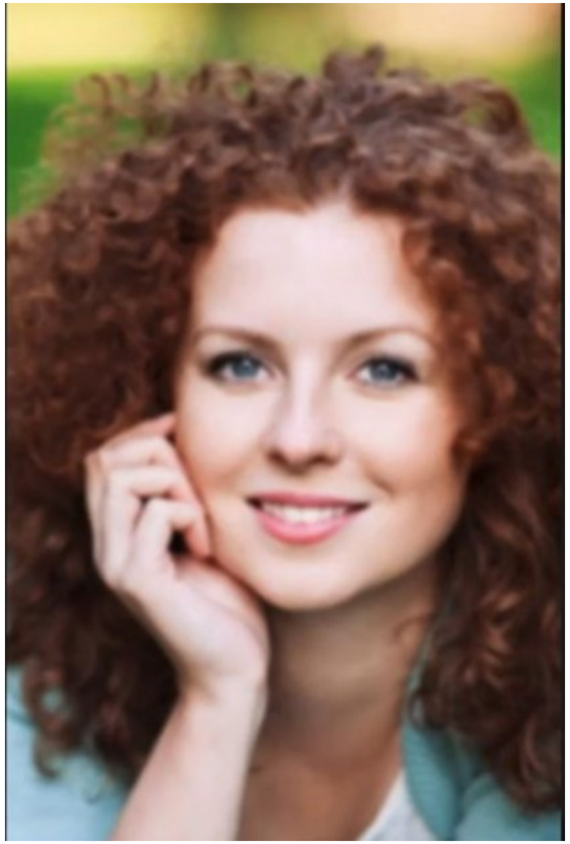
$$W_{sb} = \sum_m \sum_n n_{\sigma_s}[i-m, j-n] n_{\sigma_b}(f[m,n] - f[i,j])$$

Non-Linear Operation

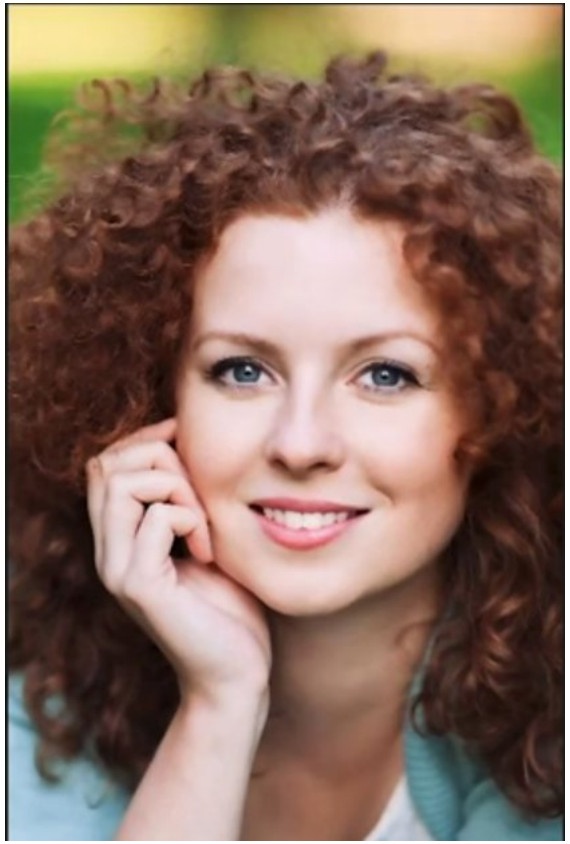
(Cannot be implemented using convolution)

Gaussian vs. Bilateral Filtering: Example

Original



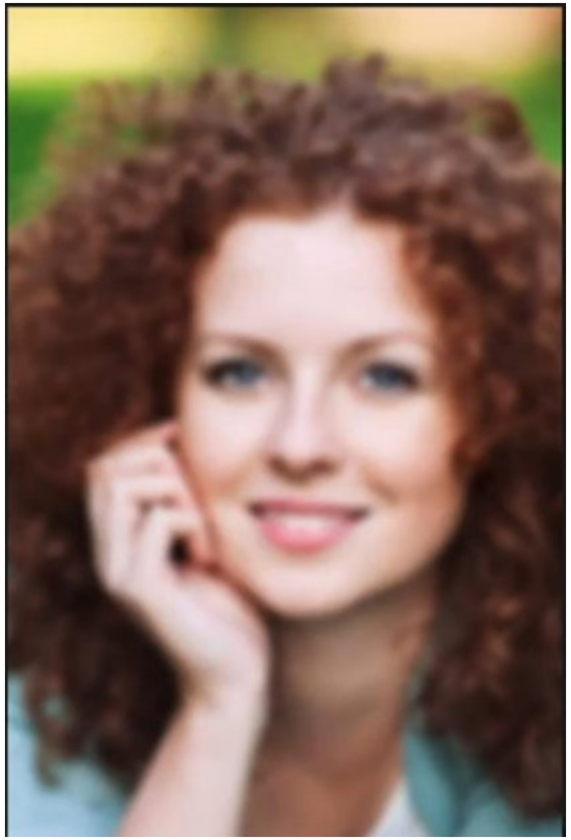
Gaussian
$\sigma_s = 2$



Bilateral
$\sigma_s = 2, \sigma_b = 10$

Gaussian vs. Bilateral Filtering: Example

Original

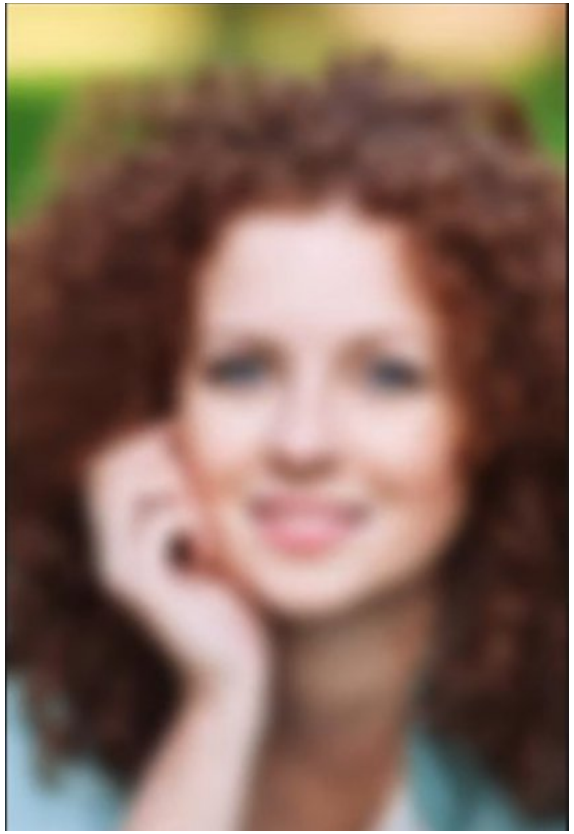


Gaussian
$\sigma_s = 4$

Bilateral
$\sigma_s = 4, \sigma_b = 10$

Gaussian vs. Bilateral Filtering: Example

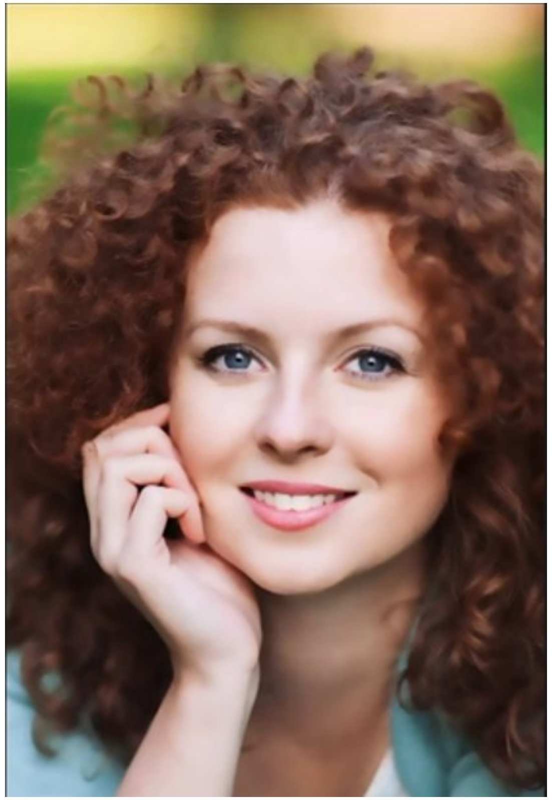
Original



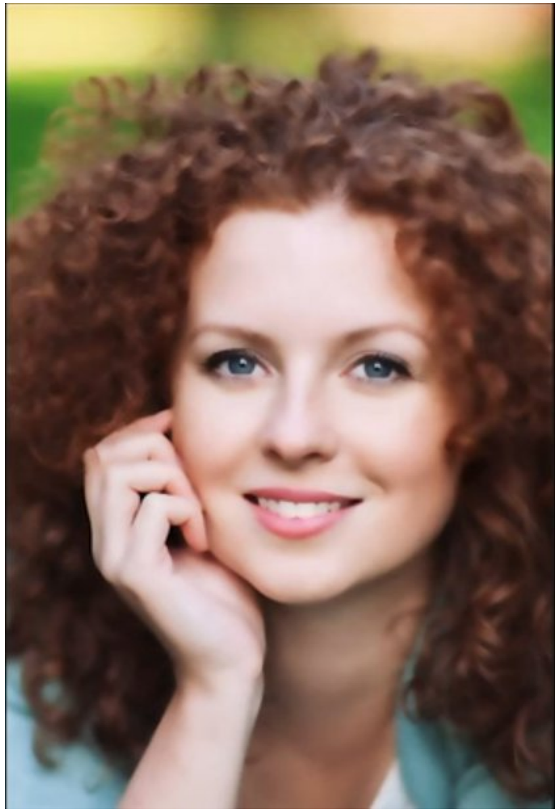
Gaussian
$\sigma_s = 8$

Bilateral
$\sigma_s = 8, \sigma_b = 10$

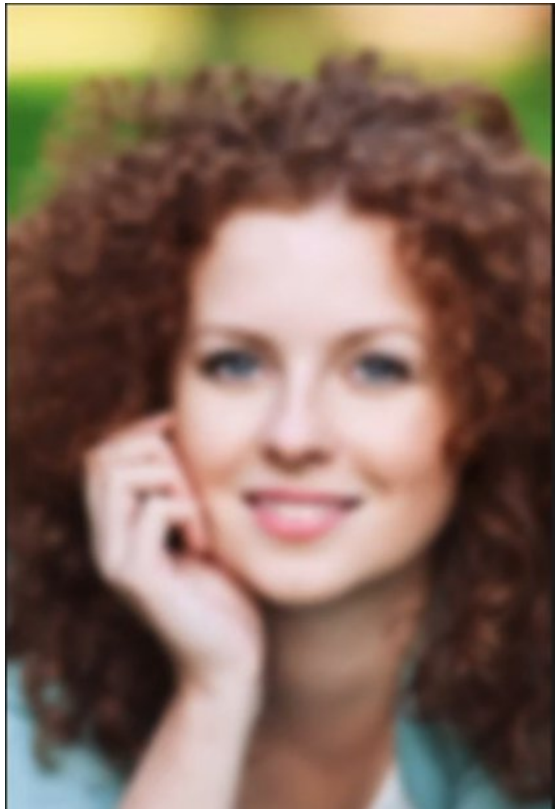
Gaussian vs. Bilateral Filtering: Example



Bilateral
$\sigma_s = 6, \sigma_b = 10$



Bilateral
$\sigma_s = 6, \sigma_b = 20$



Bilateral
$\sigma_s = 6, \sigma_b = \infty$
(Gaussian Smoothing)

Template Matching

in this image ?

Can you find the wolf's head

Template Matching

Template

How do we locate the template in the image ?

Template Matching

Template

How do we locate the template in the image ?

Minimize:

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

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)

Convolution vs Correlation

Convolution:

$$g[i,j] = \sum_m \sum_n f[m,n]\underline{t[i-m,j-n]} = t * f$$

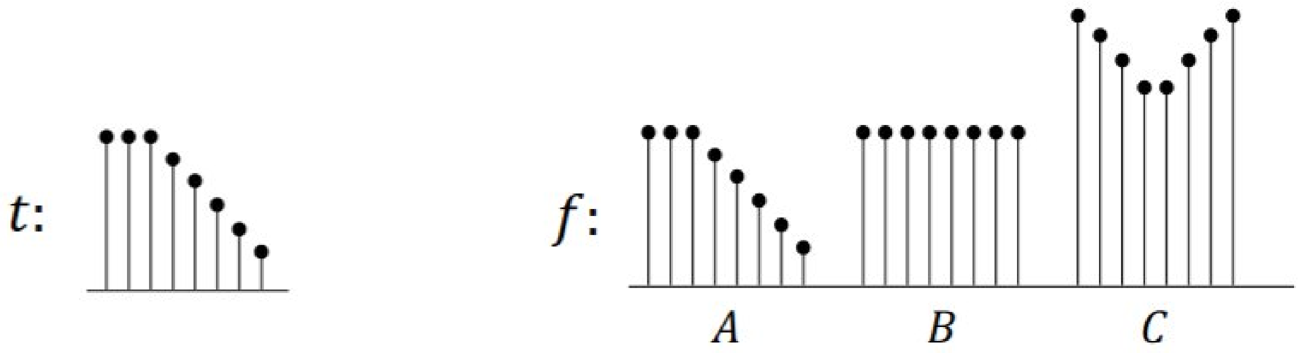
Correlation:

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

No Flipping in Correlation

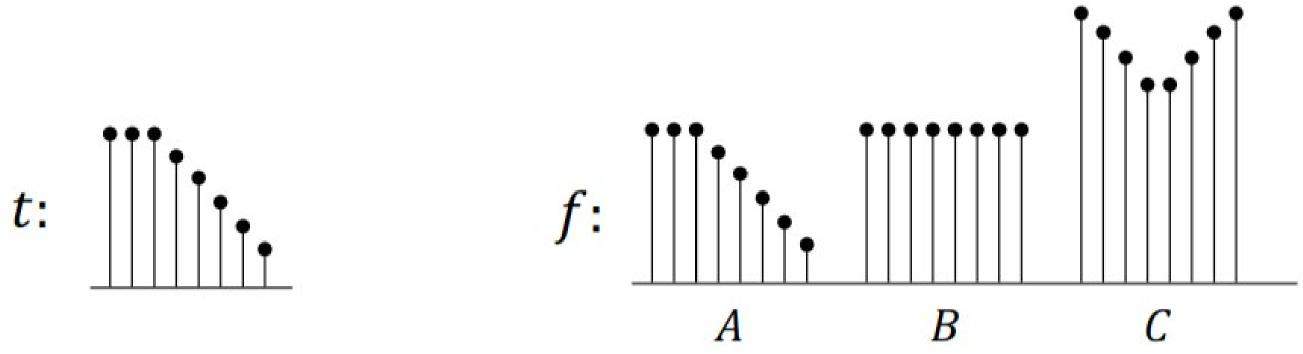
Problem with Cross-Correlation

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



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!

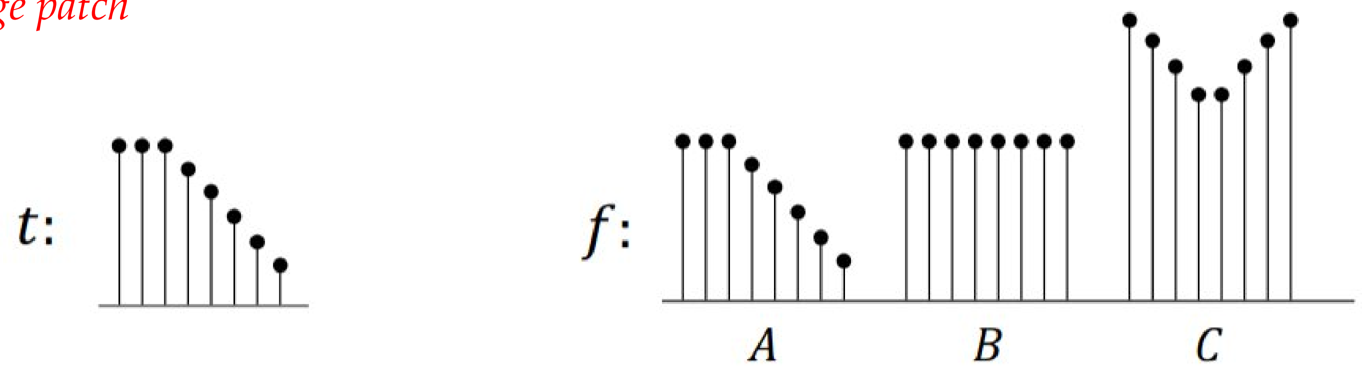
Normalized Cross-Correlation

Account for energy differences

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

$\sqrt{\textit{Energy of image patch}}$

$\sqrt{\textit{Energy of template}}$



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

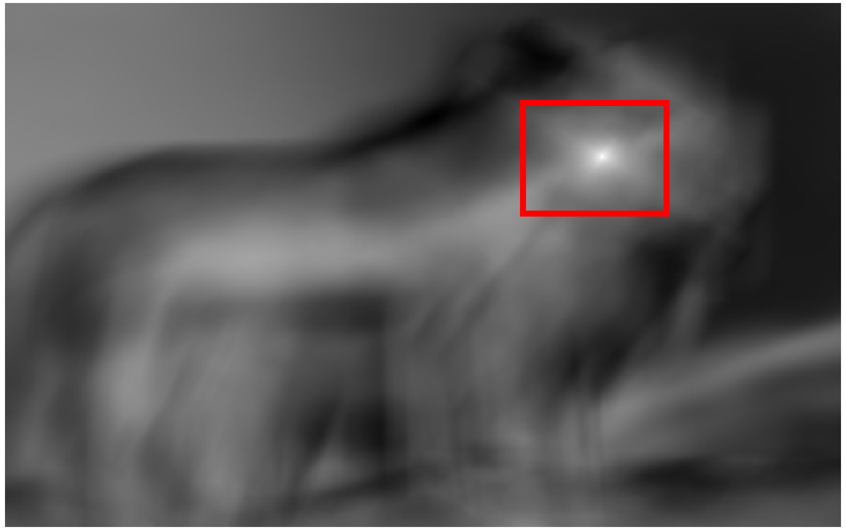
Examples of Template Matching Using Normalized Cross-Correlation

image (f)

$\otimes$

template (t)

$=$



The brighter the pixel, the higher the correlation value. (the wolf's face)

Examples of Template Matching
Using Normalized Cross-Correlation

image (f)

$\otimes$

template (t)

$=$

The brighter the pixel, the higher the correlation value. (the car’s emblem)

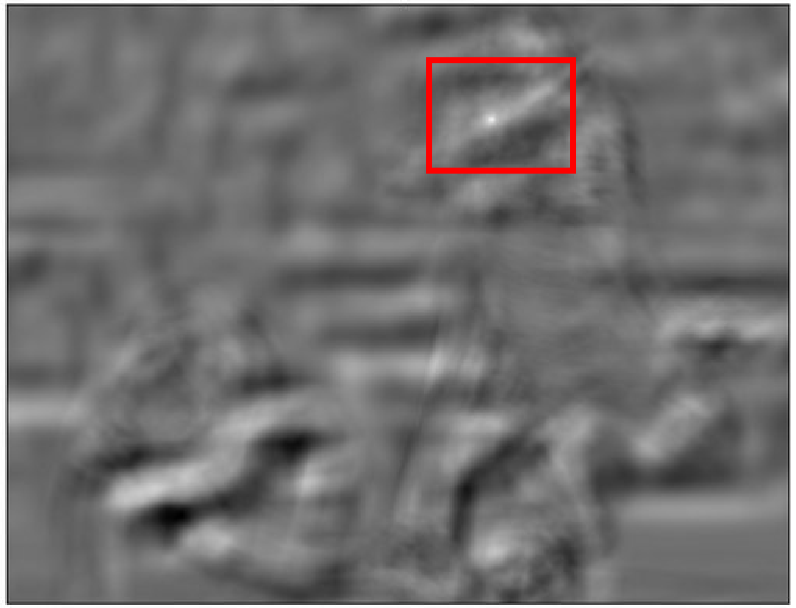
Examples of Template Matching Using Normalized Cross-Correlation

image (f)

$\otimes$

template (t)

$=$



The brighter the pixel, the higher the correlation value. (the Messi's face)

Recap: Image Processing I

Transform an image to a new one that is clearer or easier to analyze

Topics:

1. Pixel Processing
2. LSIS and Convolution
3. Linear Image Filters
4. Non Linear Image Filters
5. Template Matching by Correlation