# More image filtering

15-463, 15-663, 15-862
Computational Photography
Fall 2017, Lecture 4

http://graphics.cs.cmu.edu/courses/15-463

# Course announcements

- Any questions about Homework 1?
  - How many of you have read/started/finished the homework?

- Make sure to take the Doodle about rescheduling the September 27th lecture!
  - Link available on Piazza.
  - Currently 10 responses.

# Overview of today's lecture

- Template matching.
- Morphological filters.
- Rank filters.
- Adaptive thresholding.
- Bilateral filtering.
- Non-local means.

# Slide credits

Most of these slides were adapted directly from:

- Kris Kitani (15-463, Fall 2016).

Inspiration and some examples also came from:

- James Hays (Georgia Tech).
- Bernd Girod (Stanford).

# Template matching

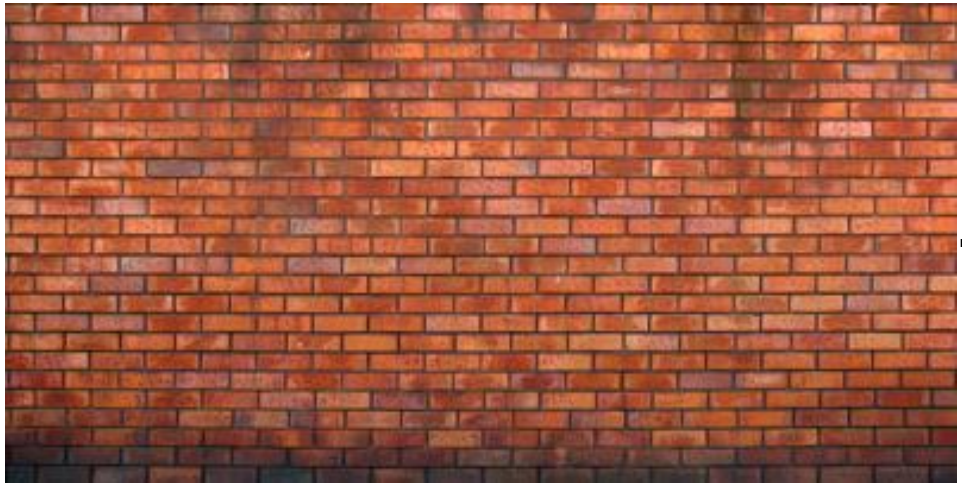
# Reminder from last time

How do we detect an edge?

# Reminder from last time

How do we detect an edge?

- We filter with something that looks like an edge.

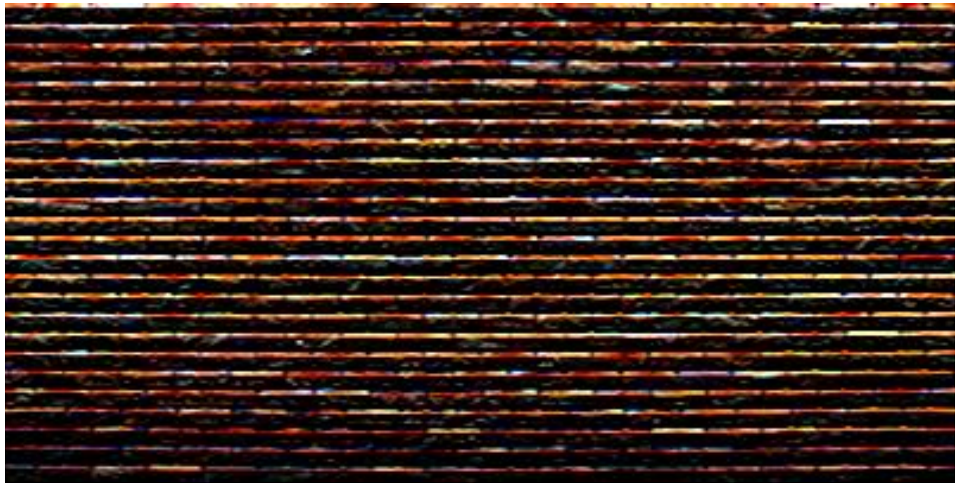


original

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

$* \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$

horizontal edge filter

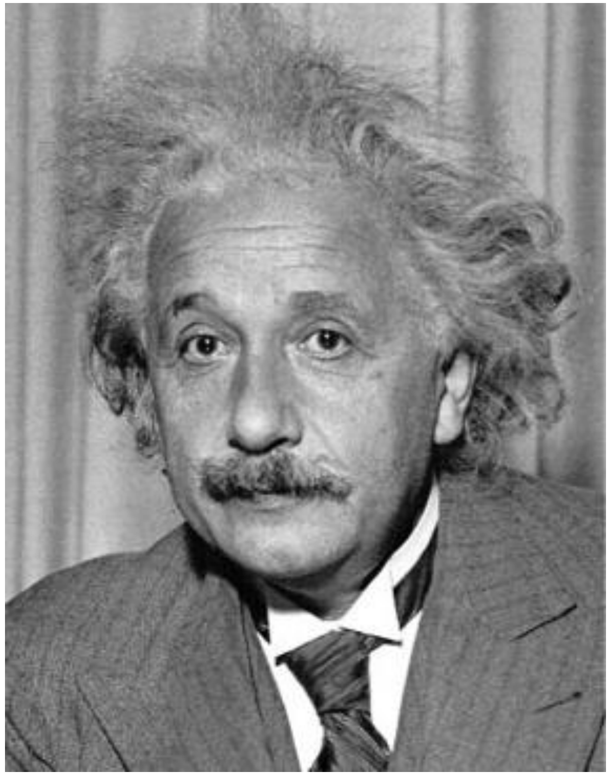


vertical edge filter

We can think of linear filtering as a way to evaluate how similar an image is *locally* to some template.

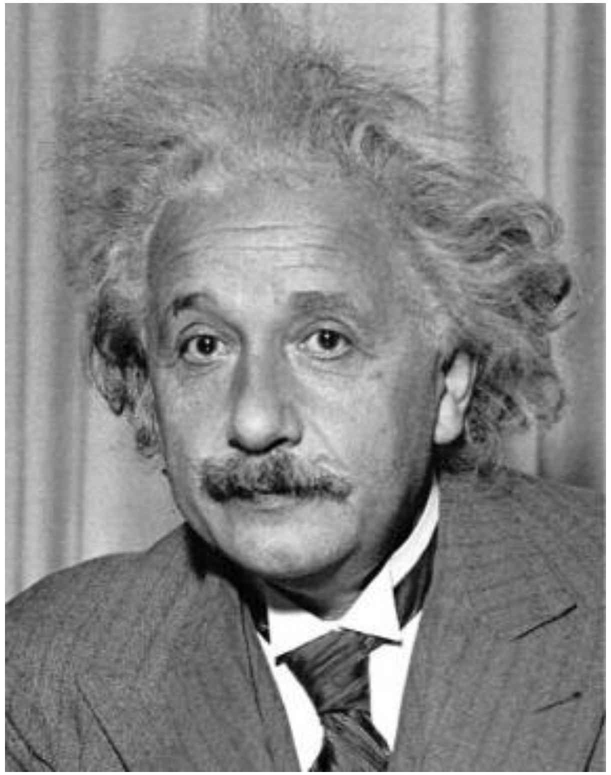
# Find this template

How do we detect the template  in he following image?



# Find this template

How do we detect the template  in he following image?



filter 

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

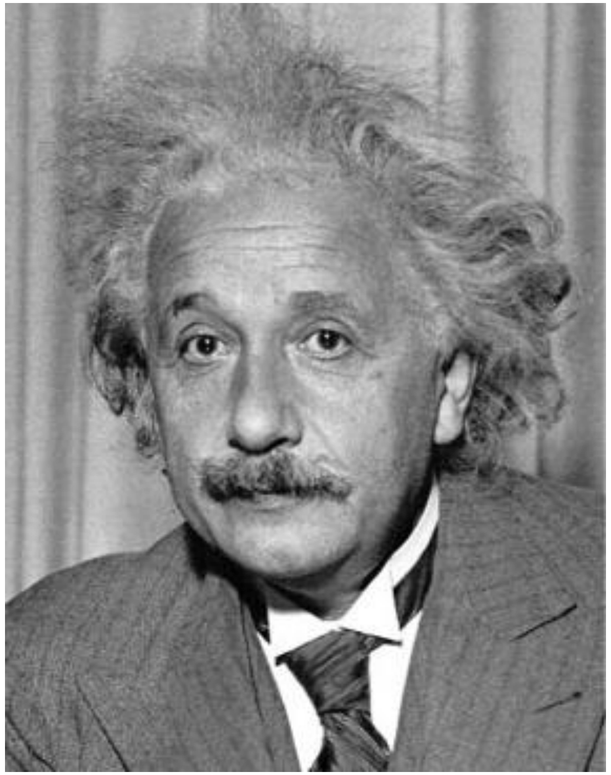
image

What will
the output
look like?

Solution 1: Filter the image using the template as filter kernel.

# Find this template

How do we detect the template  in he following image?

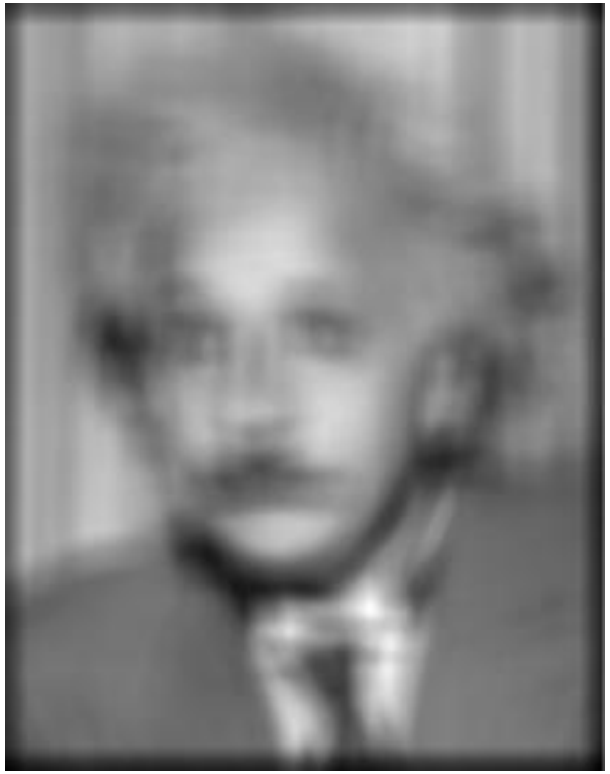


filter 

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image

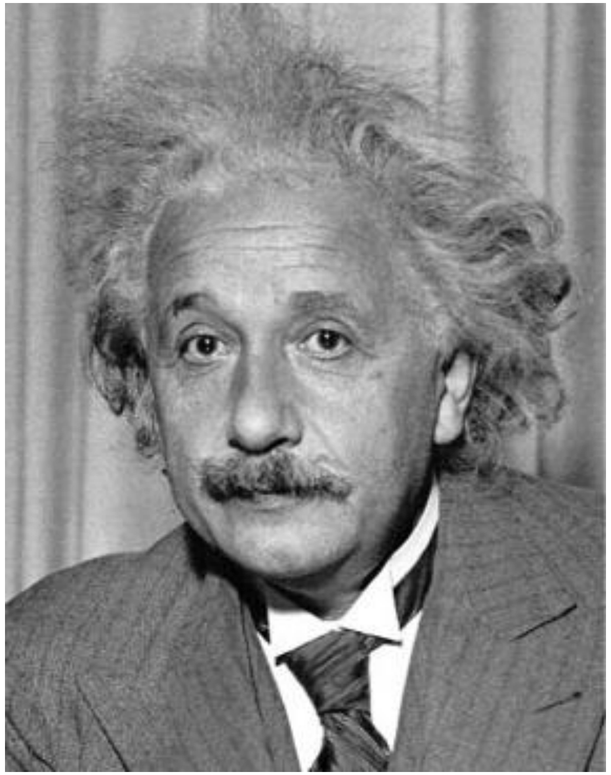


Solution 1: Filter the image using the template as filter kernel.

What went wrong?

# Find this template

How do we detect the template  in he following image?

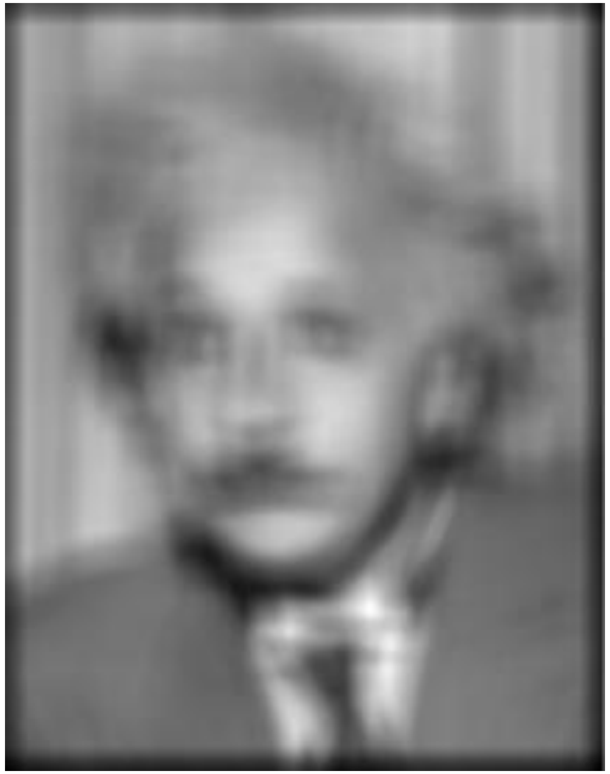


filter 

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image

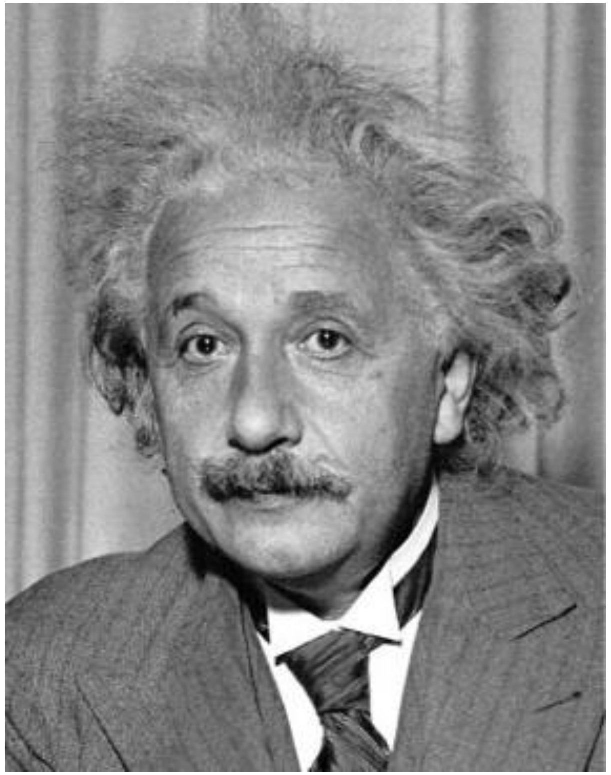


Increases for higher local intensities.

Solution 1: Filter the image using the template as filter kernel.

# Find this template

How do we detect the template  in he following image?



filter  template mean

output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

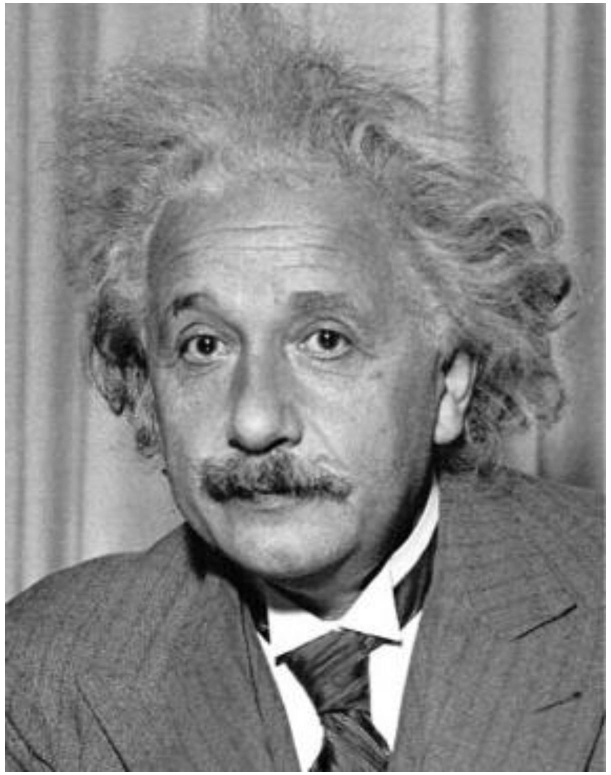
image

What will the output look like?

Solution 2: Filter the image using a *zero-mean* template.

# Find this template

How do we detect the template  in he following image?



filter 

template mean

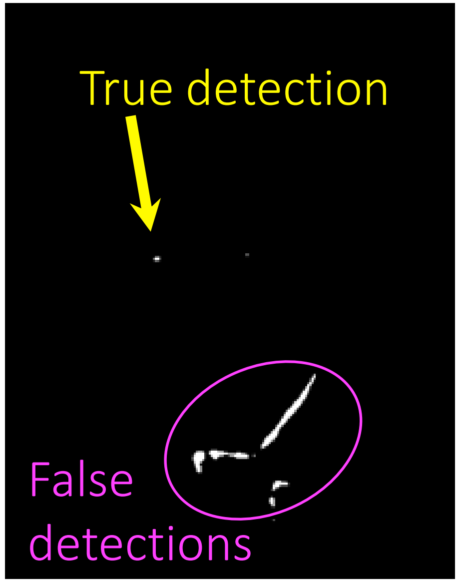
output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

image

output

thresholding

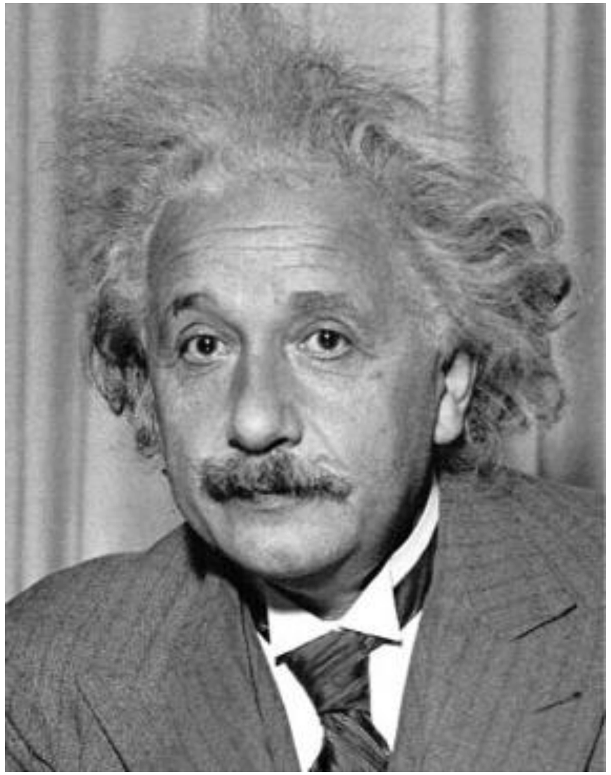



Solution 2: Filter the image using a *zero-mean* template.

What went wrong?

# Find this template

How do we detect the template  in he following image?



filter 

template mean

output

$$h[m,n] = \sum_{k,l} (g[k,l] - \bar{g}) f[m+k, n+l]$$

image

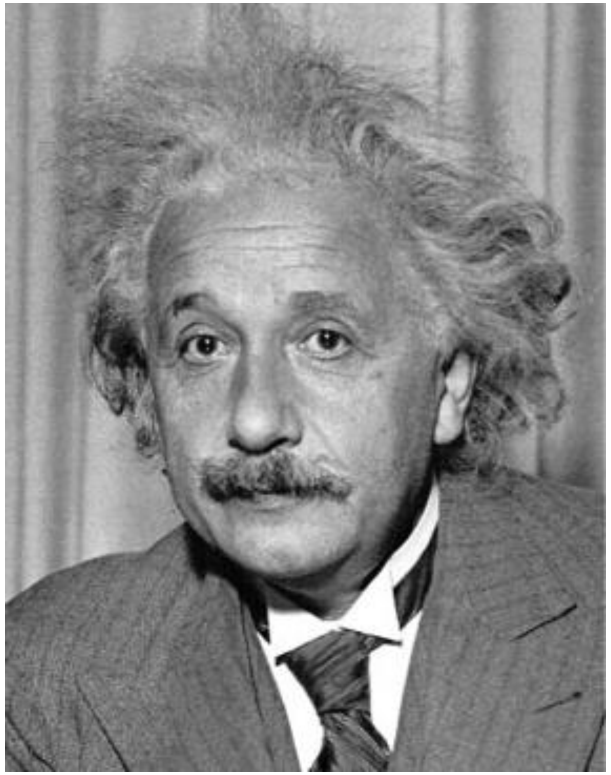
output

Not robust to high-contrast areas

Solution 2: Filter the image using a *zero-mean* template.

# Find this template

How do we detect the template  in he following image?



filter 

output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

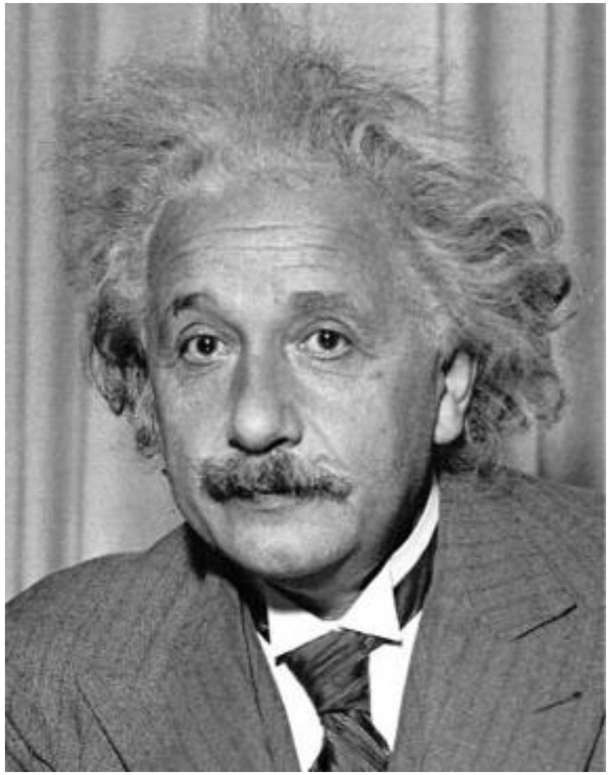
image

What will the output look like?

Solution 3: Use sum of squared differences (SSD).

# Find this template

How do we detect the template  in he following image?



filter 

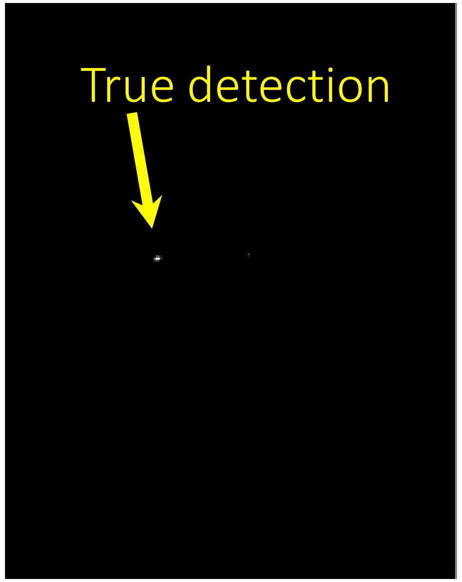
output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$

image

1-output

thresholding

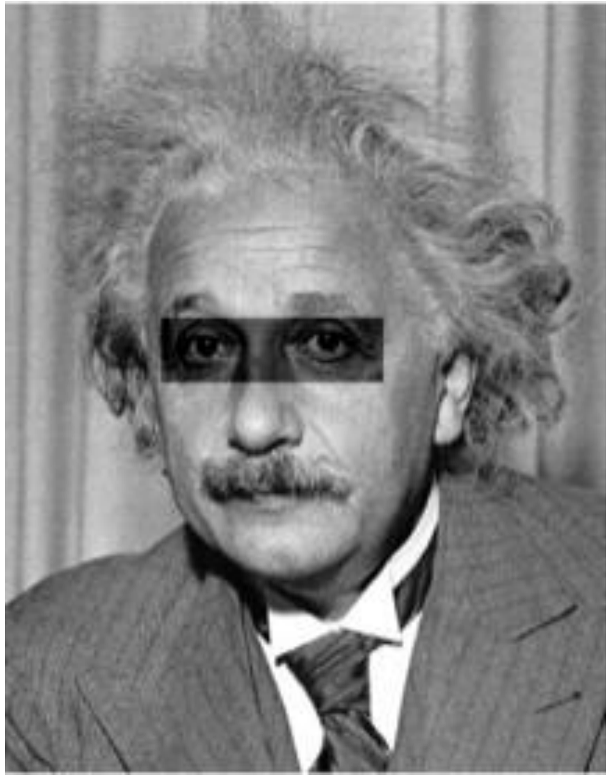



Solution 3: Use sum of squared differences (SSD).

What could go wrong?

# Find this template

How do we detect the template  in he following image?



filter 

output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

image

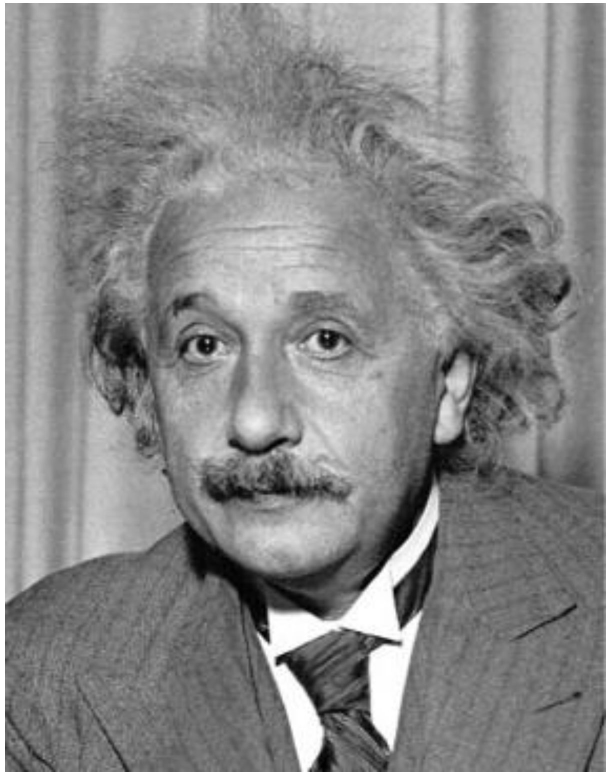
1-output

Not robust to local intensity changes

Solution 3: Use sum of squared differences (SSD).

# Find this template

How do we detect the template  in he following image?



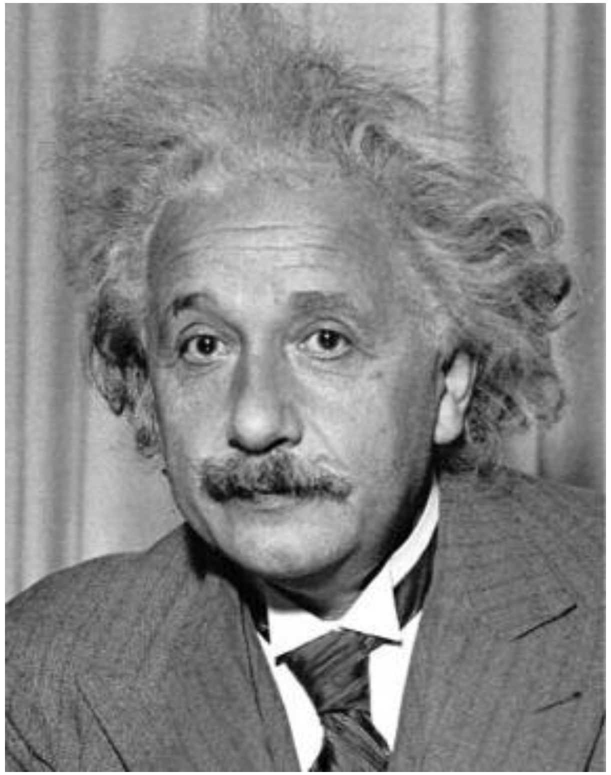
Observations so far:

- subtracting mean deals with brightness bias
- dividing by standard deviation removes contrast bias

Can we combine the two effects?

# Find this template

How do we detect the template  in he following image?



What will the output look like?

filter 

template mean

output

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l]-\bar{f}_{m,n})}{\sqrt{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)}}$$

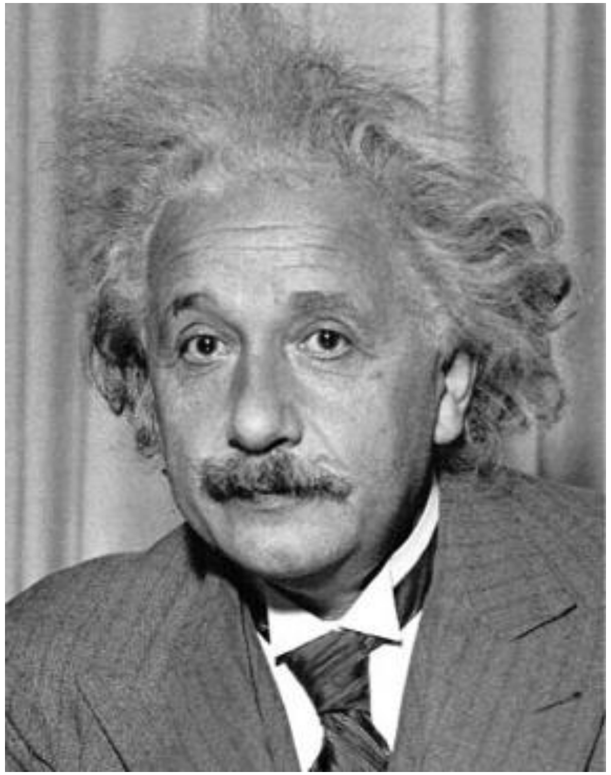
image

local patch mean

Solution 4: Normalized cross-correlation (NCC).

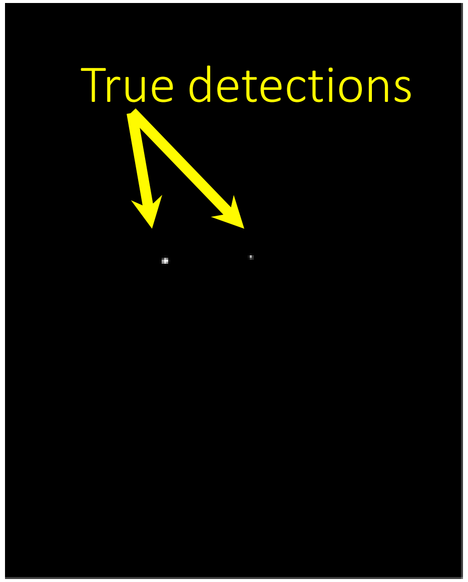
# Find this template

How do we detect the template  in he following image?



1-output

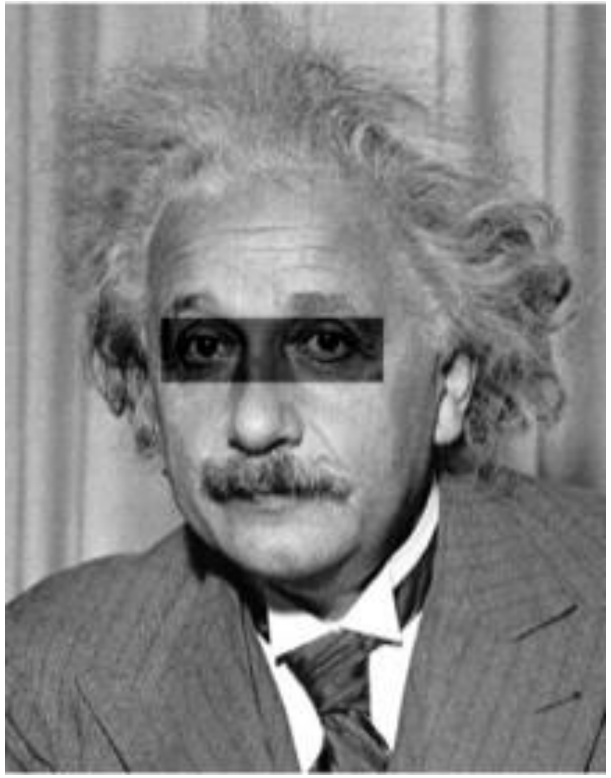
thresholding




Solution 4: Normalized cross-correlation (NCC).

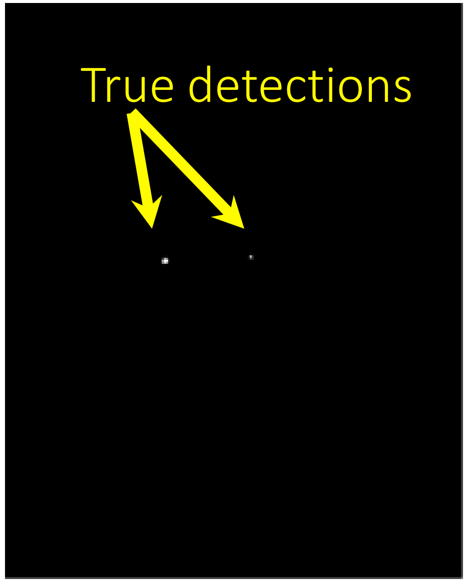
# Find this template

How do we detect the template  in he following image?



1-output

thresholding




Solution 4: Normalized cross-correlation (NCC).

# What is the best method?

It depends on whether you care about speed or invariance.

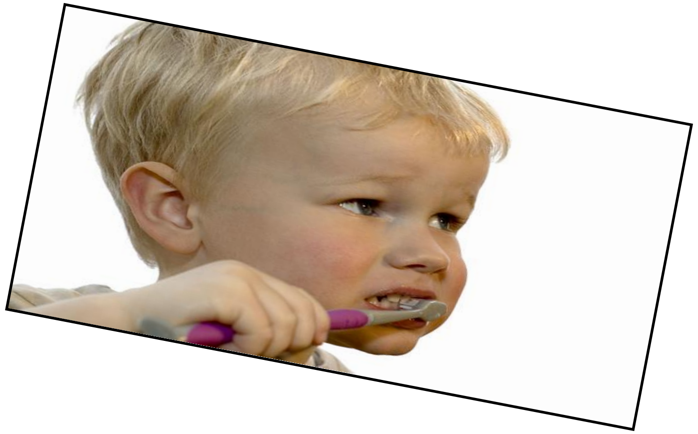
- Zero-mean: Fastest, very sensitive to local intensity.
- Sum of squared differences: Medium speed, sensitive to intensity offsets.
- Normalized cross-correlation: Slowest, invariant to contrast and brightness.

# Reminder: two types of image transformations

Filtering

changes pixel *values*

Warping



changes pixel *locations*

# Effects of image warping

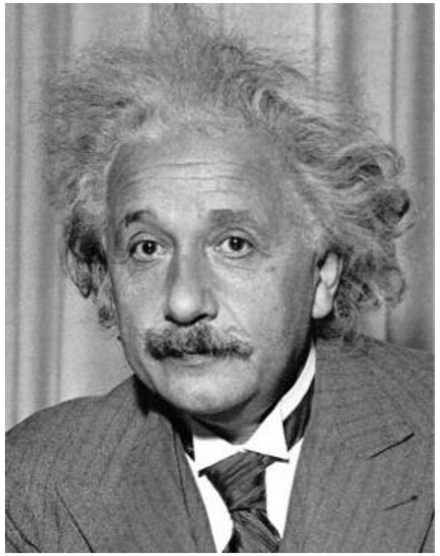
How well does patch-based template matching do under warping?

# Effects of image warping

How well does patch-based template matching do under warping?

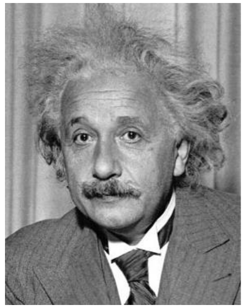
- Not at all.

can handle

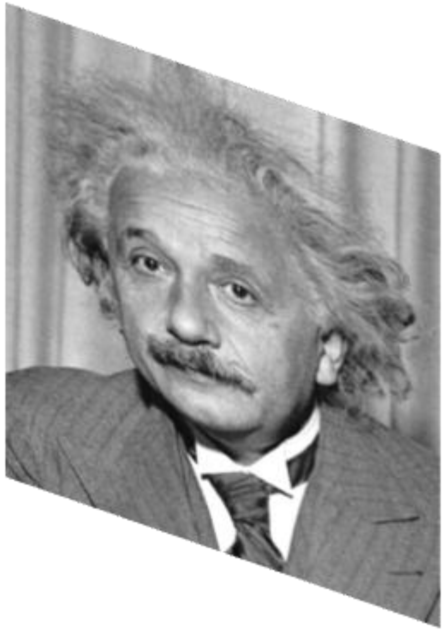


original

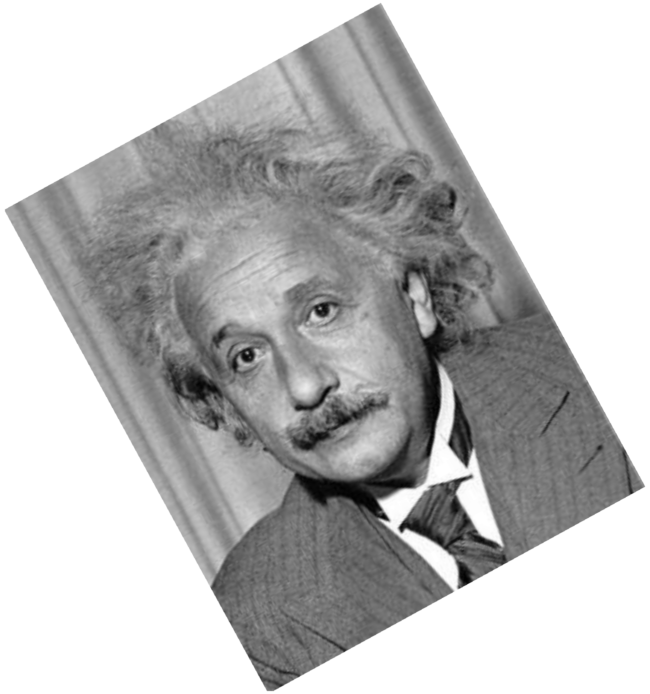
can't handle



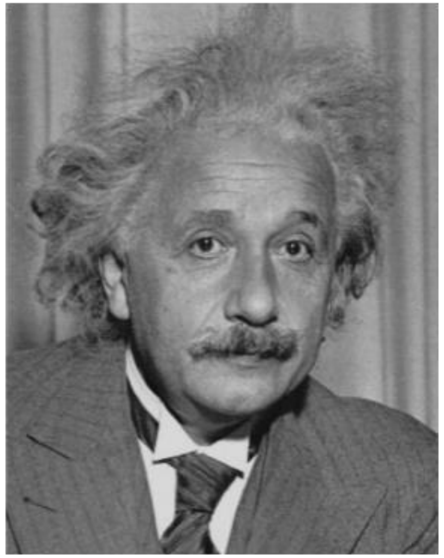
scaling



shearing



rotation

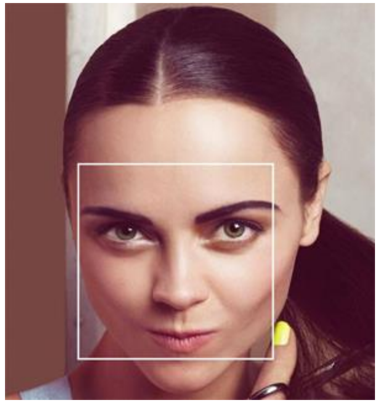


reflection

How would you handle these cases?

# Applications of template matching

## Face detection



http://davidwalsh.name/face-detection-jquery

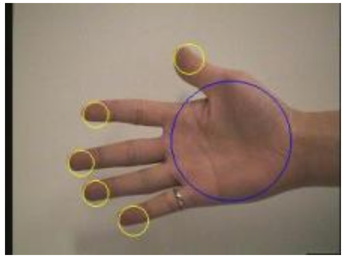
## Alignment

http://hugin.sourceforge.net/tech/

## Light fields

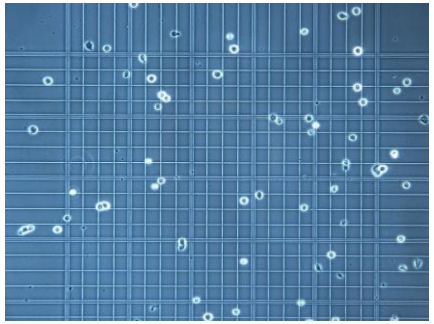
Homework 4

## Fingertip detection



https://www.cim.mcgill.ca/sre/projects/fingertip/

## Counting



http://en.wikipedia.org/wiki/File:Neubauer_improved_with_cells.jpg

## ASCII art

http://fr.wikipedia.org/wiki/Art_ASCII

“Every computer vision problem can be described as a registration problem.”

# Morphological filtering

# Theme for the rest of this lecture

Last time we discussed filtering operations that are both:

- linear
- shift-invariant

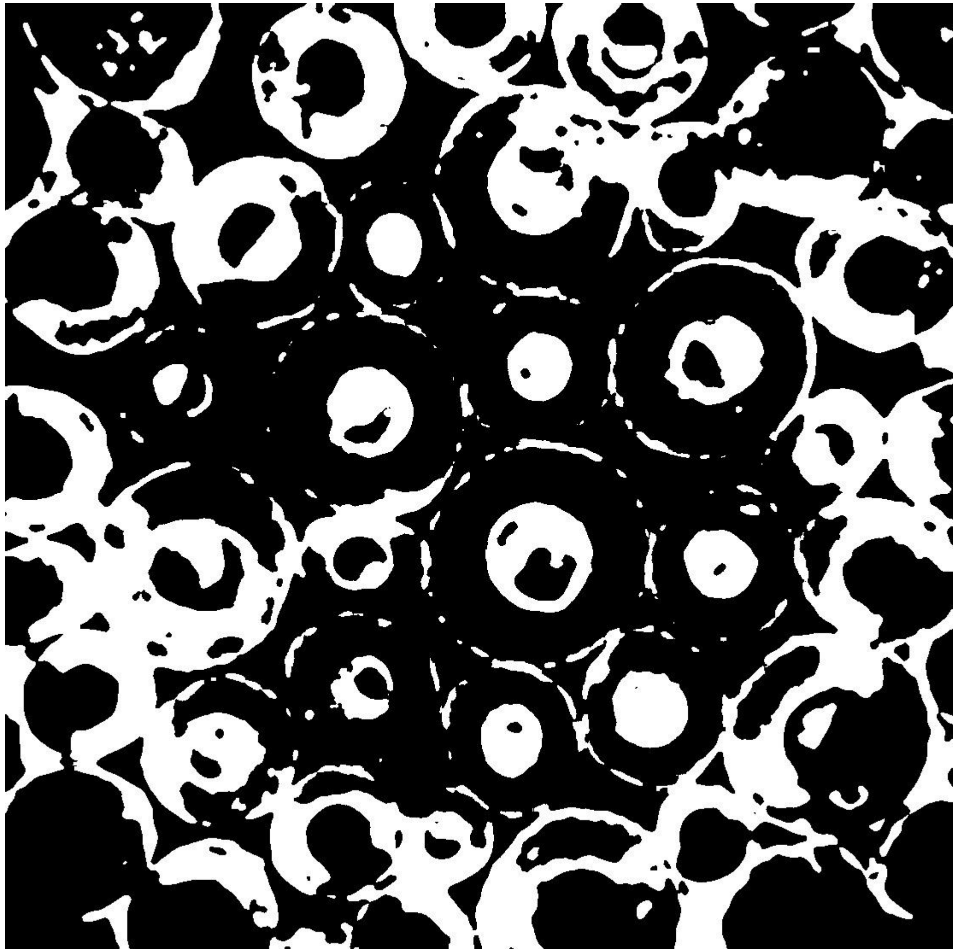
This time we will see filters where we remove one or both of these properties.

# Processing binary images

Binary images are quite common:
- segmentation
- template matching
- text
- thresholding

Mathematical morphology:
- set-theoretic study of binary image processing
- well-studied field with rich history

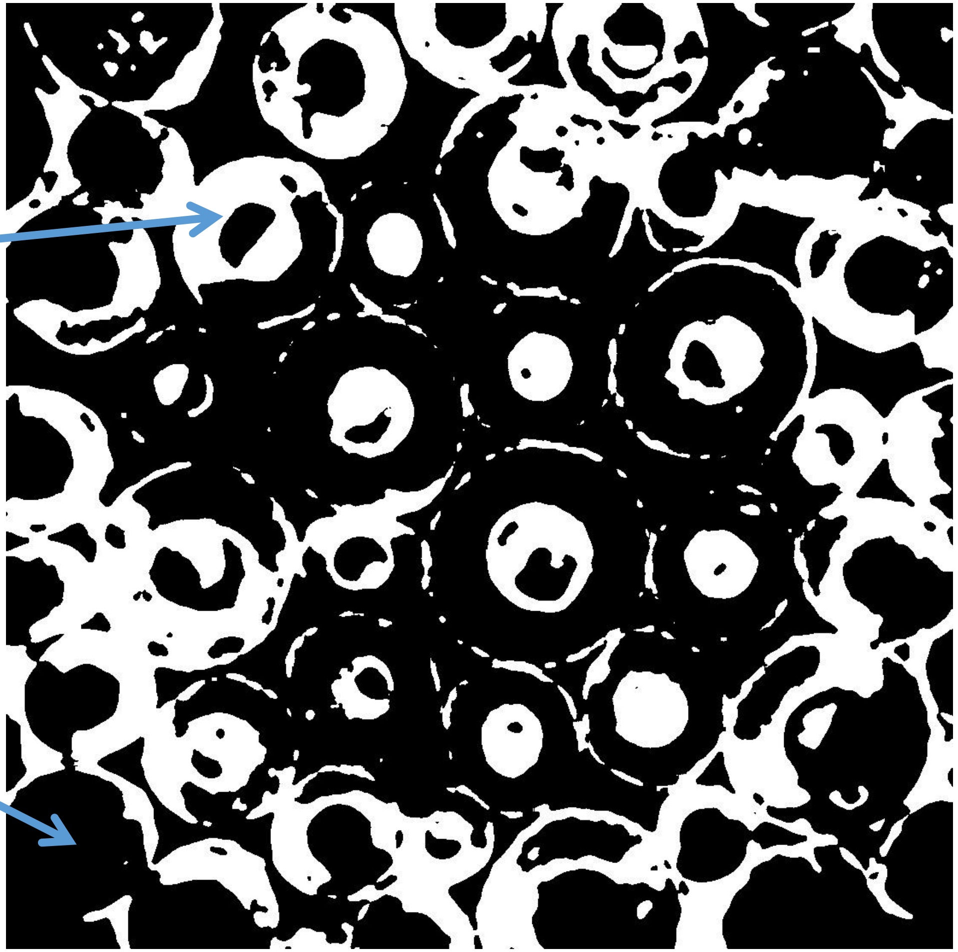
Generalizes to:
- grayscale image filtering
- distance transforms
- diffusion operations



# Representation of binary images

Foreground or object pixels:
- intensity value 1 (white)

Background pixels:
- intensity value 0 (black)



# Some logic preliminaries

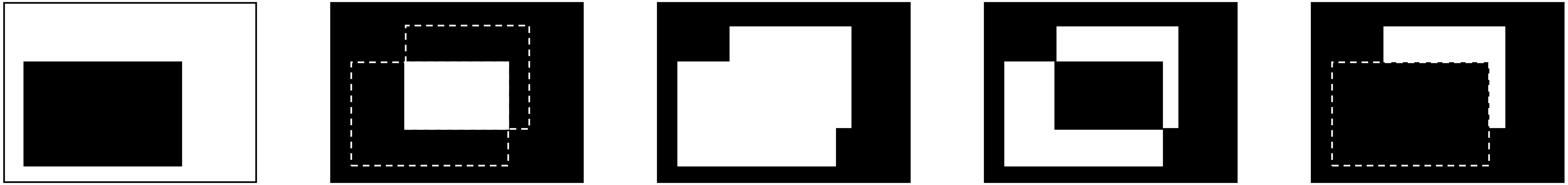
Image A

Image B

| $p$ | $q$ | $p$ AND $q$ (also $p \cdot q$) | $p$ OR $q$ (also $p + q$) | NOT ($p$) (also $\bar{p}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Basic logic operations

How do you create these images as logical combinations of A and B?



# Some logic preliminaries

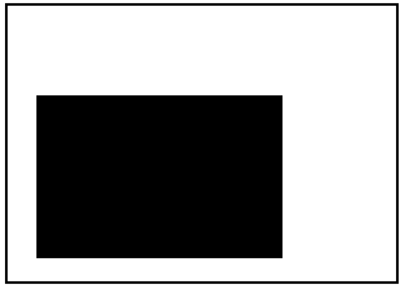
Image A

Image B

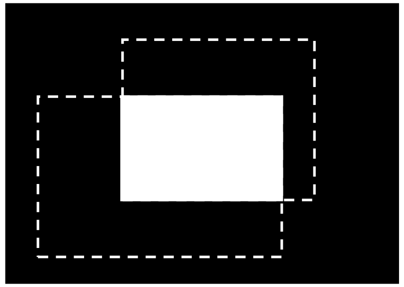
| $p$ | $q$ | $p$ **AND** $q$ (also $p \cdot q$) | $p$ **OR** $q$ (also $p + q$) | **NOT** ($p$) (also $\bar{p}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

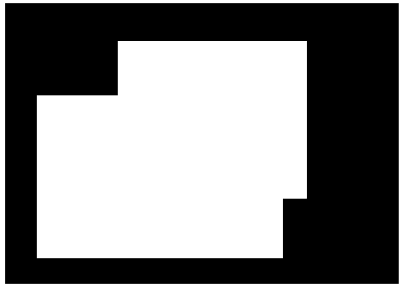
Basic logic operations

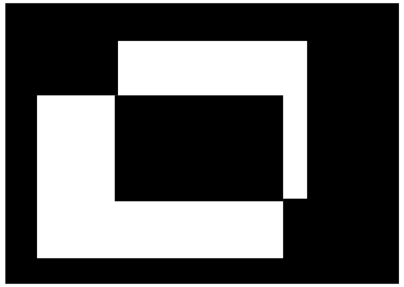
How do you create these images as logical combinations of A and B?

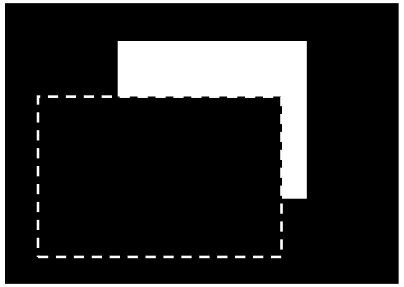


NOT(A)









# Some logic preliminaries

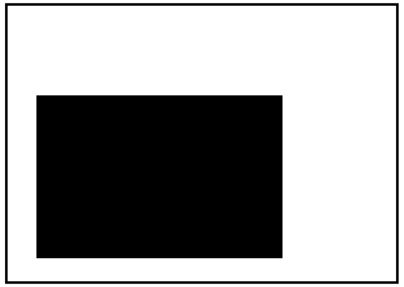

Image A

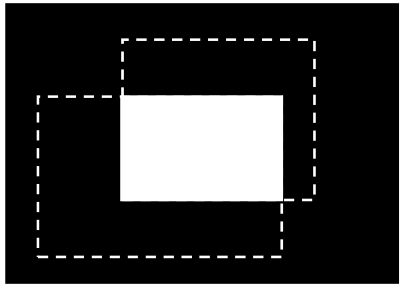

Image B

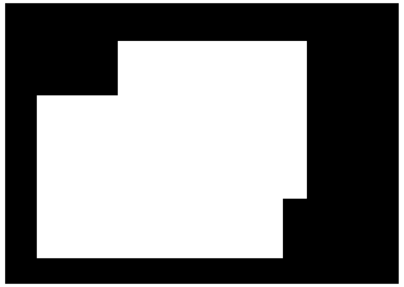
| $p$ | $q$ | $p$ AND $q$ (also $p \cdot q$) | $p$ OR $q$ (also $p + q$) | NOT ($p$) (also $\bar{p}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

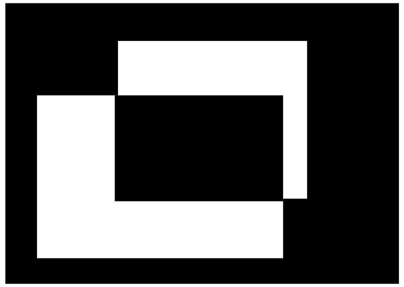
Basic logic operations

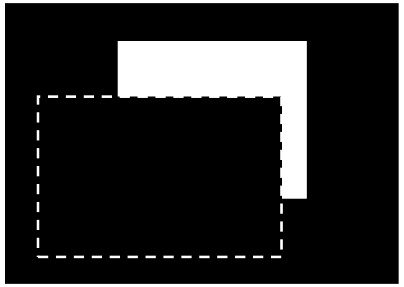
How do you create these images as logical combinations of A and B?


NOT(A)


AND(A,B)







# Some logic preliminaries

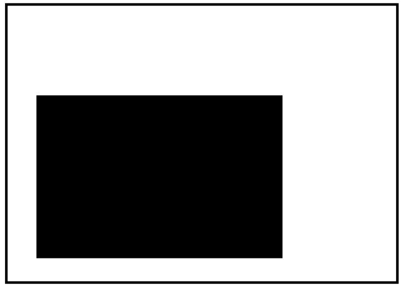
Image A

Image B

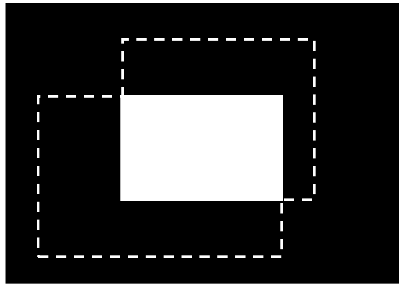
| $p$ | $q$ | $p$ AND $q$ (also $p \cdot q$) | $p$ OR $q$ (also $p + q$) | NOT ($p$) (also $\bar{p}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Basic logic operations

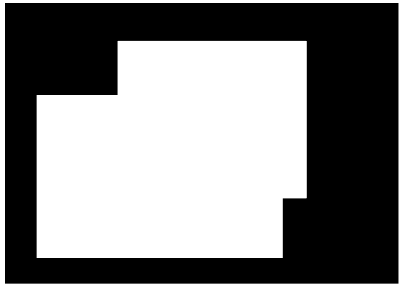
How do you create these images as logical combinations of A and B?



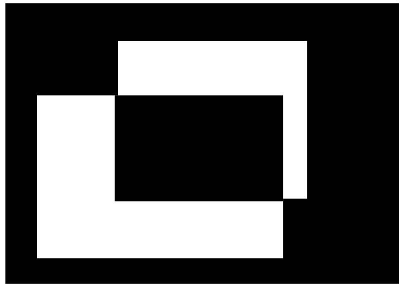
NOT(A)

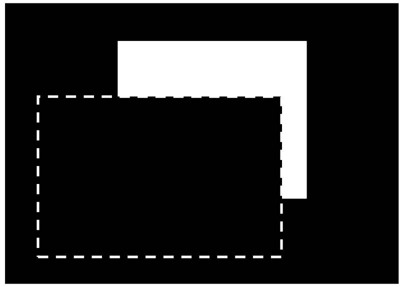


AND(A,B)



OR(A,B)





# Some logic preliminaries

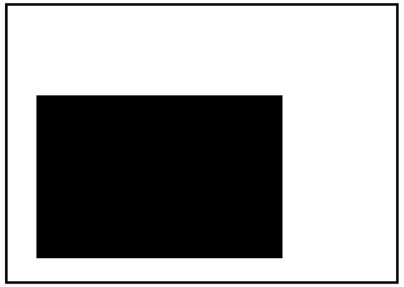
Image A

Image B

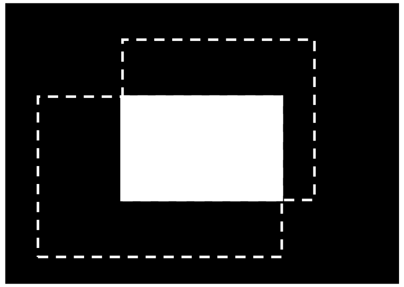
| $p$ | $q$ | $p$ AND $q$ (also $p \cdot q$) | $p$ OR $q$ (also $p + q$) | NOT ($p$) (also $\bar{p}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Basic logic operations

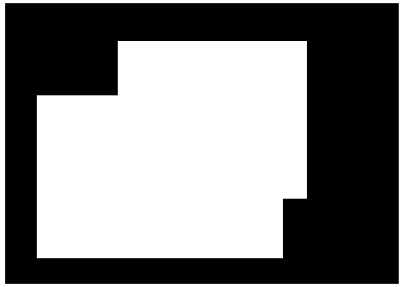
How do you create these images as logical combinations of A and B?



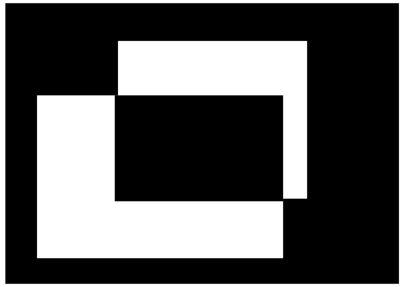
NOT(A)



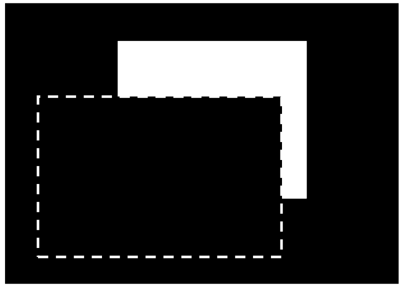
AND(A,B)



OR(A,B)



XOR(A,B)



# Some logic preliminaries

Image A

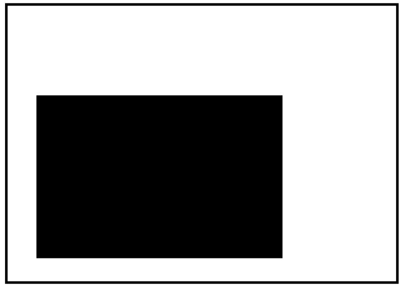
Image B

| $p$ | $q$ | $p$ AND $q$ (also $p \cdot q$) | $p$ OR $q$ (also $p + q$) | NOT ($p$) (also $\bar{p}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

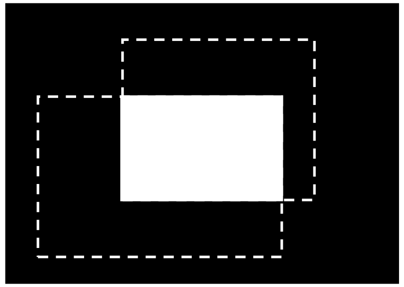
Basic logic operations

How do you create these images as logical combinations of A and B?

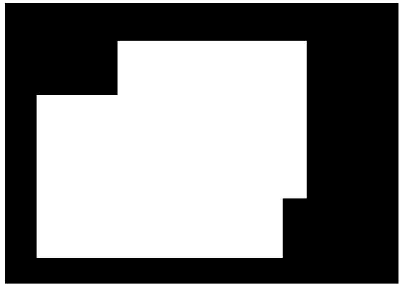
Notation: B-A



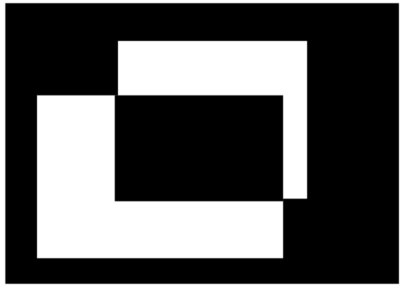
NOT(A)



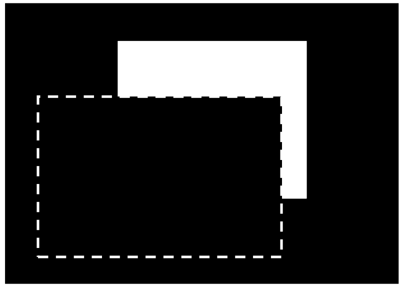
AND(A,B)



OR(A,B)



XOR(A,B)



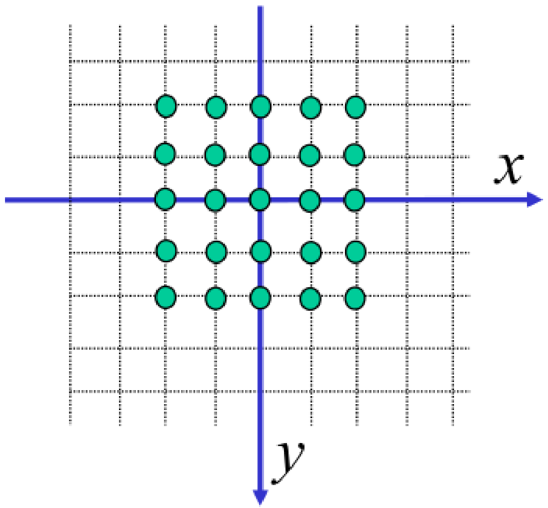
AND(NOT(A), B)

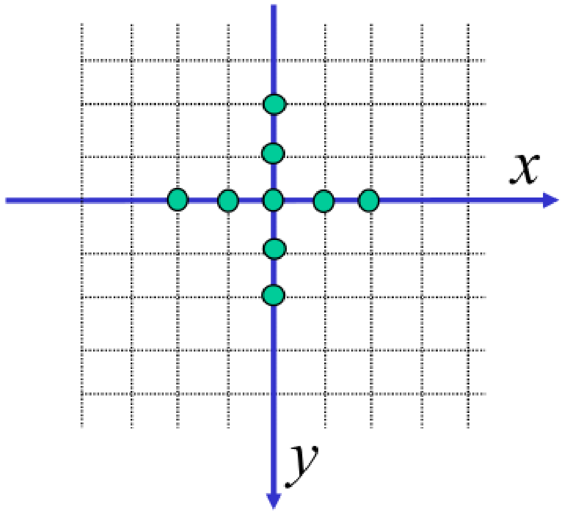
# Structuring element

Basically the binary equivalent of a kernel

- specifies a neighborhood around a binary pixel

5x5 square




crosss

For each structuring element, we can specify a corresponding windowing operator:

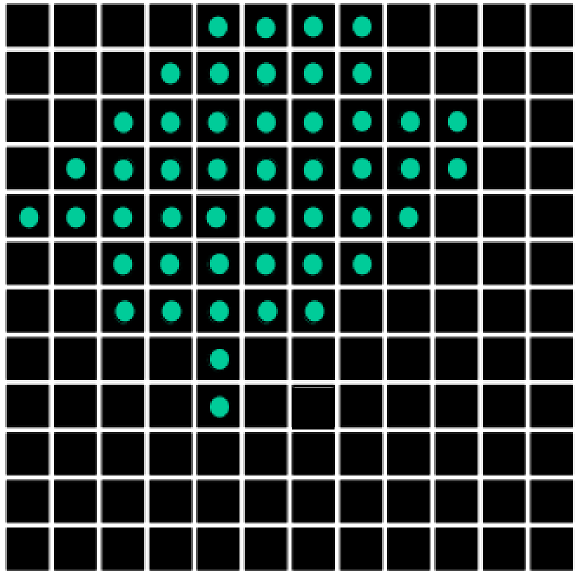
$$W\{f[x,y]\} = \{f[x-x',y-y'] : [x',y'] \in \Pi_{xy}\}$$

structuring element

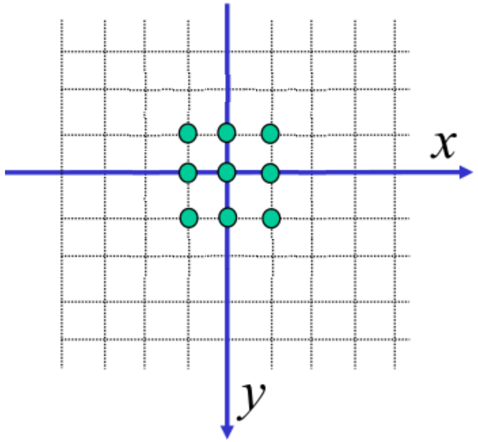
# Basic morphological filters

Dilation: expand a binary image based on some structuring element

$$g\left[x,y\right]=OR\left[W\left\{f\left[x,y\right]\right\}\right]:=dilate\left(f,W\right)$$



$\oplus$



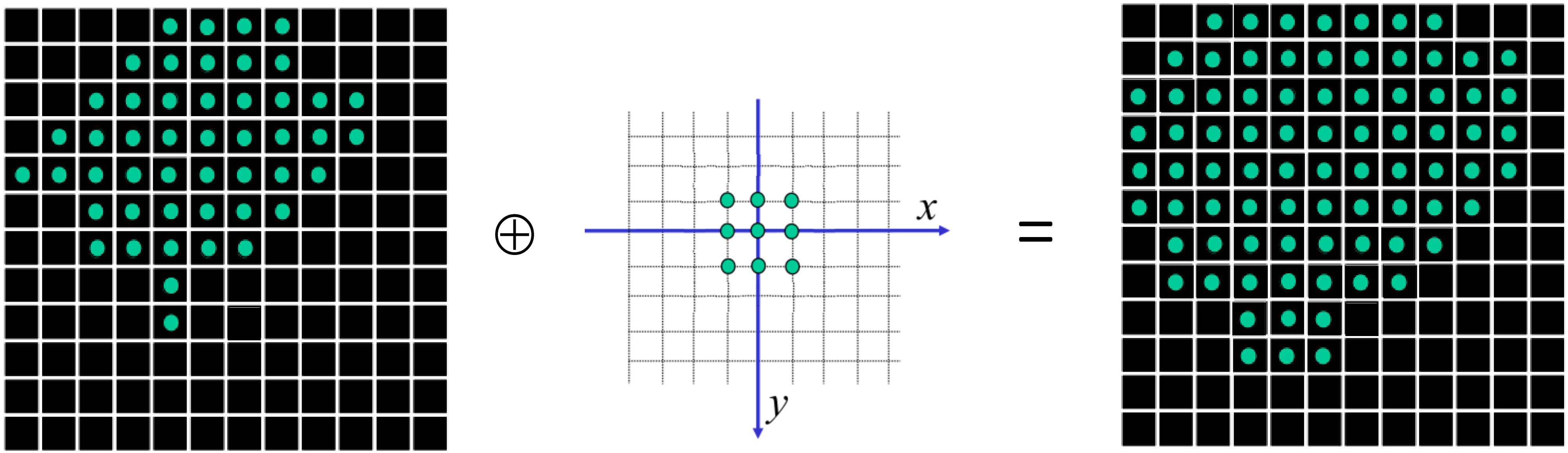

=

What does the output look like?

# Basic morphological filters

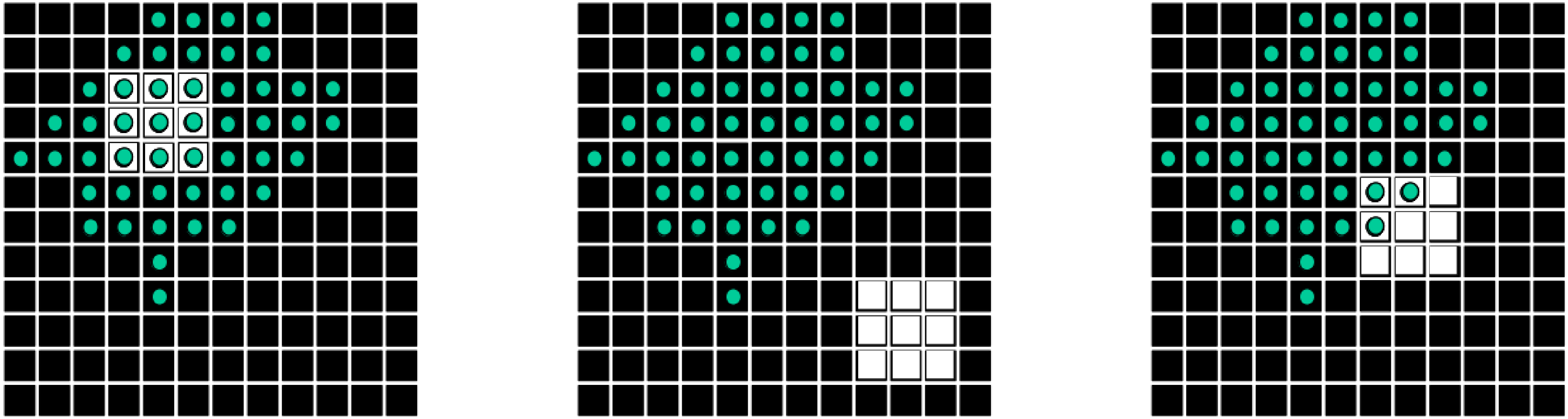
Dilation: expand a binary image based on some structuring element

$$g[x,y] = OR\left[W\left\{f[x,y]\right\}\right] := dilate(f,W)$$



# Performing dilation

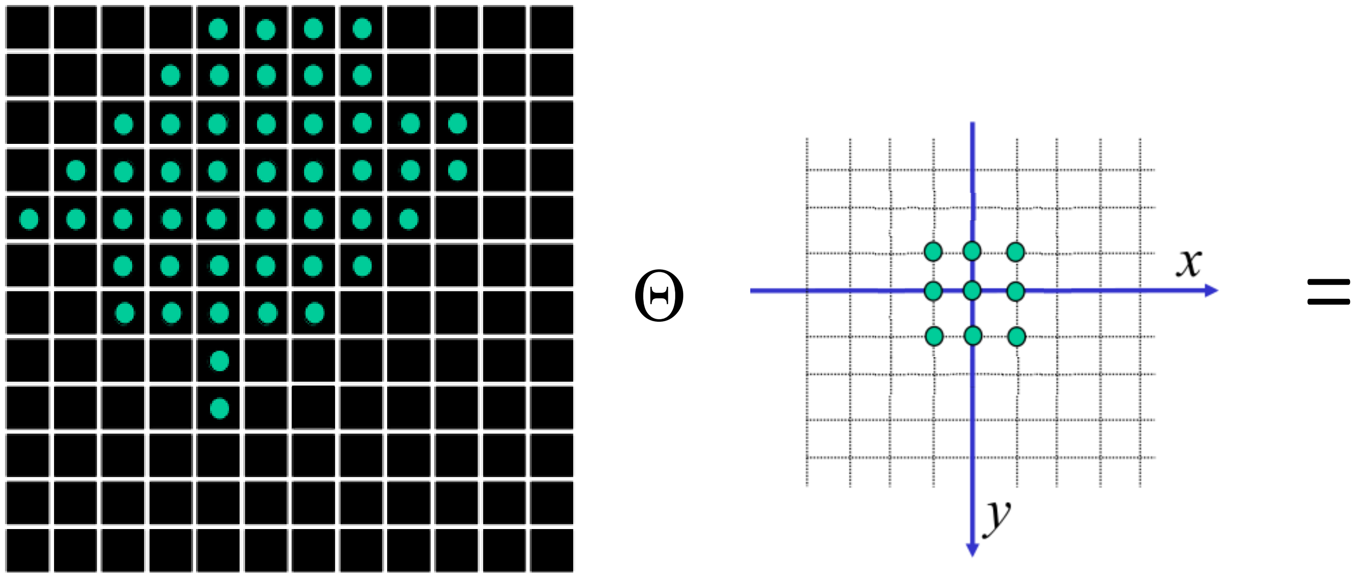
Shift structuring element to every pixel, then compute the OR operator in the neighborhood defined by the structuring element



# Basic morphological filters

Erosion: shrink a binary image based on some structuring element

$$g[x,y] = AND\left[W\left\{f[x,y]\right\}\right] := erode(f, W)$$

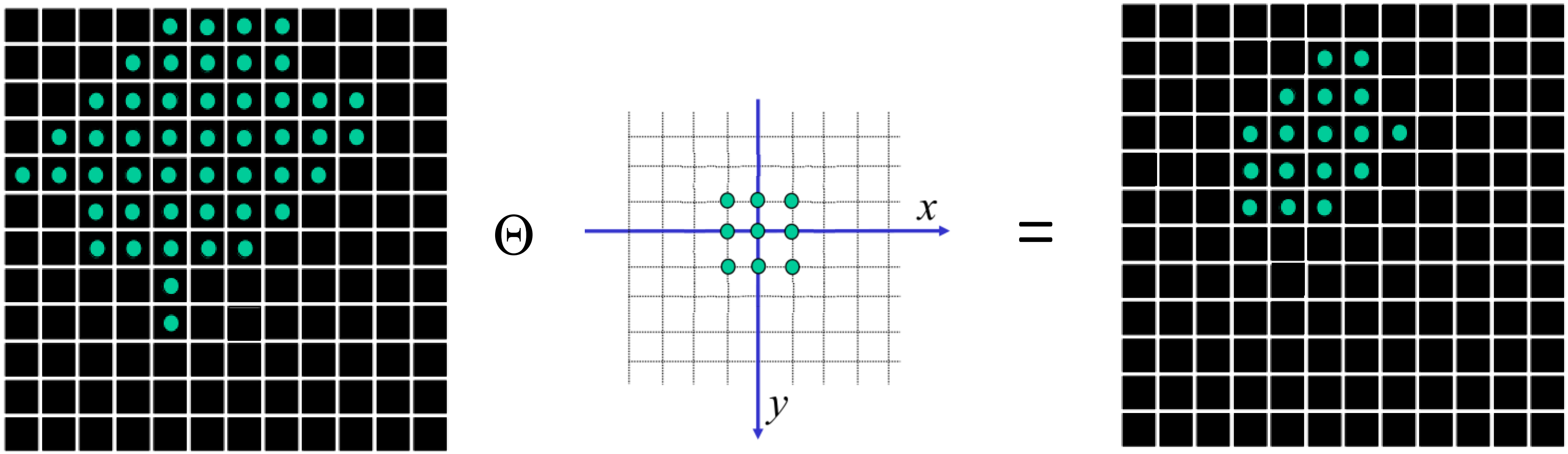



What does the output look like?

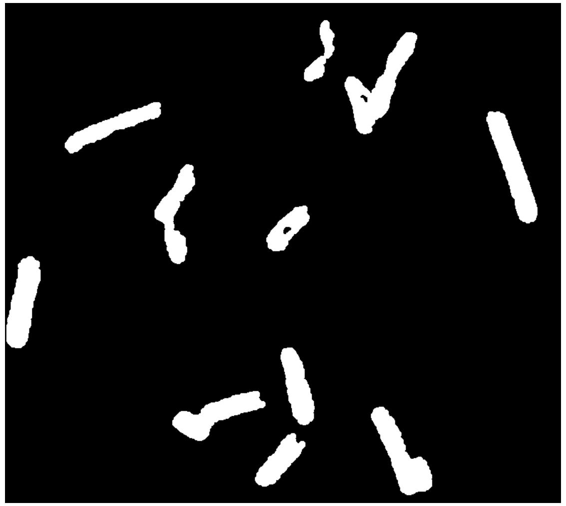
# Basic morphological filters

Erosion: shrink a binary image based on some structuring element

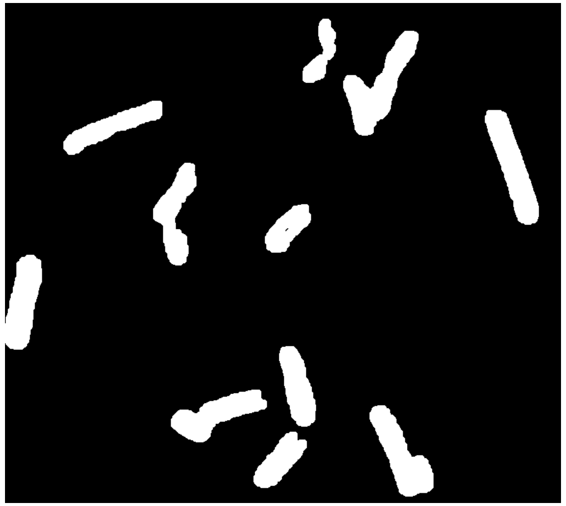
$$g[x,y] = AND\left[W\left\{f[x,y]\right\}\right] := erode(f, W)$$



# Example

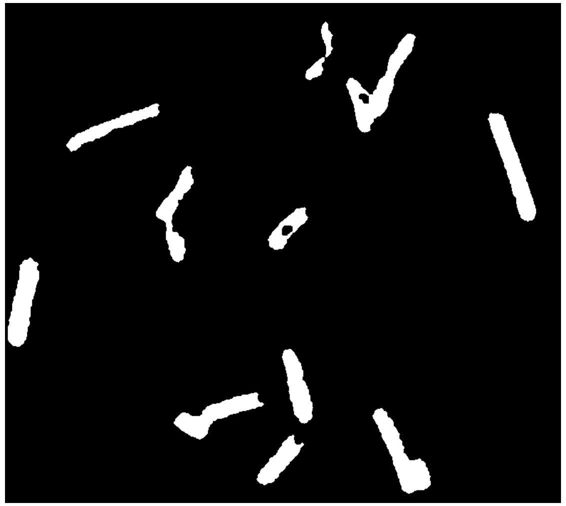


original



dilation with 3 x 3

dilation with 7 x 7

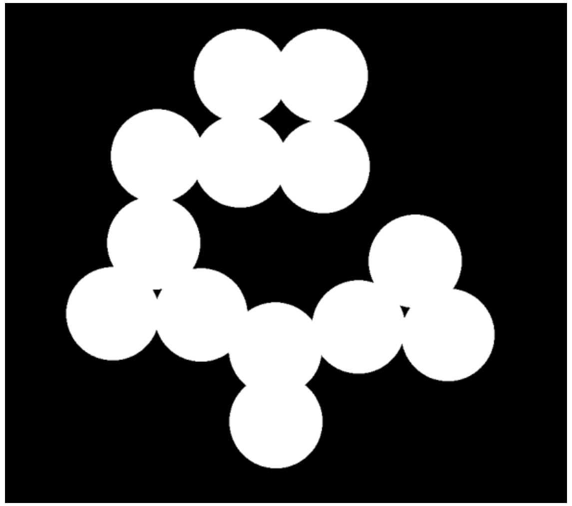


erosion with 3 x 3

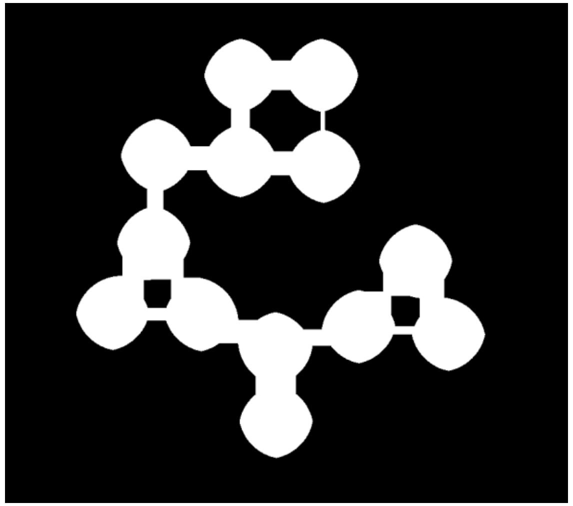
erosion with 7 x 7

# Example

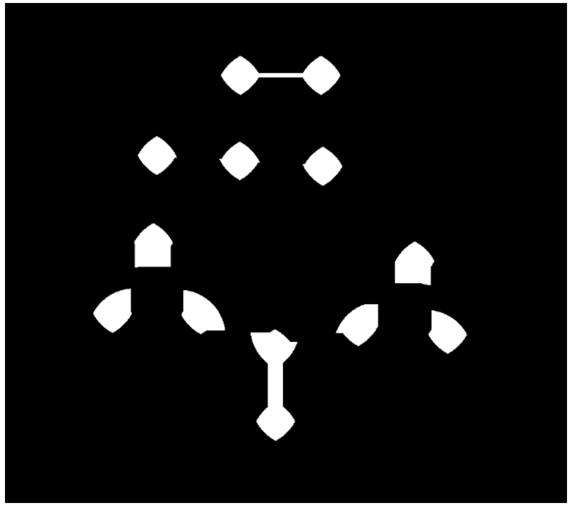
Erosion with structuring elements of different shapes



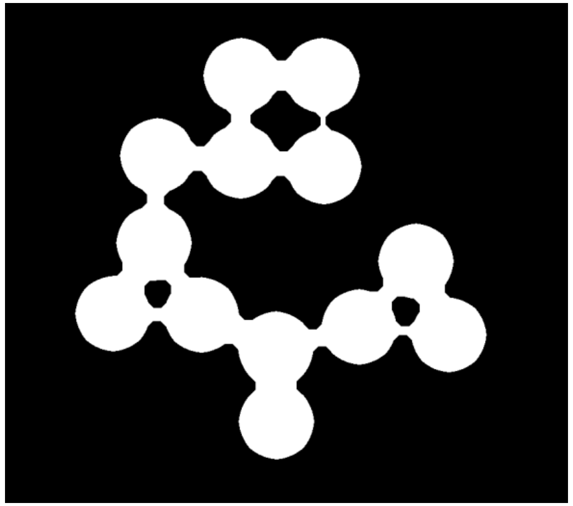
original



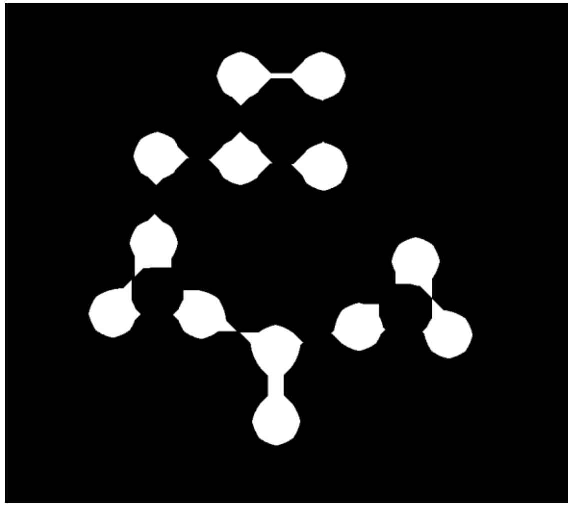
30 x 30 square



70 x 70 square

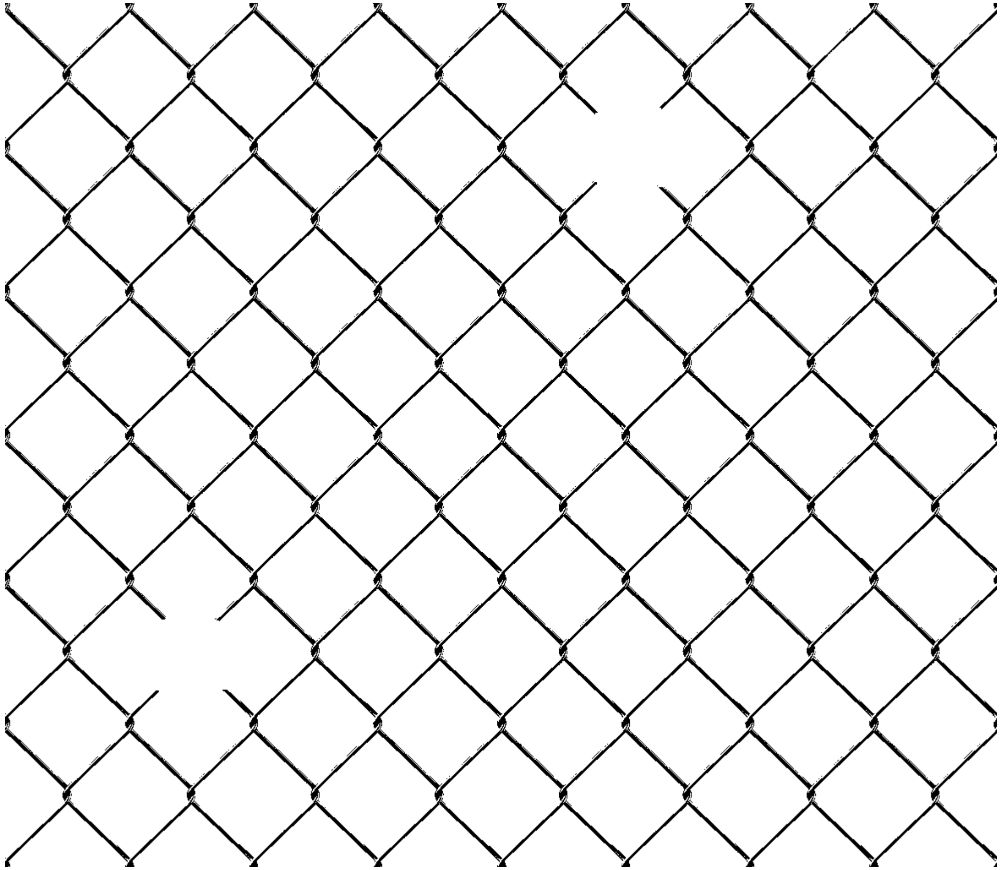


diam = 30 circle



diam = 70 circle

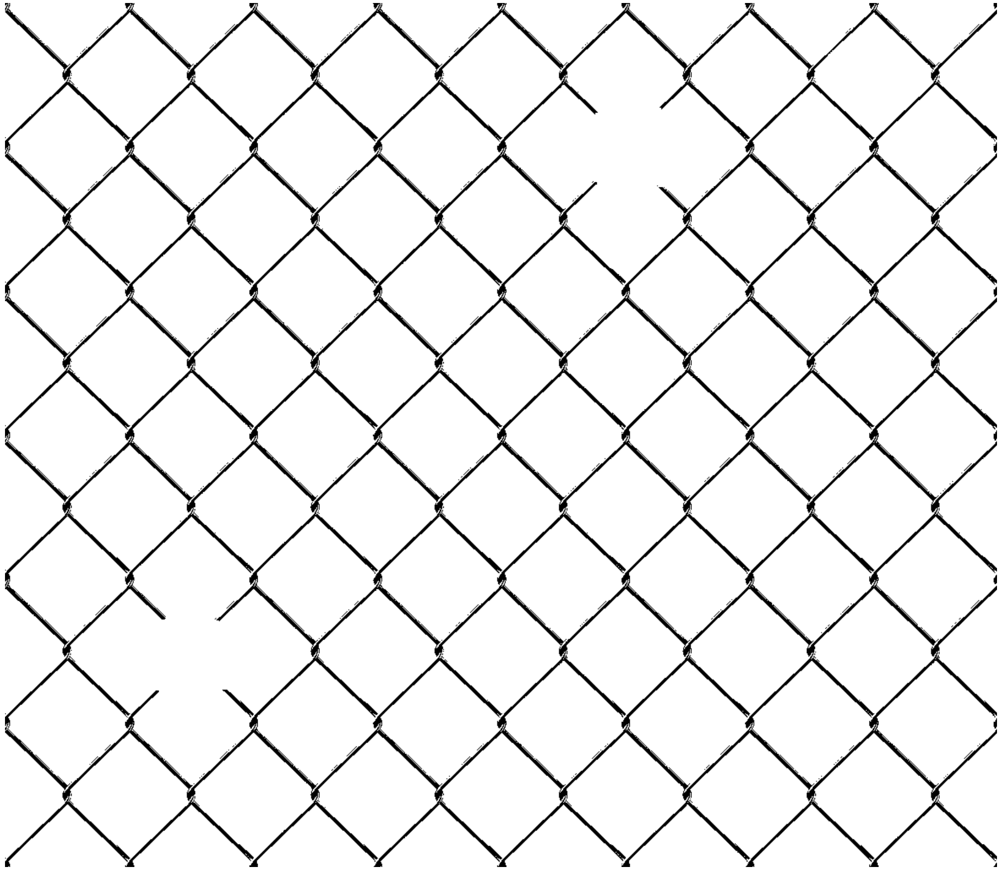
# Template matching using morphological filters



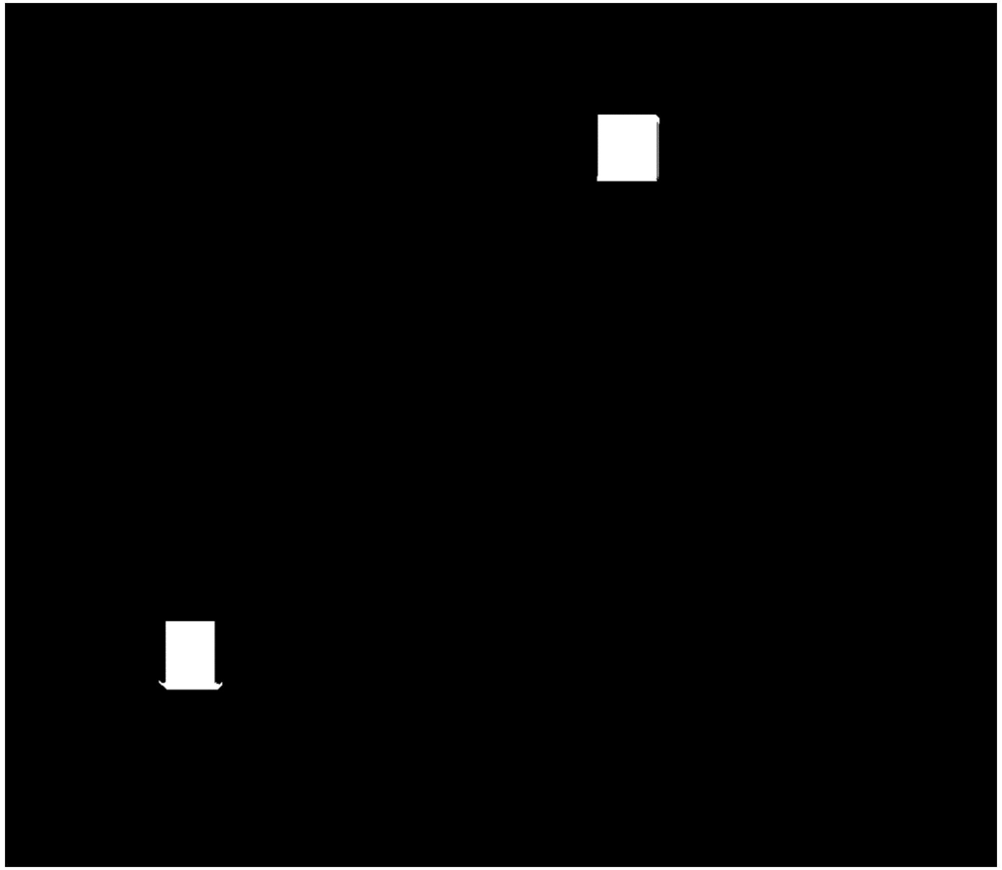
binary fence image

How to detect the gaps in the fence?

# Template matching using morphological filters



binary fence image



erosion with 150 x 150 cross

# Template matching using morphological filters

INTEREST-POINT DETECTION

Feature extraction typically starts by finding the salient interest points in the image. For robust image matching, we desire interest points to be repeatable under perspective transformations (or, at least, scale changes, rotation, and translation) and real-world lighting variations. An example of feature extraction is illustrated in Figure 3. To achieve scale invariance, interest points are typically computed at multiple scales using an image pyramid [15]. To achieve rotation invariance, the patch around each interest point is canonically oriented in the direction of the dominant gradient. Illumination changes are compensated by normalizing the mean and standard deviation of the pixels of the gray values within each patch [16].

binarized text

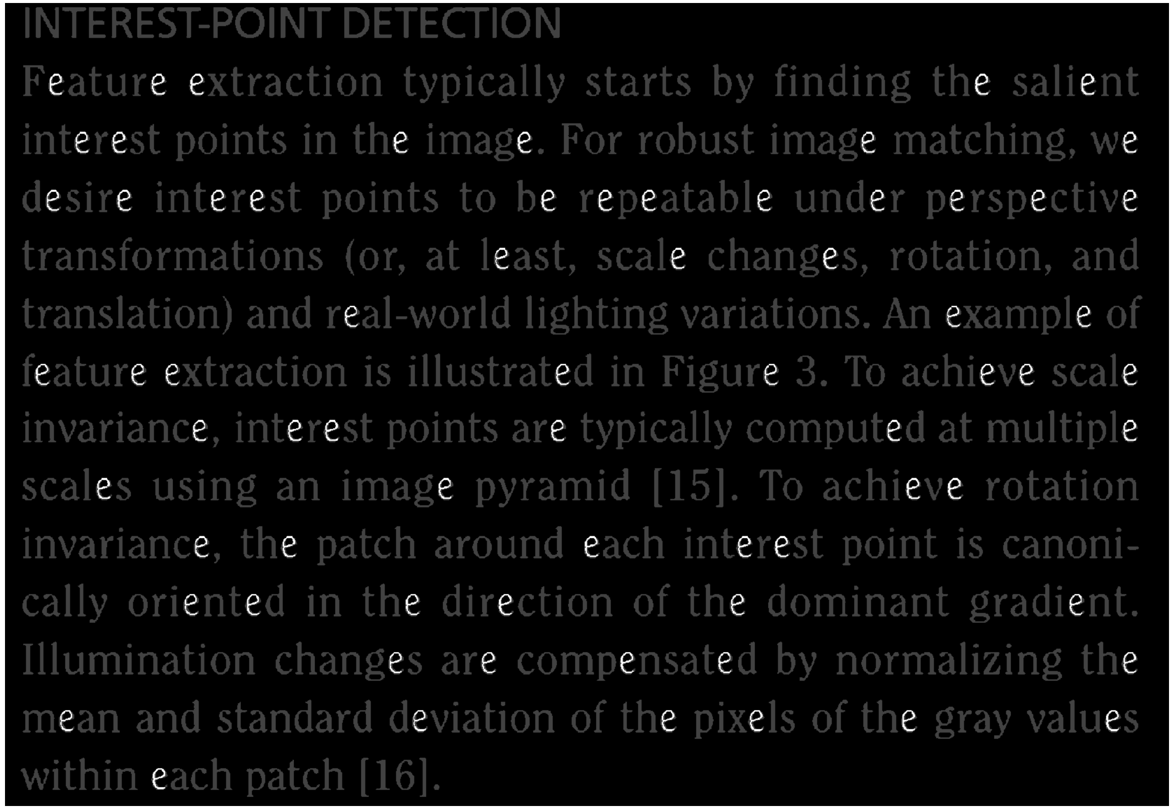
How to detect all instances of the letter “e”?

# Template matching using morphological filters

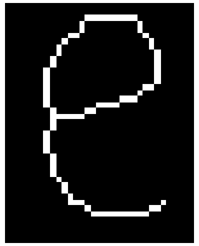
INTERVIEW-POINT DETECTION

Feature extraction typically starts by finding the salient interest points in the image. For robust image matching, we desire interest points to be repeatable under perspective transformations (or, at least, scale changes, rotation, and translation) and real-world lighting variations. An example of feature extraction is illustrated in Figure 3. To achieve scale invariance, interest points are typically computed at multiple scales using an image pyramid [15]. To achieve rotation invariance, the patch around each interest point is canonically oriented in the direction of the dominant gradient. Illumination changes are compensated by normalizing the mean and standard deviation of the pixels of the gray values within each patch [16].

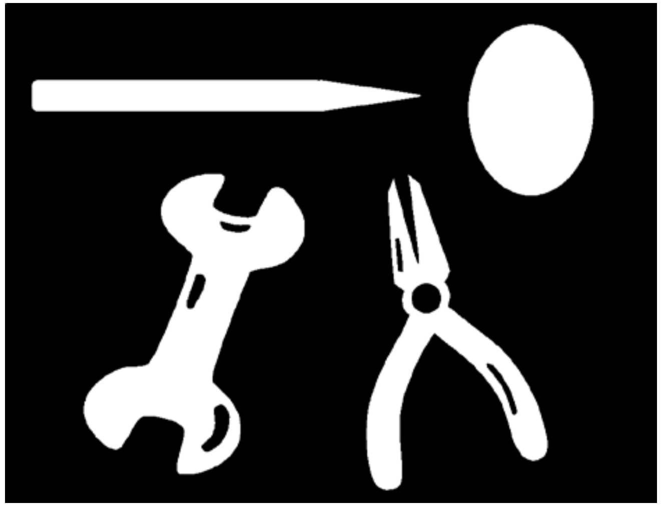
binarized text



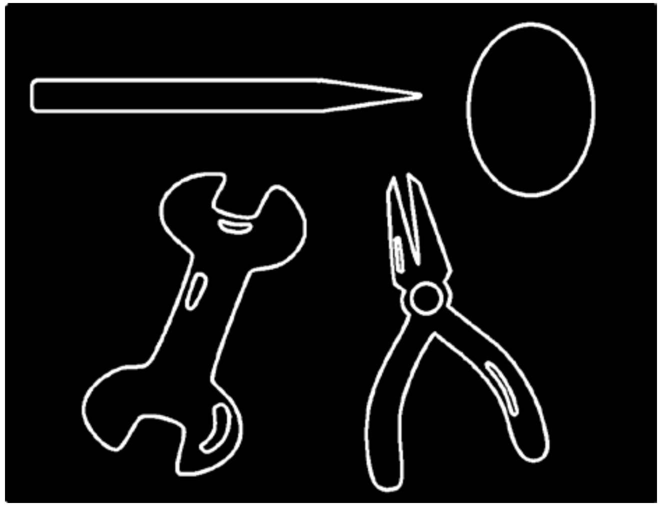
erosion with structuring element



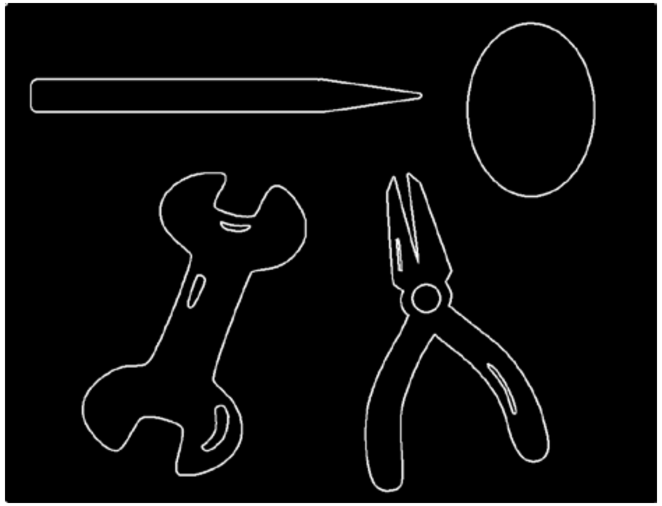
# Edge detection using morphological filters



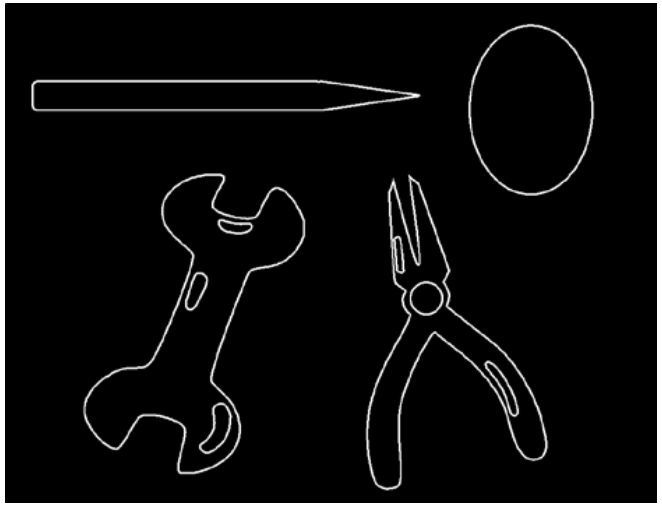
original



dilated - eroded

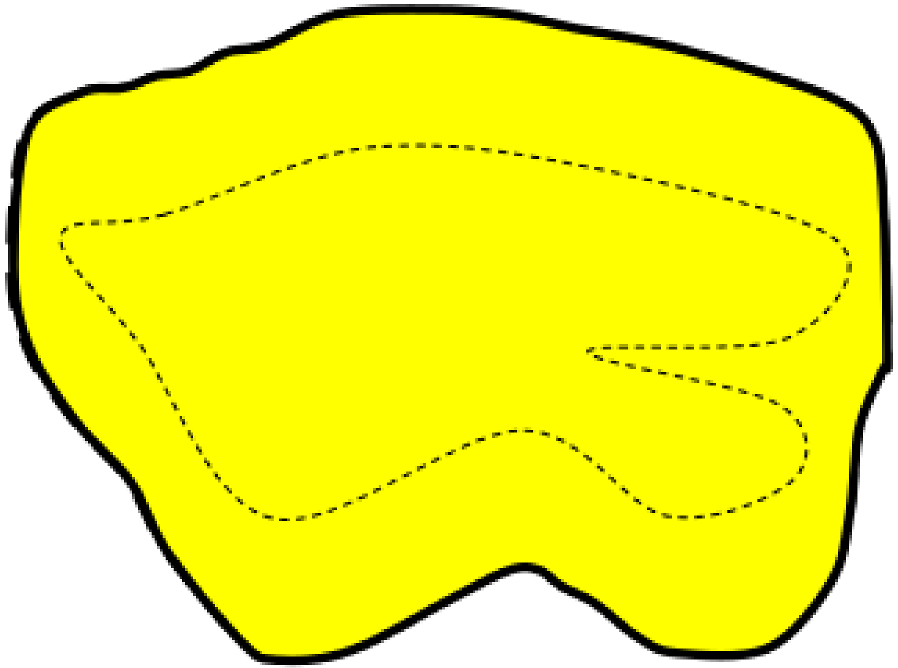


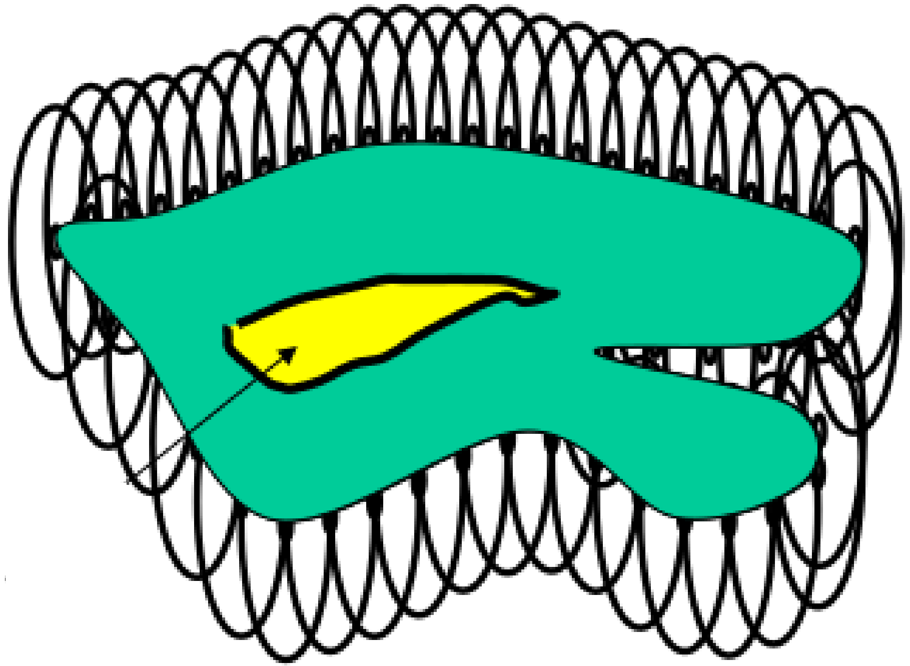
dilated - original



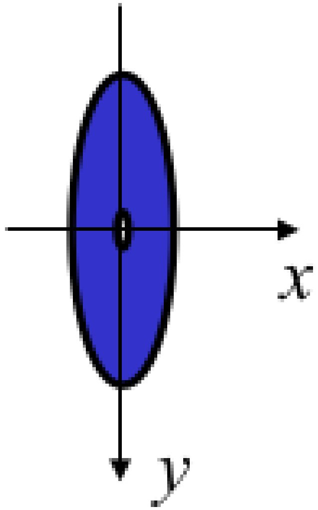
original - eroded

# Set-theoretic interpretation





structuring element




Dilation: Minkowski set addition

$$g[x,y] = OR\left[W\{f[x,y]\}\right] := dilate(f,W)$$

Erosion: Minkowski set subtraction

$$g[x,y] = AND\left[W\{f[x,y]\}\right] := erode(f,W)$$

# Which of the following is true?

Assume we always use the same structuring element.

- Eroding and then dilating an image returns the original image.

- First eroding and then dilating an image produces the same result as first dilating and then eroding the image.

# Which of the following is true?

Assume we always use the same structuring element.

- Eroding and then dilating an image returns the original image.

  Nope.

- First eroding and then dilating an image produces the same result as first dilating and then eroding the image.

  Nope.

“Dual” morphological operations generally neither commute nor are inverses of each other.

# More morphological filters

Closing: first dilate then erode image

$$close(f, W) = erode(dilate(f, W), W)$$

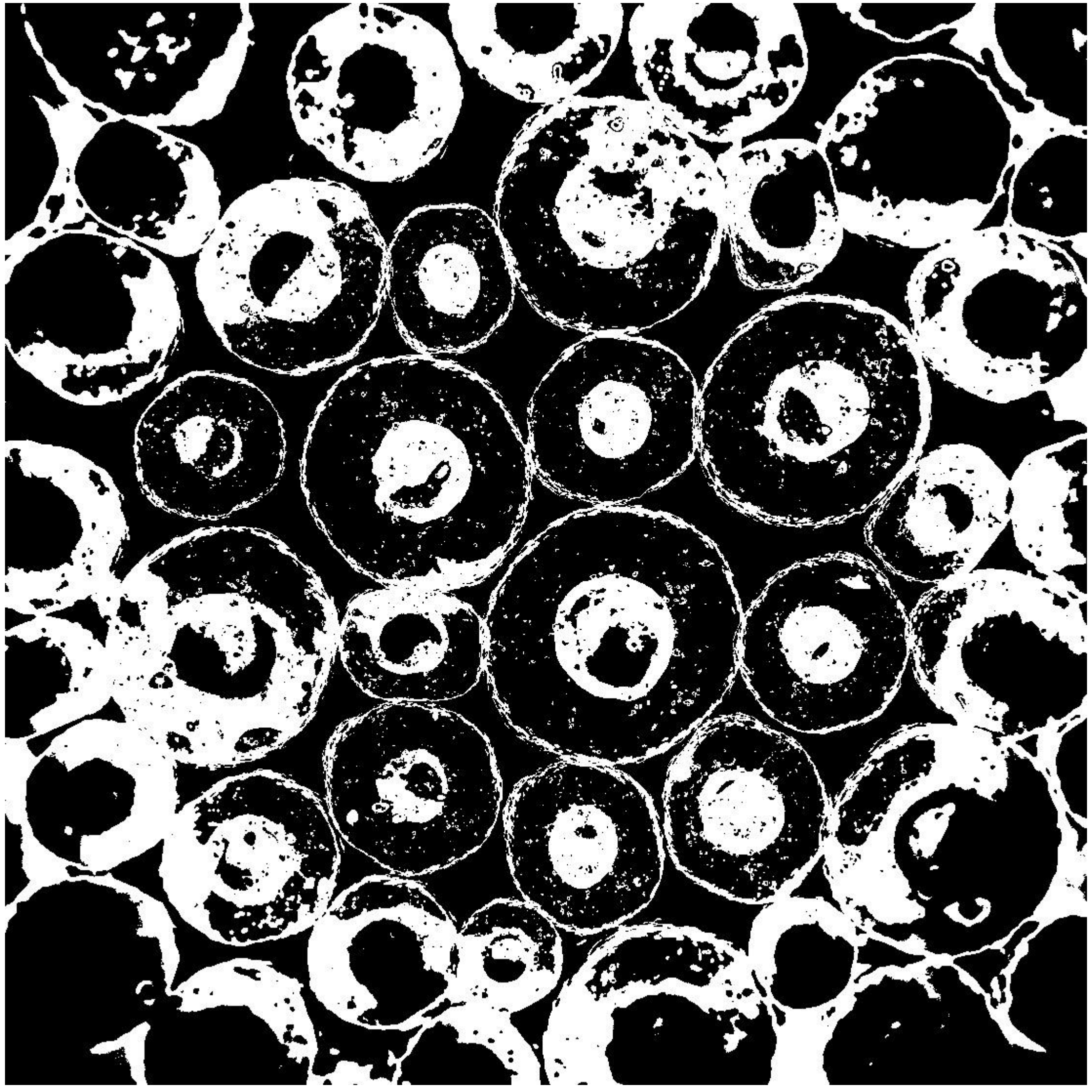
Opening: first erode then dilate image

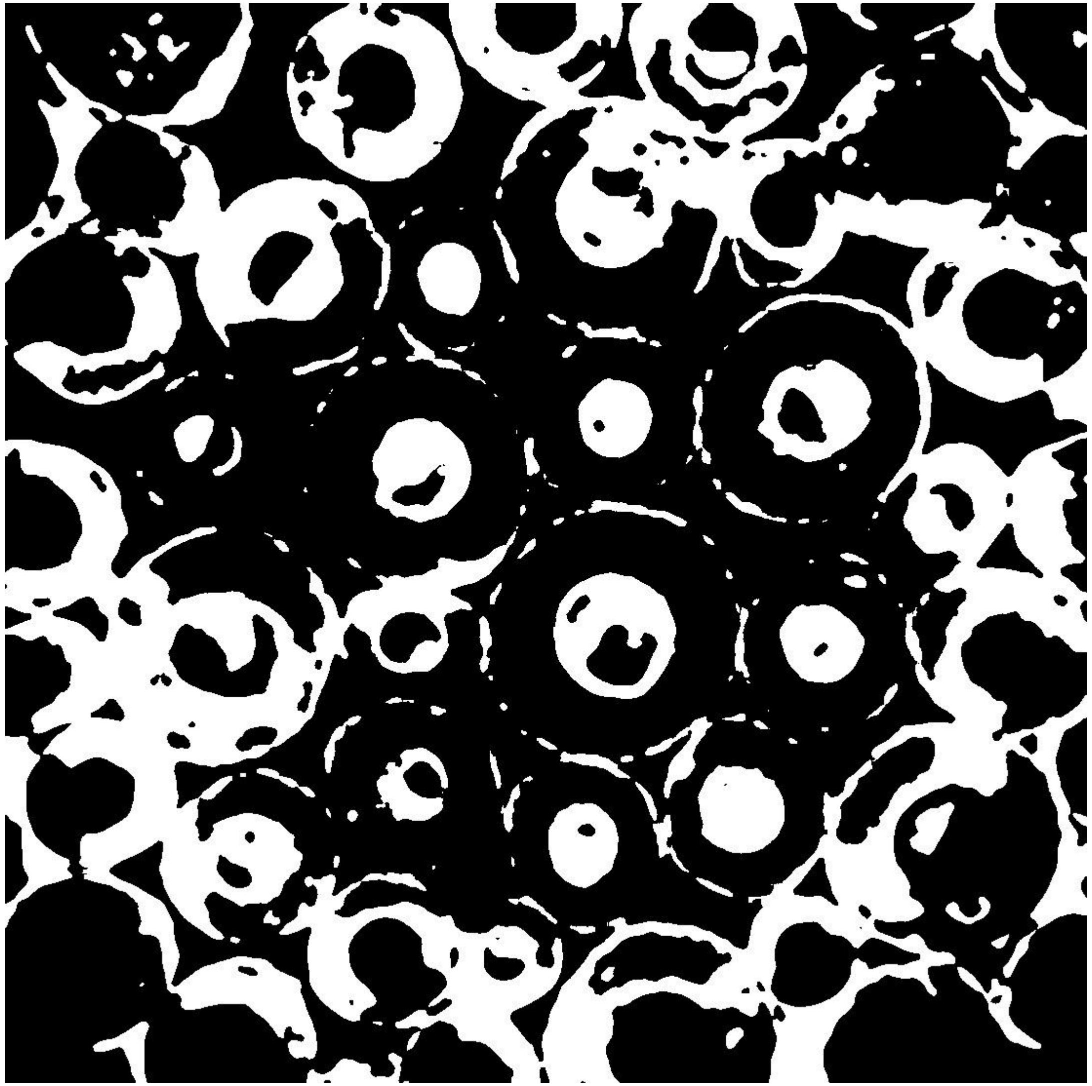
$$open(f, W) = dilate(erode(f, W), W)$$

Majority: replace pixel with majority value in neighborhood

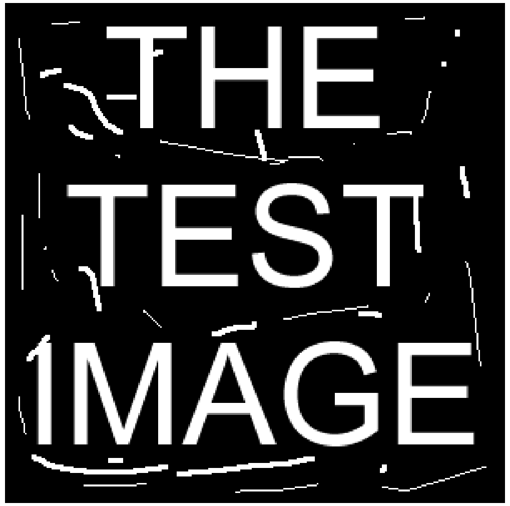
$$g[x, y] = MAJ[W\{f[x, y]\}] := majority(f, W)$$

# Denoising using majority operation

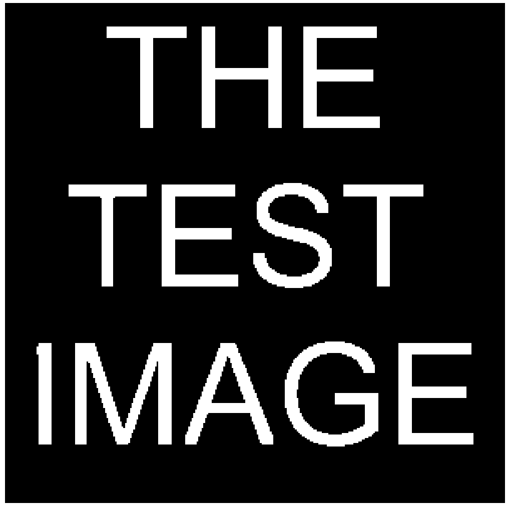




# Opening and closing

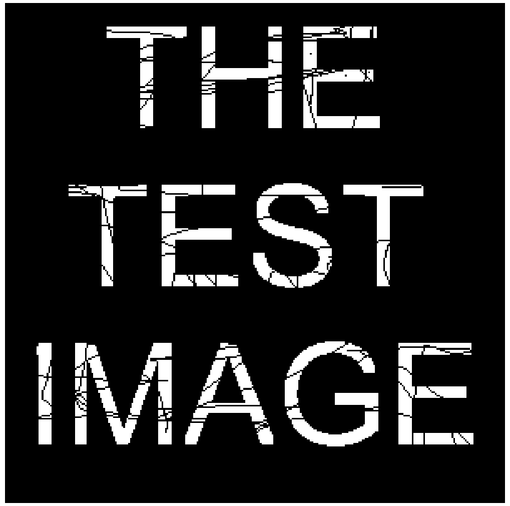



original

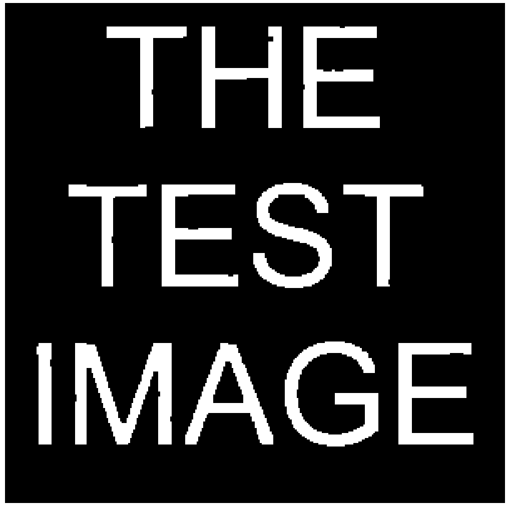



opening


erosion

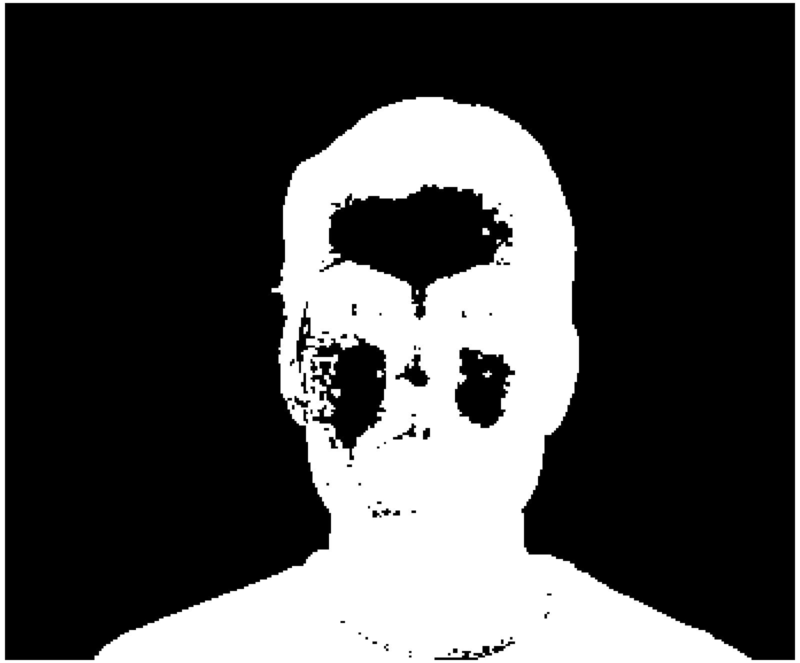



original

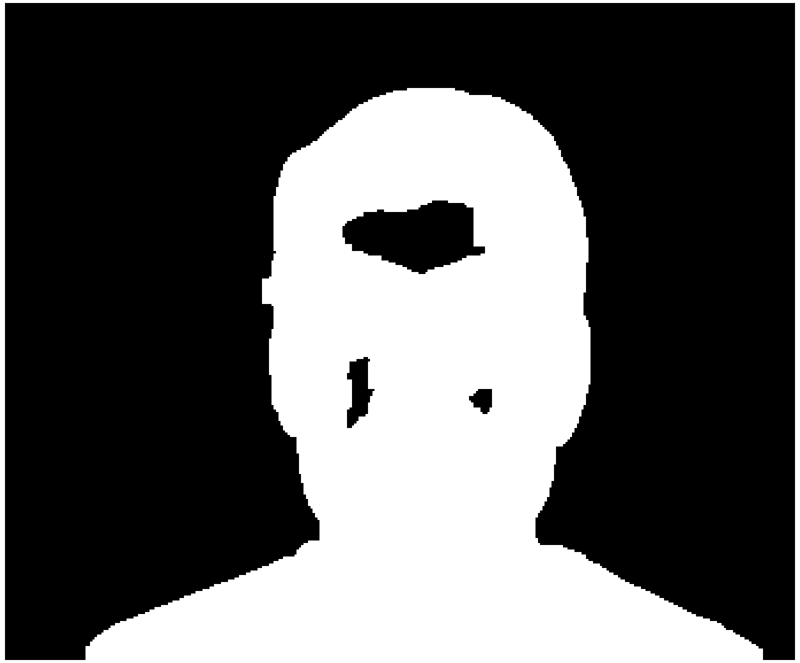



closing

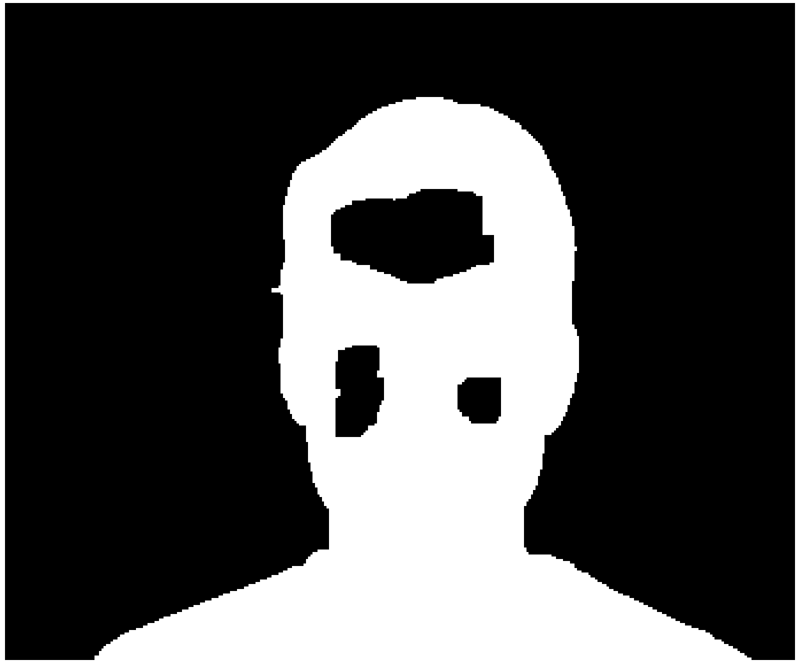
# Small hole closing



original



dilation



closing

# Are morphological filters:

Linear?

Shift-invariant?

# Are morphological filters:

Linear?

- No.

Shift-invariant?

- Yes.

We can prove that morphological filters are equivalent generalized forms of convolution, where maximum (supremum) replaces summation, and additions replace products:

$$g\left[x,y\right]=\sup_{\alpha,\beta}\left\{f\left[x-\alpha,y-\beta\right]+w\left[\alpha,\beta\right]\right\}=\sup_{\alpha,\beta}\left\{w\left[x-\alpha,y-\beta\right]+f\left[\alpha,\beta\right]\right\}$$

# How to generalize morphological filters to grayscale images?

# How to generalize morphological filters to grayscale images?

General theory based on image level sets:

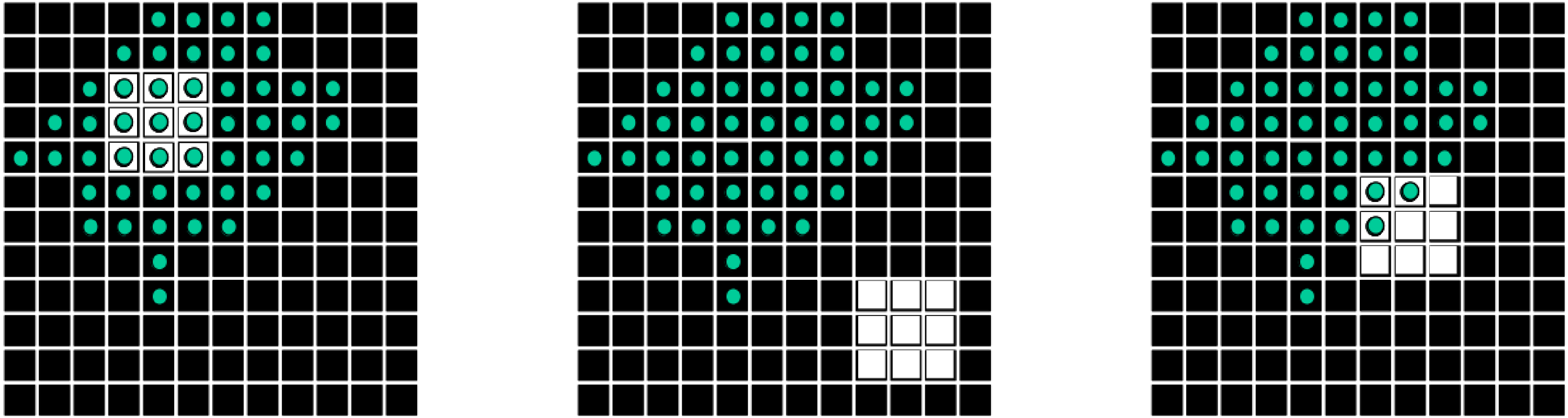
- Separate image into multiple binary images, by thresholding at each possible intensity level (“level sets”).
- Apply morphological filter to each level set image.
- Combine results using maximum across level set images.

We will see one simple instance of this.

# Rank filters

# Replacing logical operators

Can you think of a function of the binary pixel values in an image neighborhood that produces the same result as the logical OR operator?



# Replacing logical operators

Dilation: $g[x,y] = OR\left[W\{f[x,y]\}\right] := dilate(f,W)$ $\longrightarrow$ Replace OR with MAX

Erosion: $g[x,y] = AND\left[W\{f[x,y]\}\right] := erode(f,W)$ $\longrightarrow$ Replace AND with ?

Majority: $g[x,y] = MAJ\left[W\{f[x,y]\}\right] := majority(f,W)$ $\longrightarrow$ Replace MAJ with ?

# Replacing logical operators

Dilation: $g[x,y] = OR[W\{f[x,y]\}] := dilate(f,W)$ $\longrightarrow$ Replace OR with MAX

Erosion: $g[x,y] = AND[W\{f[x,y]\}] := erode(f,W)$ $\longrightarrow$ Replace AND with MIN

Majority: $g[x,y] = MAJ[W\{f[x,y]\}] := majority(f,W)$ $\longrightarrow$ Replace MAJ with ?

# Replacing logical operators

Dilation: $$g[x,y] = OR\left[W\{f[x,y]\}\right] := dilate(f,W)$$ $\longrightarrow$ Replace OR with MAX

Erosion: $$g[x,y] = AND\left[W\{f[x,y]\}\right] := erode(f,W)$$ $\longrightarrow$ Replace AND with MIN

Majority: $$g[x,y] = MAJ\left[W\{f[x,y]\}\right] := majority(f,W)$$ $\longrightarrow$ Replace MAJ with MEDIAN

Given these replacements, how would you generalize these filters to grayscale images?

# Rank filters

Grayscale dilation $\longrightarrow$ Max filtering

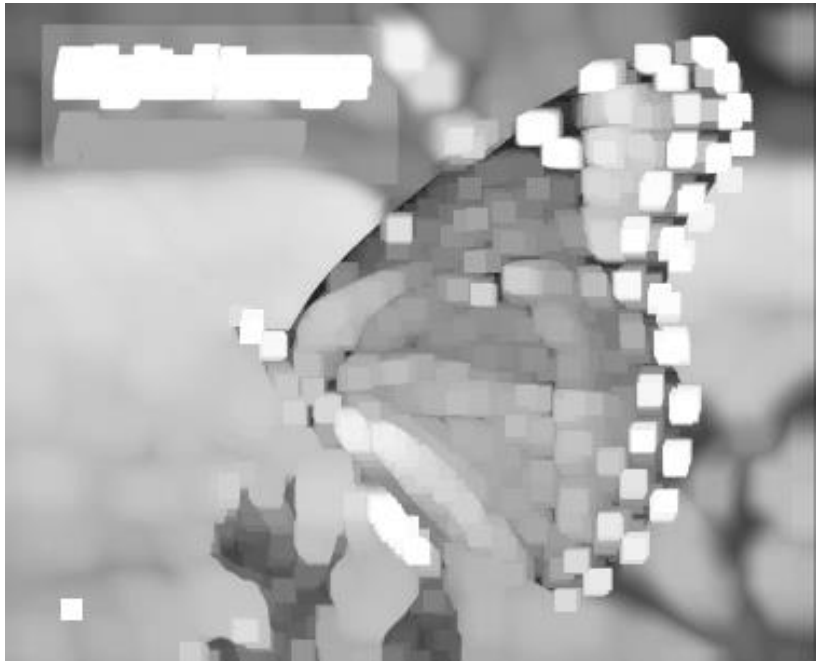
Grayscale erosion $\longrightarrow$ Min filtering

Grayscale majority $\longrightarrow$ Median filtering

- Are these filters linear, shift invariant, neither, or both?
- How would you generalize opening and closing to grayscale images?

# Min and max filtering example


original



dilation (max filtering)

erosion (min filtering)

# Effect of structuring element


original

disk

diamond

20-degree line

2 horizontal lines

9 points

# Morphological edge detection

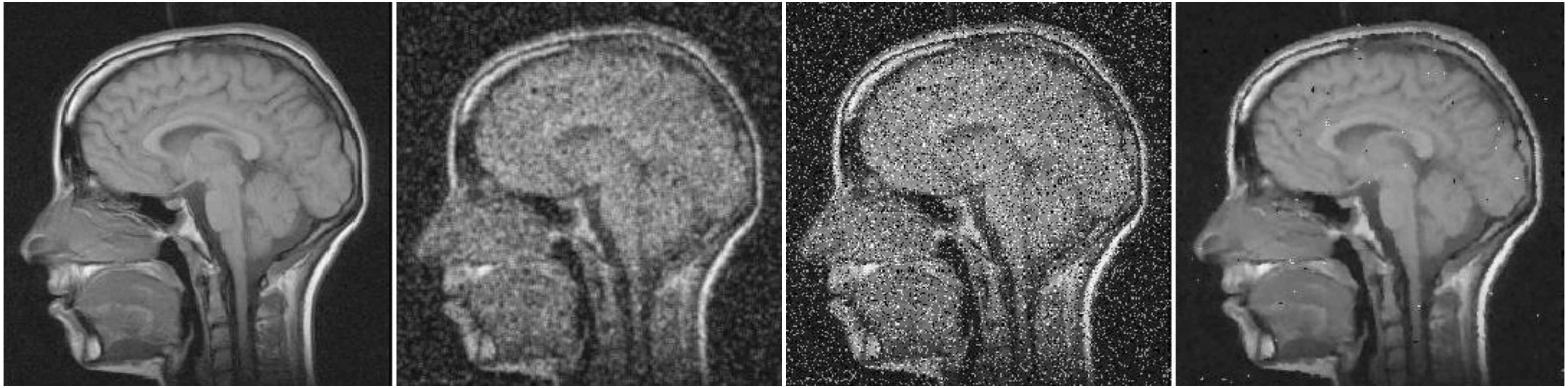
original

dilation - erosion

thresholded result

# Denoising

Standard “salt and pepper” noise example

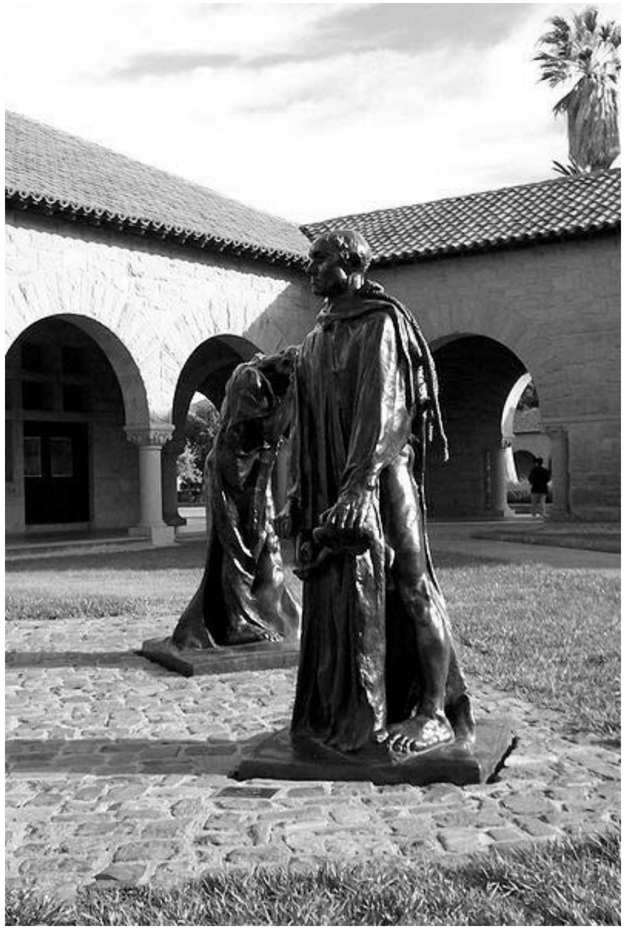


(a) (b) (c) (d)

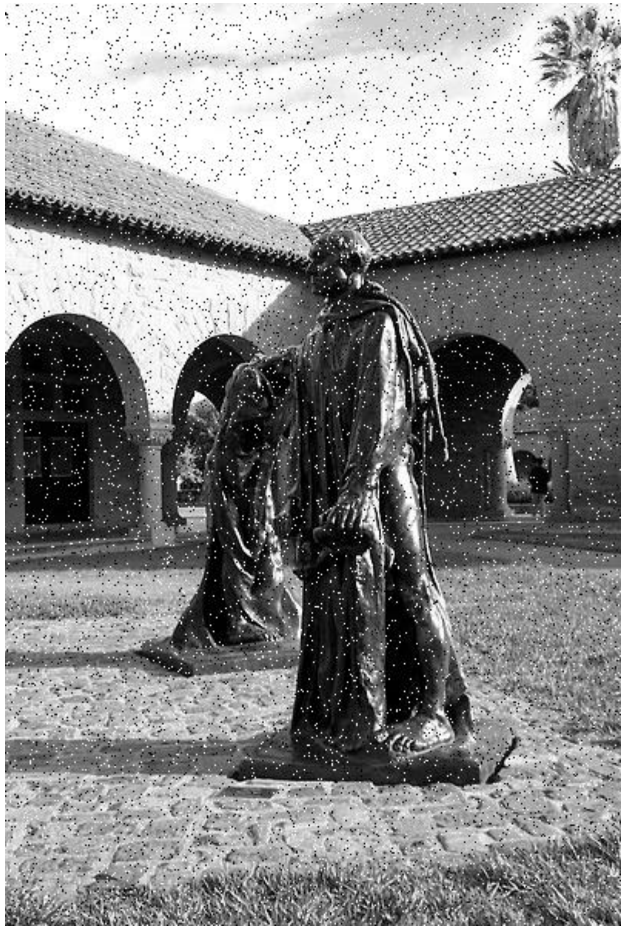
| | | | |
|---|---|---|---|
| Salt and Pepper noise | Original | Median filter | Gaussian filter |

Which is which?

# More realistic denoising



original



salt and pepper noise

3x3 median filtering

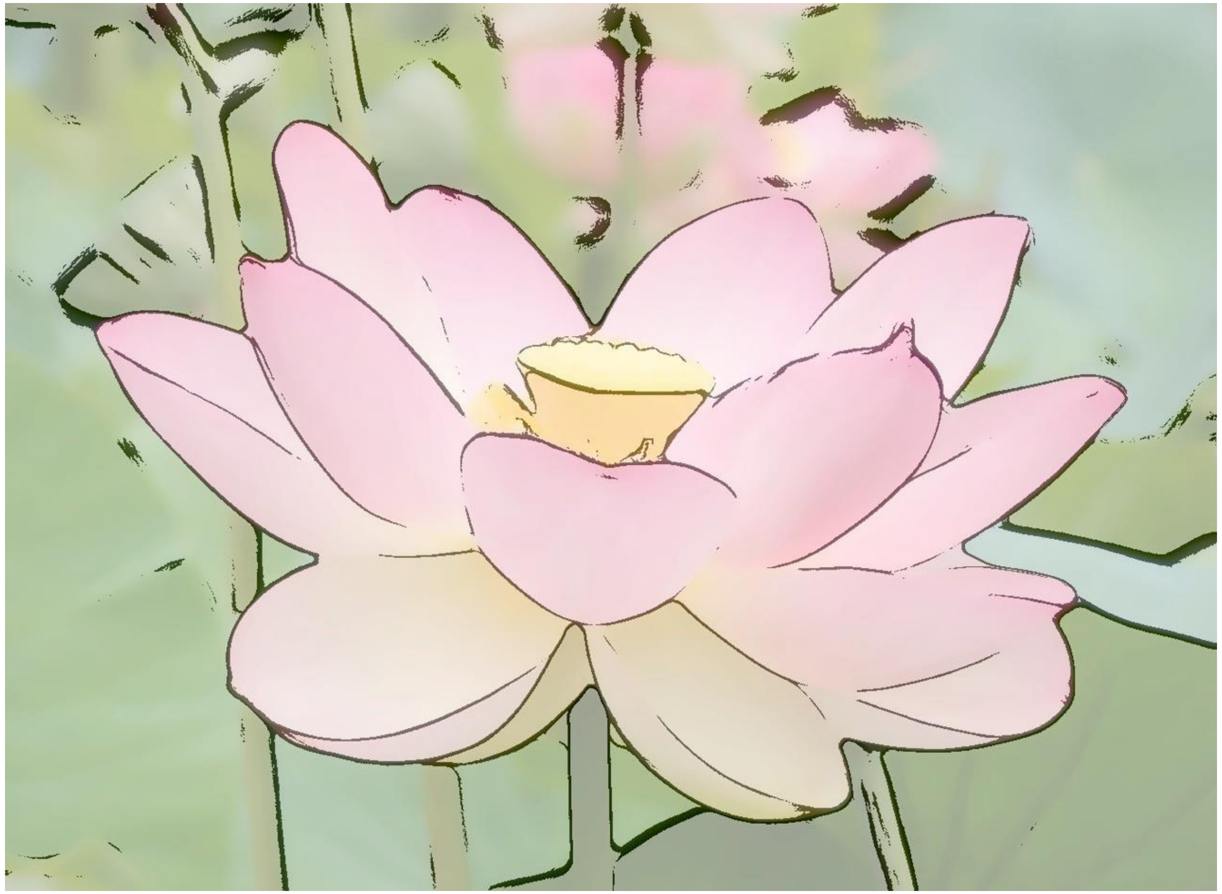
7x7 median filtering

# Removing annoying artifacts

Original

Median filtering

# Cartoonization



How would you create this effect?

# Cartoonization

edges from median blurred image    median blurred image    animated effect

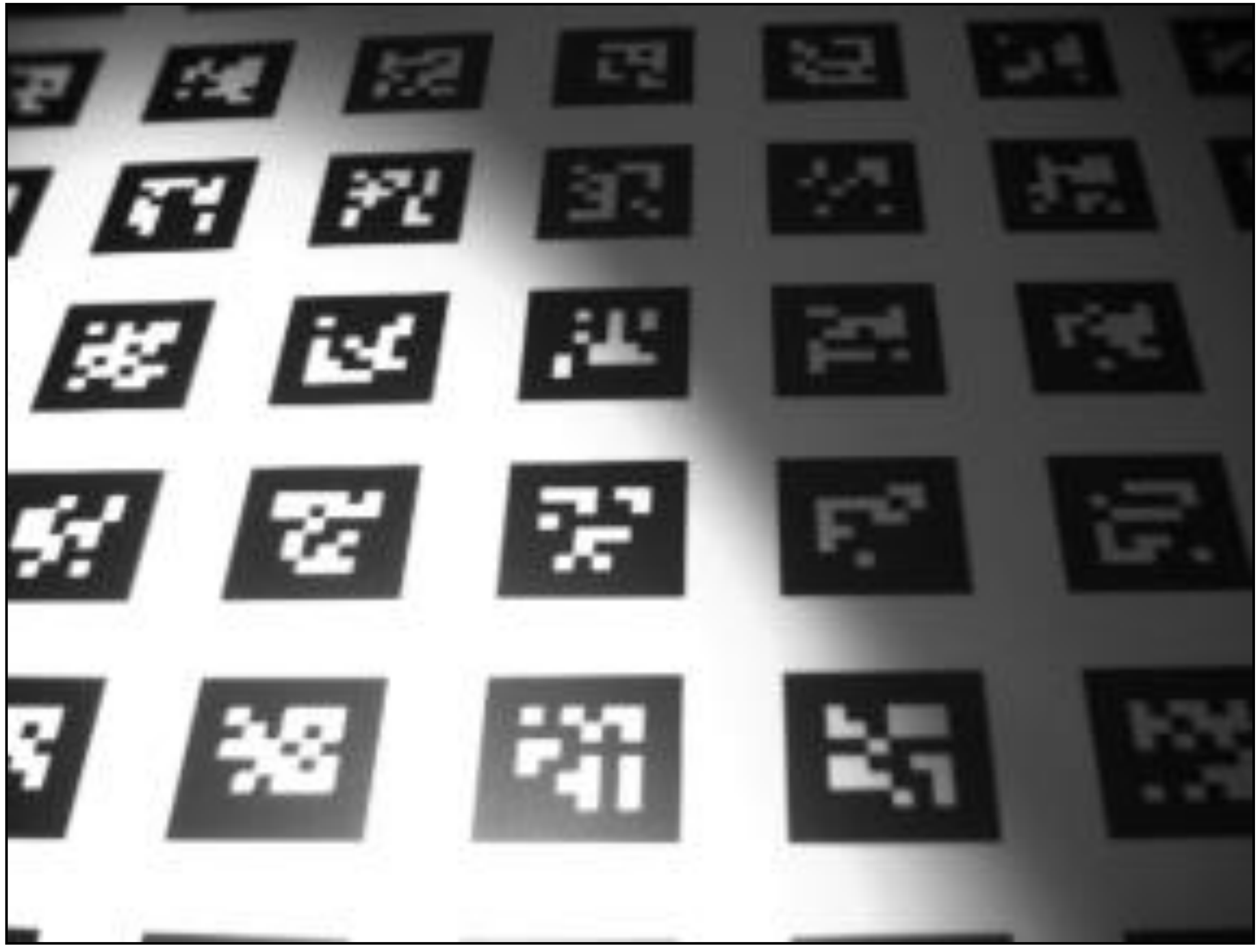
+

=

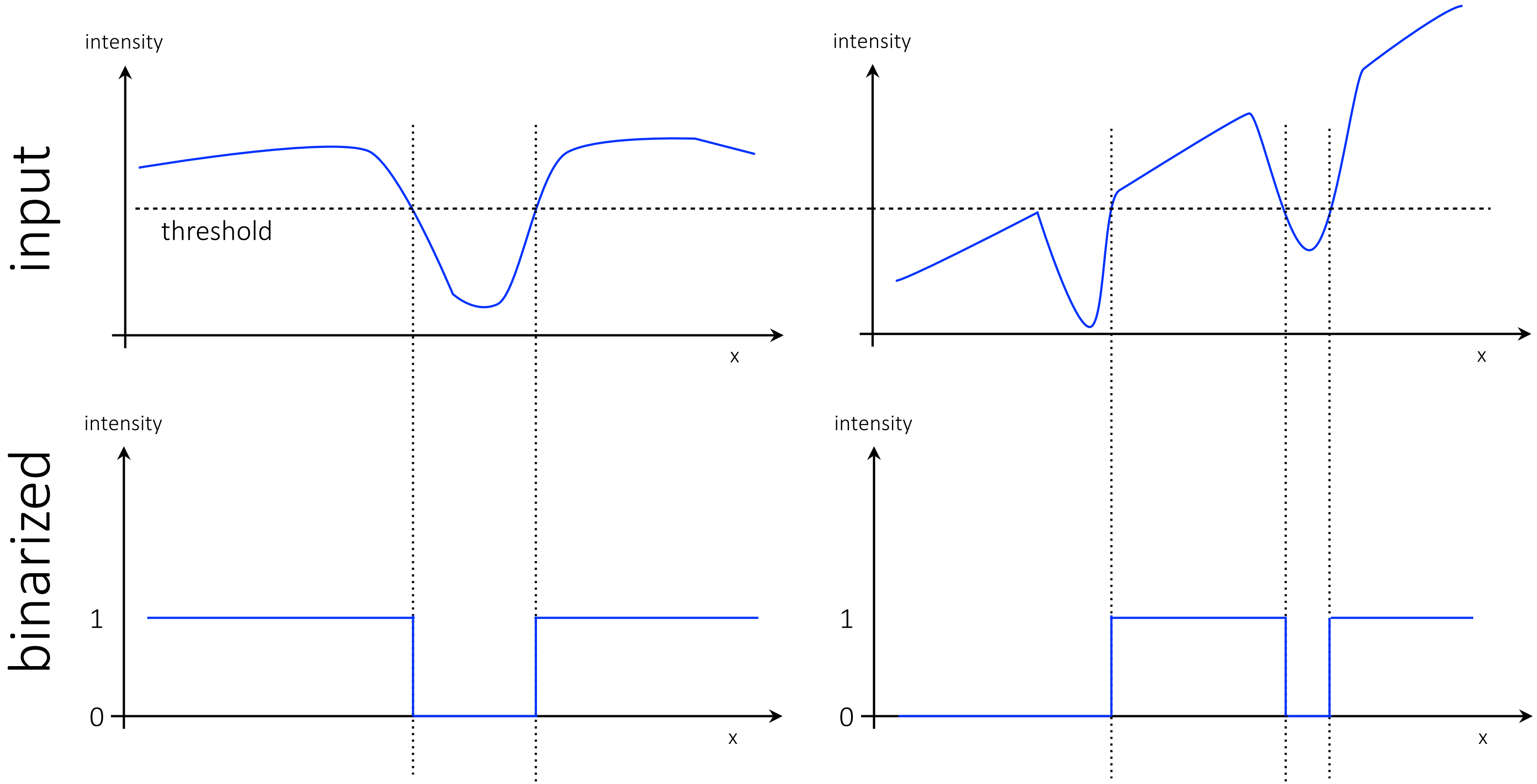
Note: image cartoonization and abstraction are very active research areas.

# Adaptive thresholding

# How would you turn this into a bright binary image?

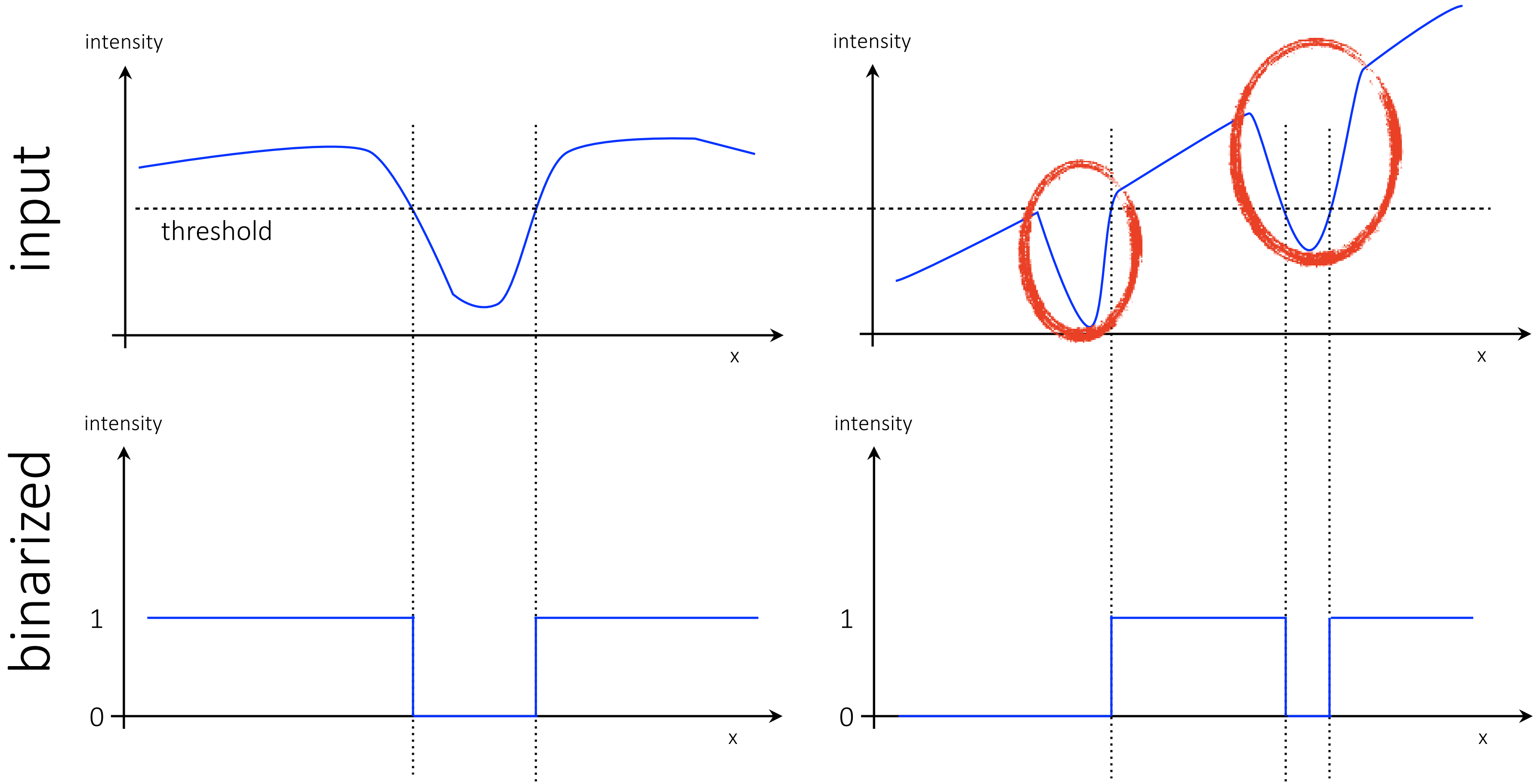


# Single-value thresholding




What is the problem here?

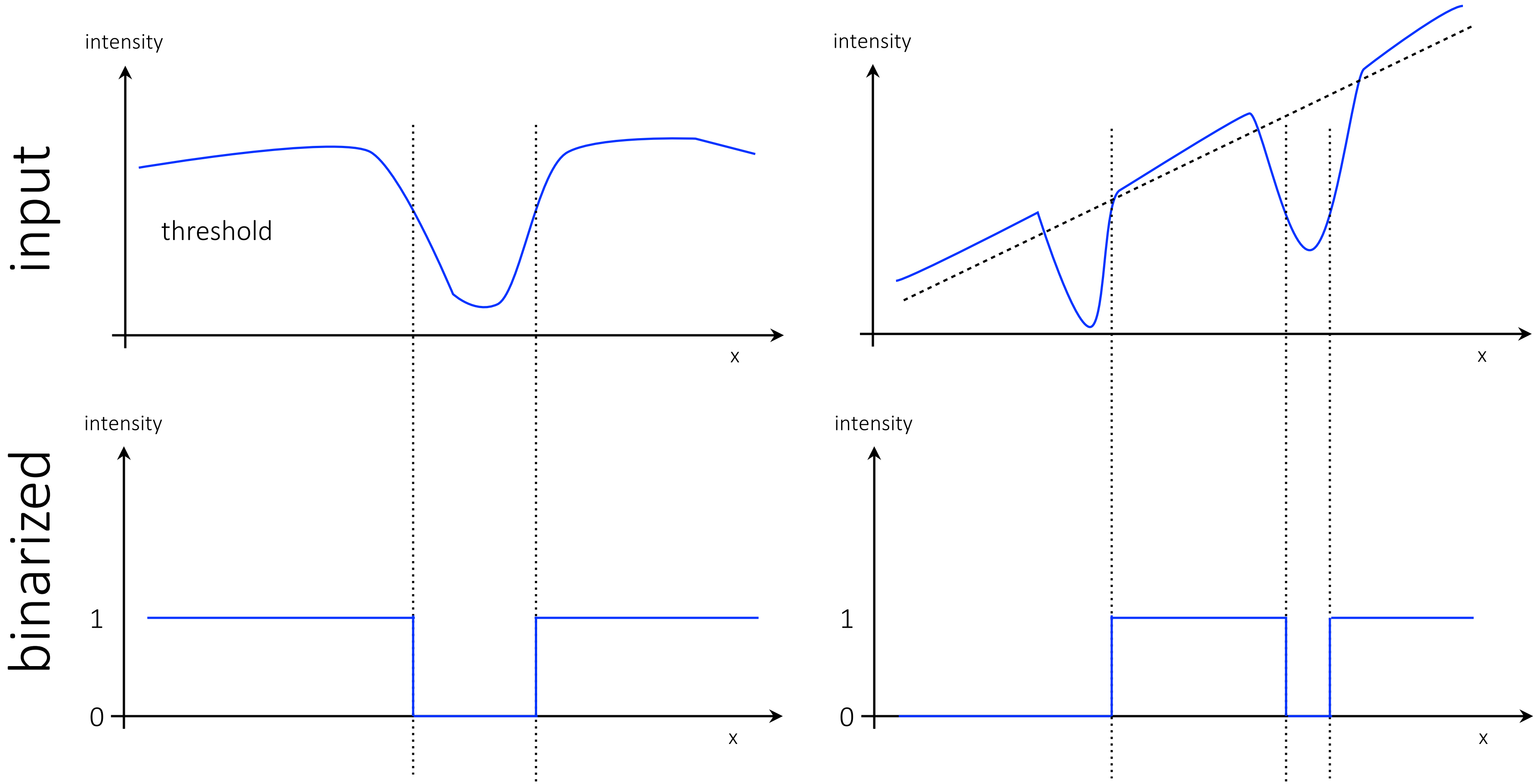
# Single-value thresholding




We can't get both dips with single-value thresholding

How would you do thresholding here?

# Single-value thresholding




Adapt threshold to local values

Can you think of a way to implement this using filtering?

# Adaptive thresholding

Global thresholding:

$$g[x, y] = \begin{cases} 1, & f[x, y] > \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

Adaptive thresholding using mean filtering:

$$g[x, y] = \begin{cases} 1, & f[x, y] > \text{mean}(W(x, y)) \\ 0, & \text{otherwise} \end{cases}$$

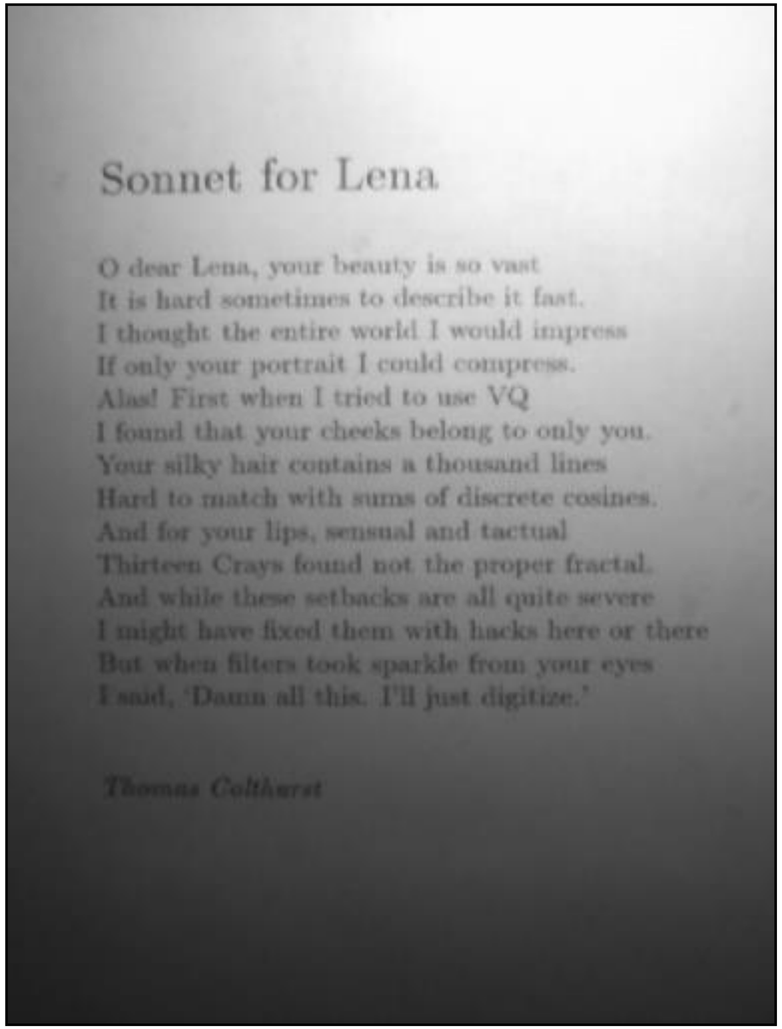
Adaptive thresholding using median filtering:

$$g[x, y] = \begin{cases} 1, & f[x, y] > \text{median}(W(x, y)) \\ 0, & \text{otherwise} \end{cases}$$

Median: greater than 50%
You can use any other percentile

When using rank filters, this is a generalized version of morphological operations.

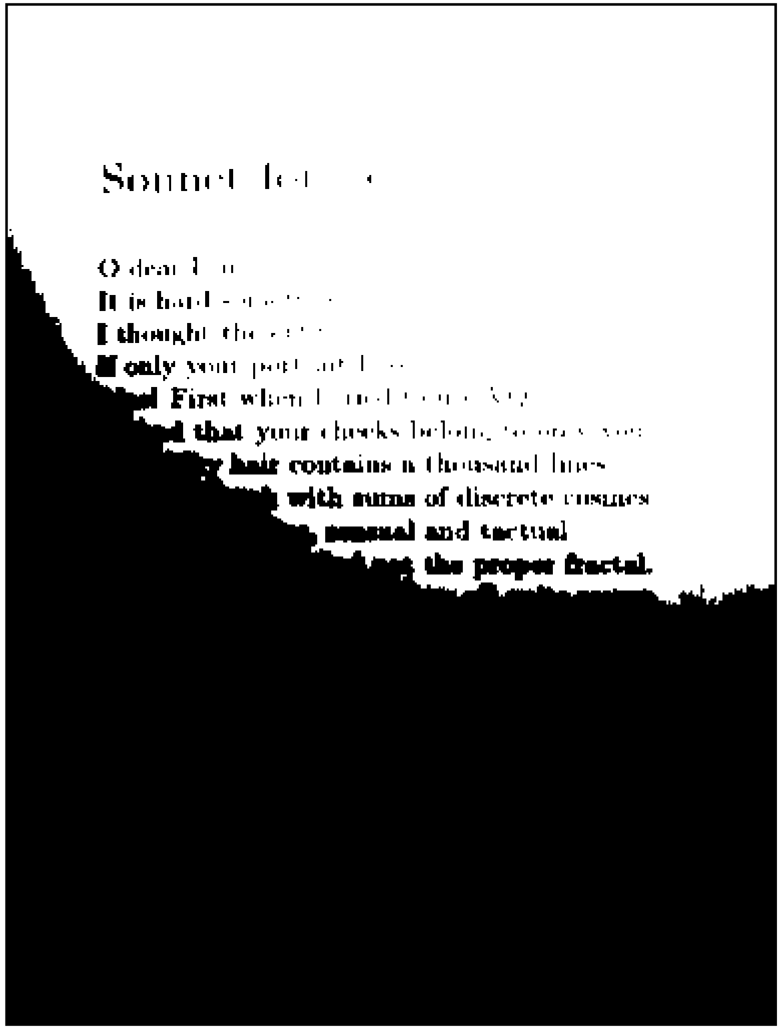
# Examples


Sonnet for Lena

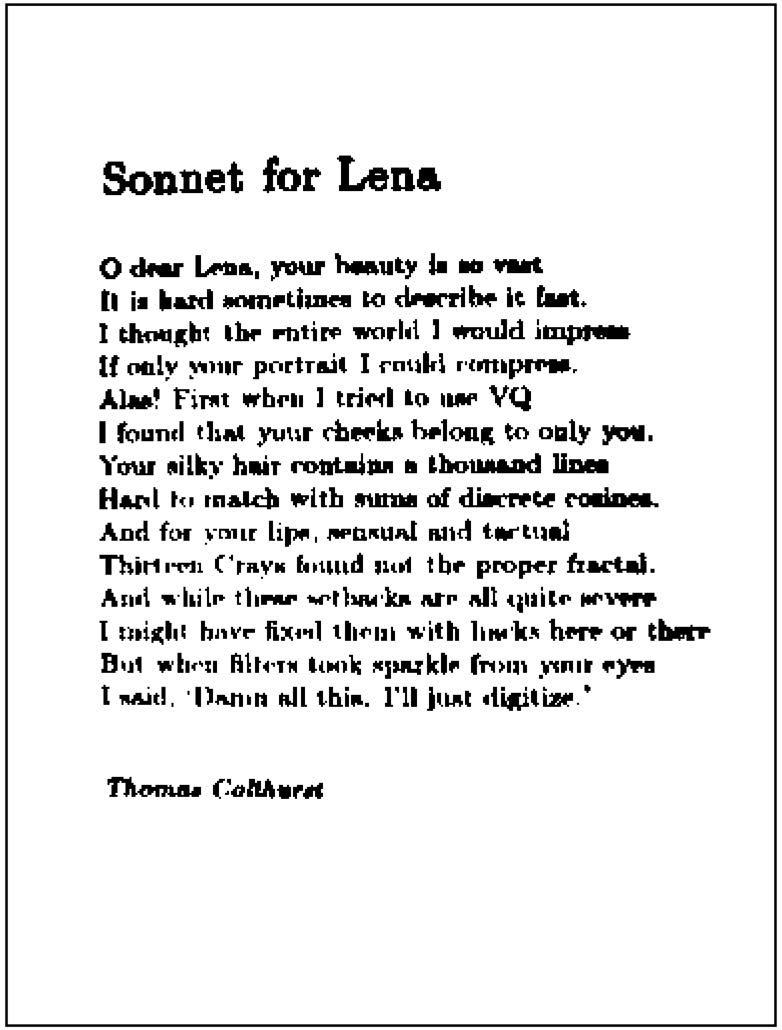
O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

*Thomas Colthurst*

original



global thresholding

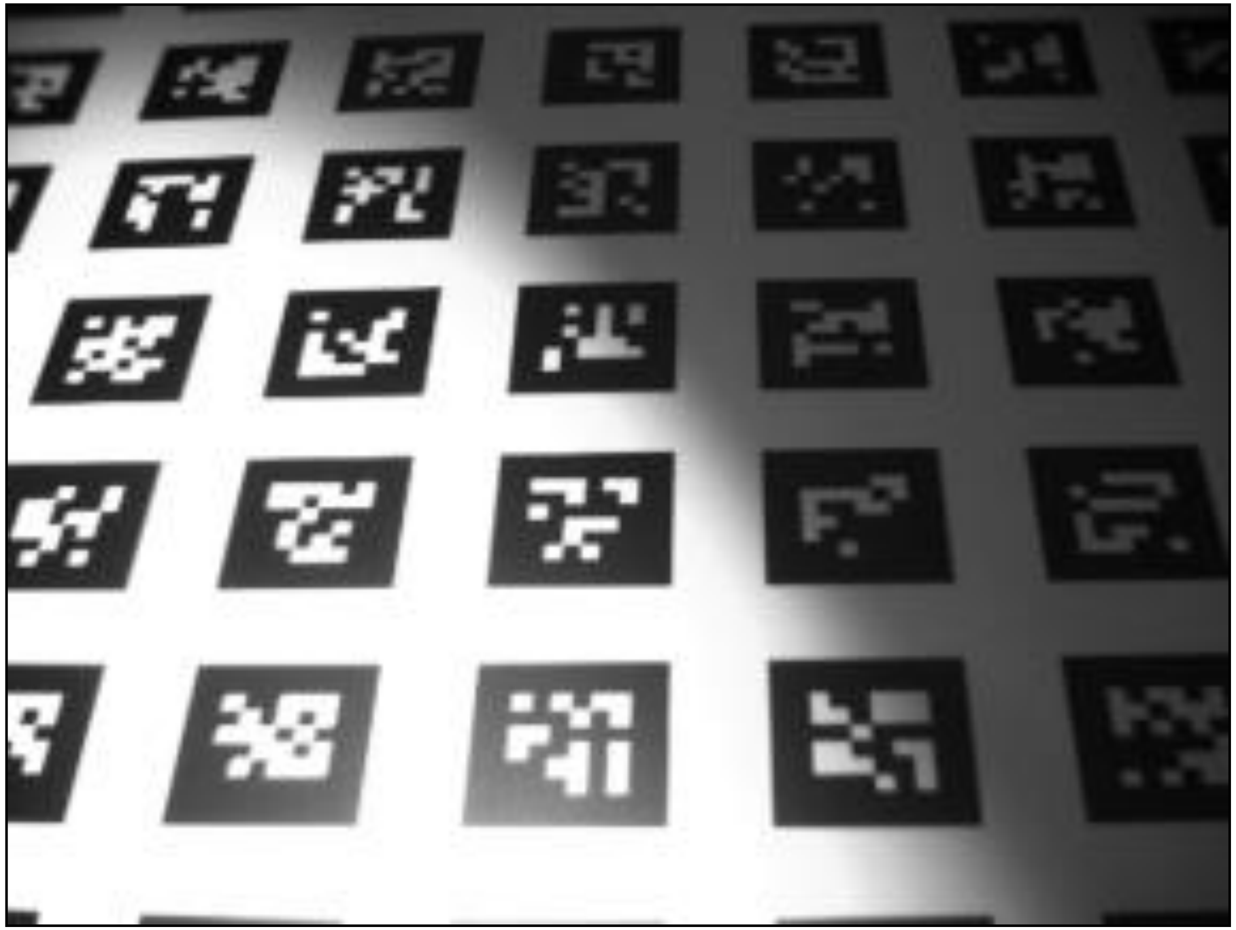

Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

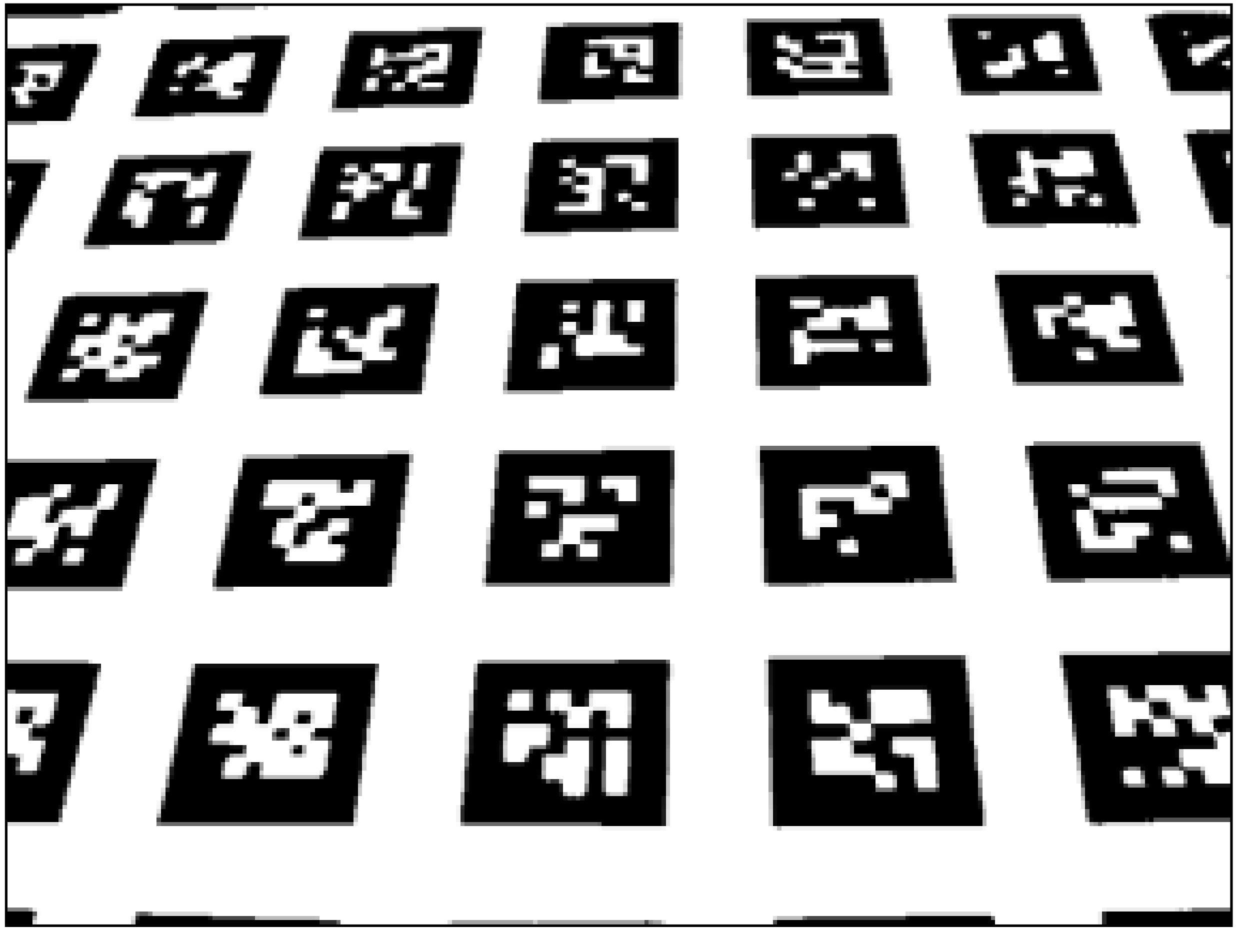
*Thomas Colthurst*

adaptive thresholding

# Examples



original



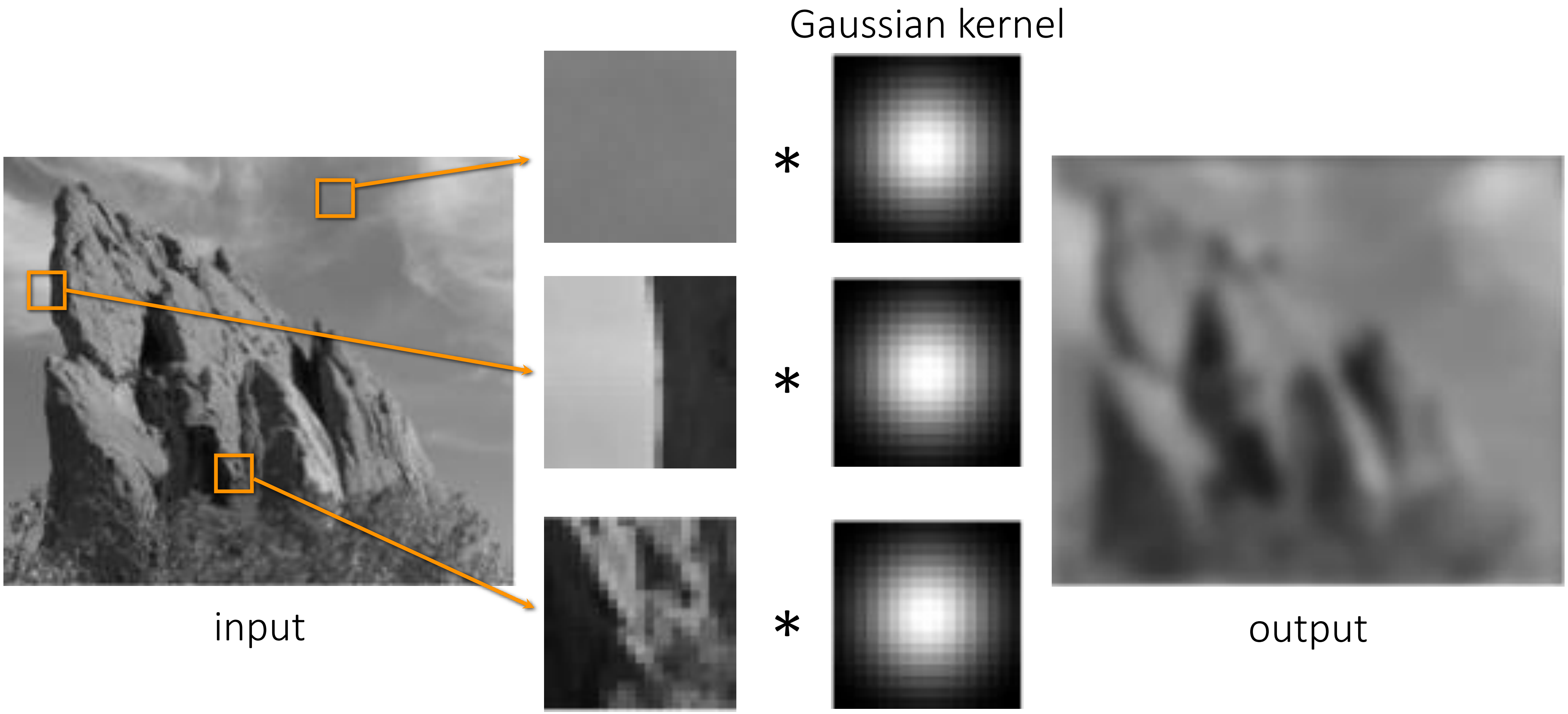
adaptive thresholding

# Bilateral filtering

# Fixing Gaussian blur

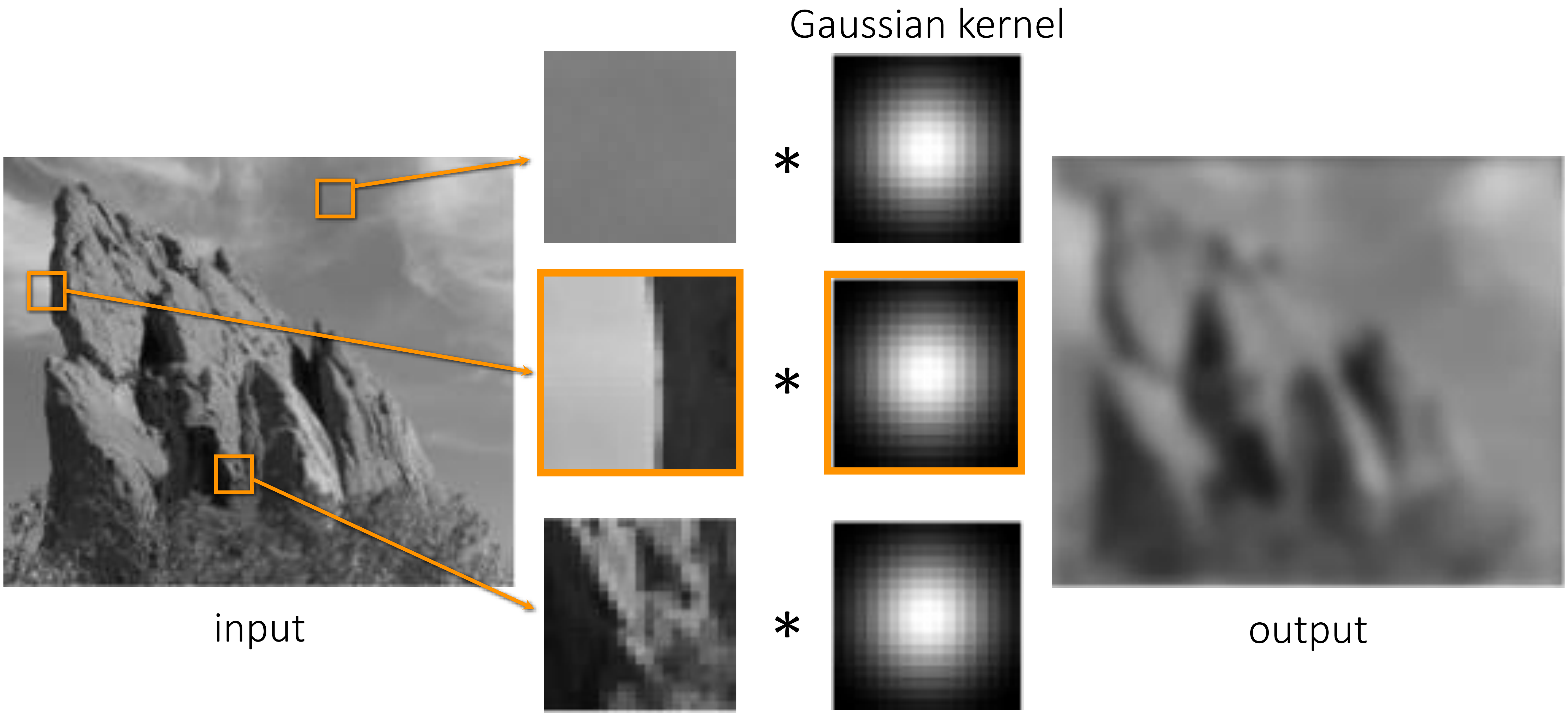
How to smooth out the details in an image without losing the important edges?

# The problem with Gaussian filtering




Why is the output so blurry?

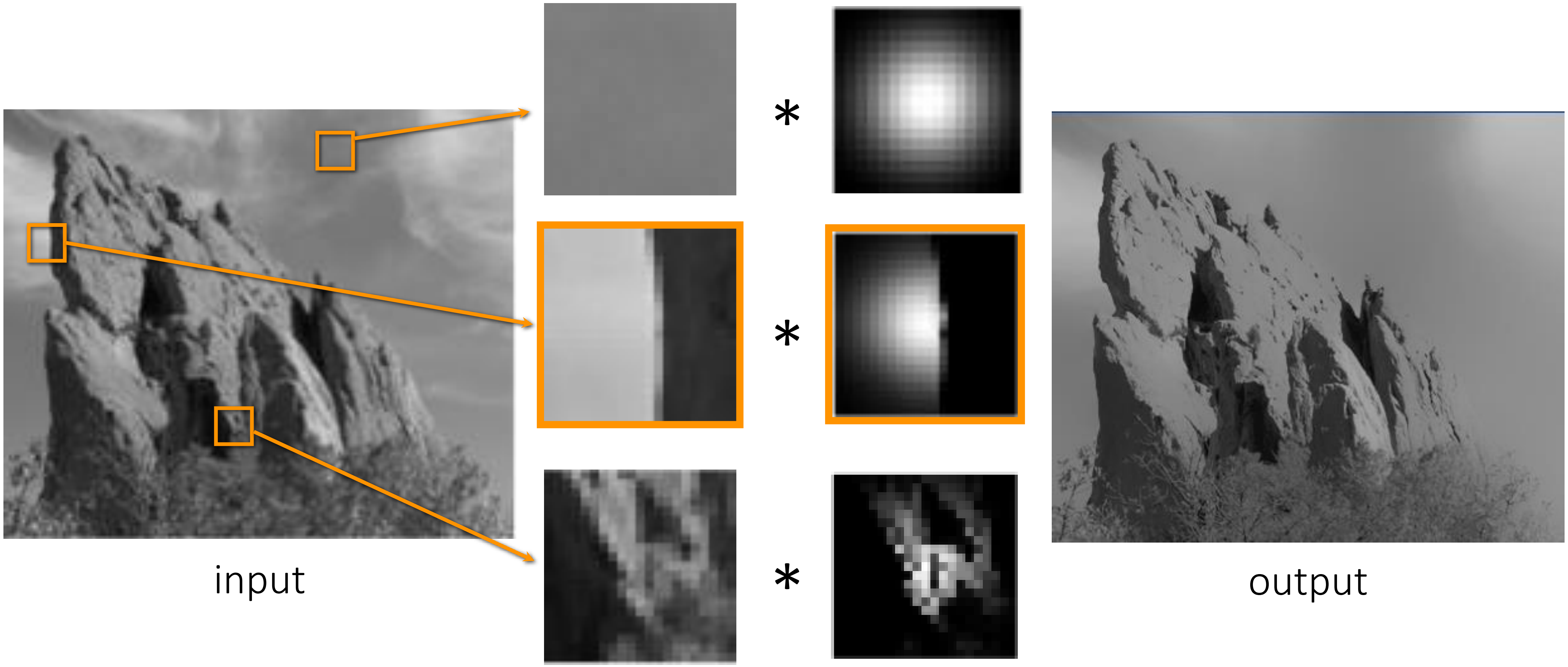
# The problem with Gaussian filtering




Blur kernel averages across edges

# The bilateral filtering solution

bilateral filter kernel



input

output

Do not blur if there is an edge! How does it do that?

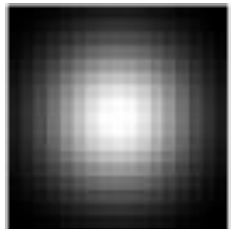
# Bilateral filtering vs Gaussian filtering

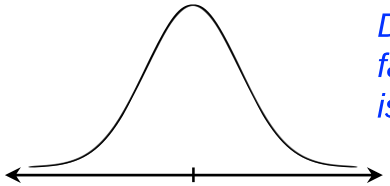
$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

Normalization factor

Spatial weighting

Intensity range weighting





*Does it matter how far the pixel position is?*

$$x = f[m,n] - f[m+k, n+l]$$

$\sigma_s$

$\sigma_r$

if it's nearby and it looks like me

# Bilateral filtering vs Gaussian filtering

Which is which?

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

# Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

Bilateral filtering

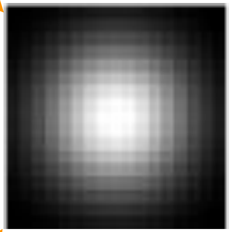
$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

# Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

$\sigma_s$



Spatial weighting: favor *nearby* pixels

Bilateral filtering

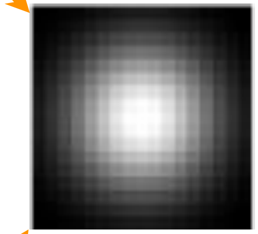
$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

# Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} \boxed{g[k,l]} f[m+k, n+l]$$

$\sigma_s$

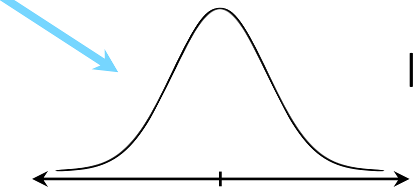


Spatial weighting: favor *nearby* pixels

Bilateral filtering

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} \boxed{g[k,l]}\,\boxed{r_{mn}[k,l]} f[m+k, n+l]$$

$\sigma_r$



$x = f[m,n] - f[m+k, n+l]$

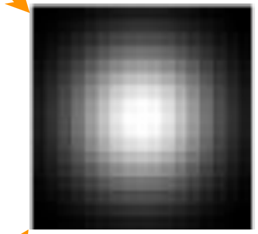
Intensity range weighting: favor *similar* pixels

# Bilateral filtering vs Gaussian filtering

Gaussian filtering

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

$\sigma_s$



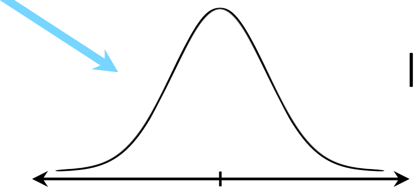
Spatial weighting: favor *nearby* pixels

Bilateral filtering

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

Normalization factor

$\sigma_r$



$x = f[m,n] - f[m+k,n+l]$

Intensity range weighting: favor *similar* pixels

# Bilateral filtering vs Gaussian filtering

Gaussian filtering

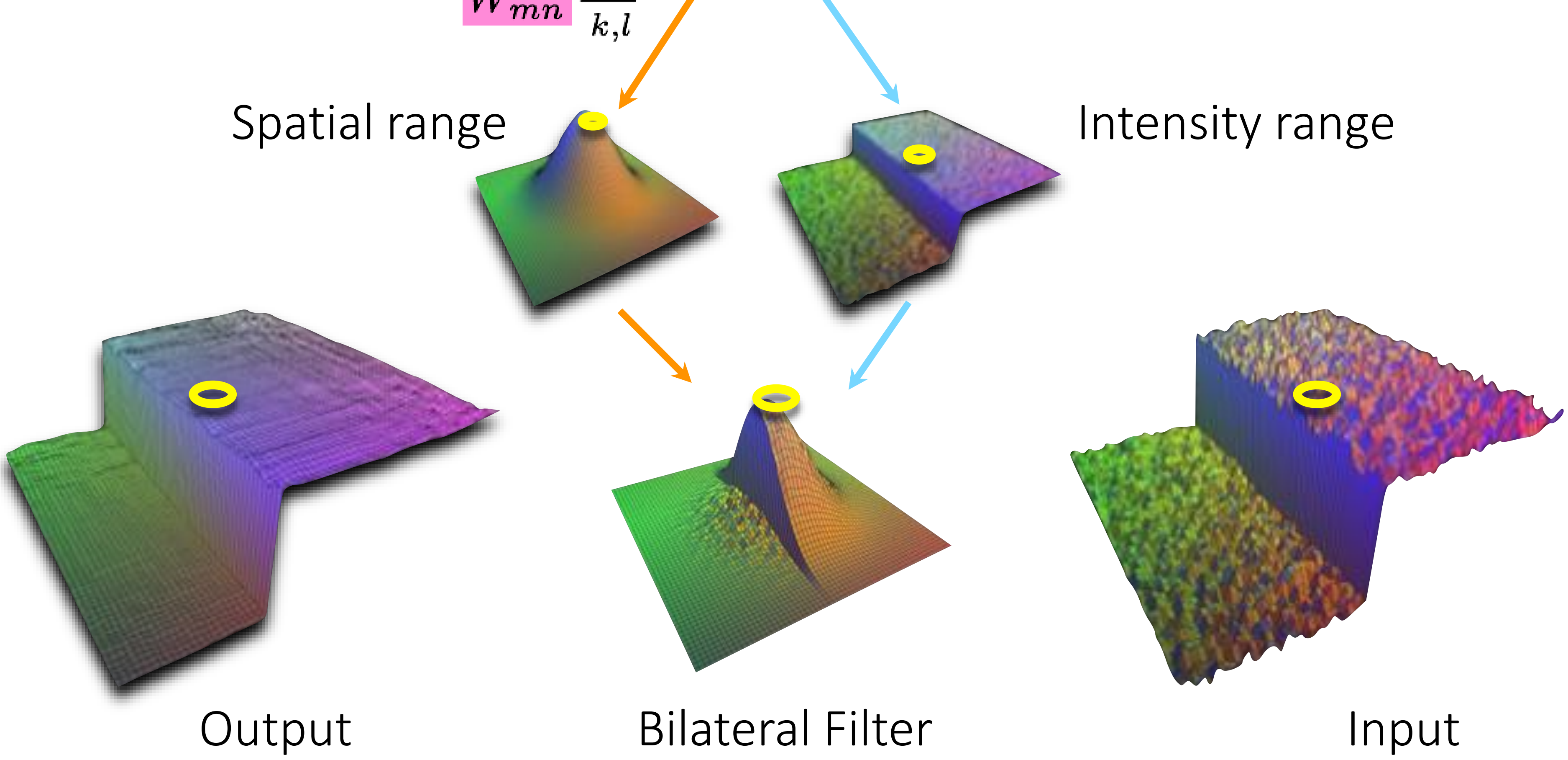
Smooths everything nearby (even edges)
Only depends on *spatial* distance

Bilateral filtering

Smooths ‘close’ pixels in space and intensity
Depends on *spatial* and *intensity* distance

# Bilateral filtering visualization

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$



# Exploring the bilateral filter parameter space

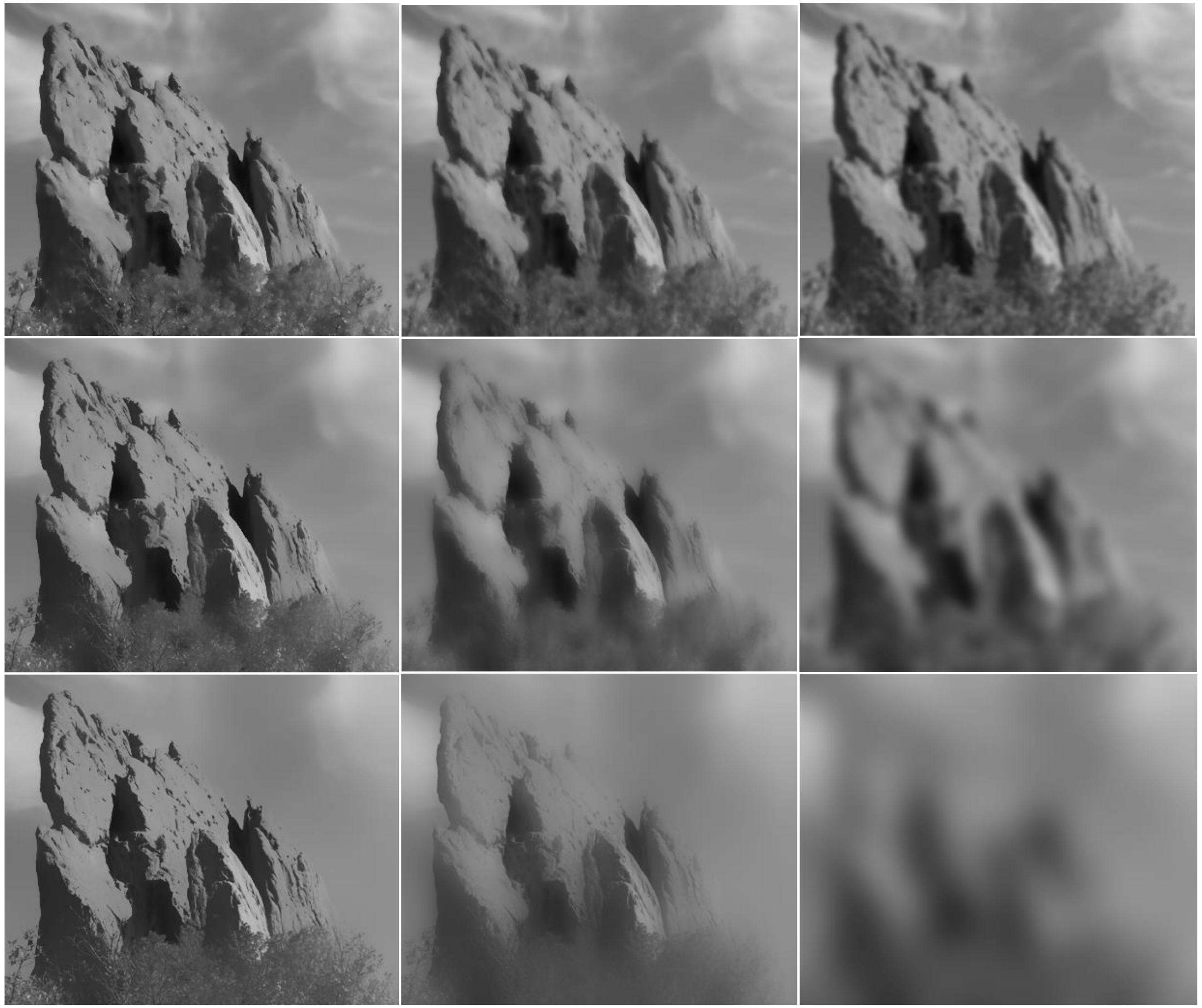
input

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (Gaussian blur)

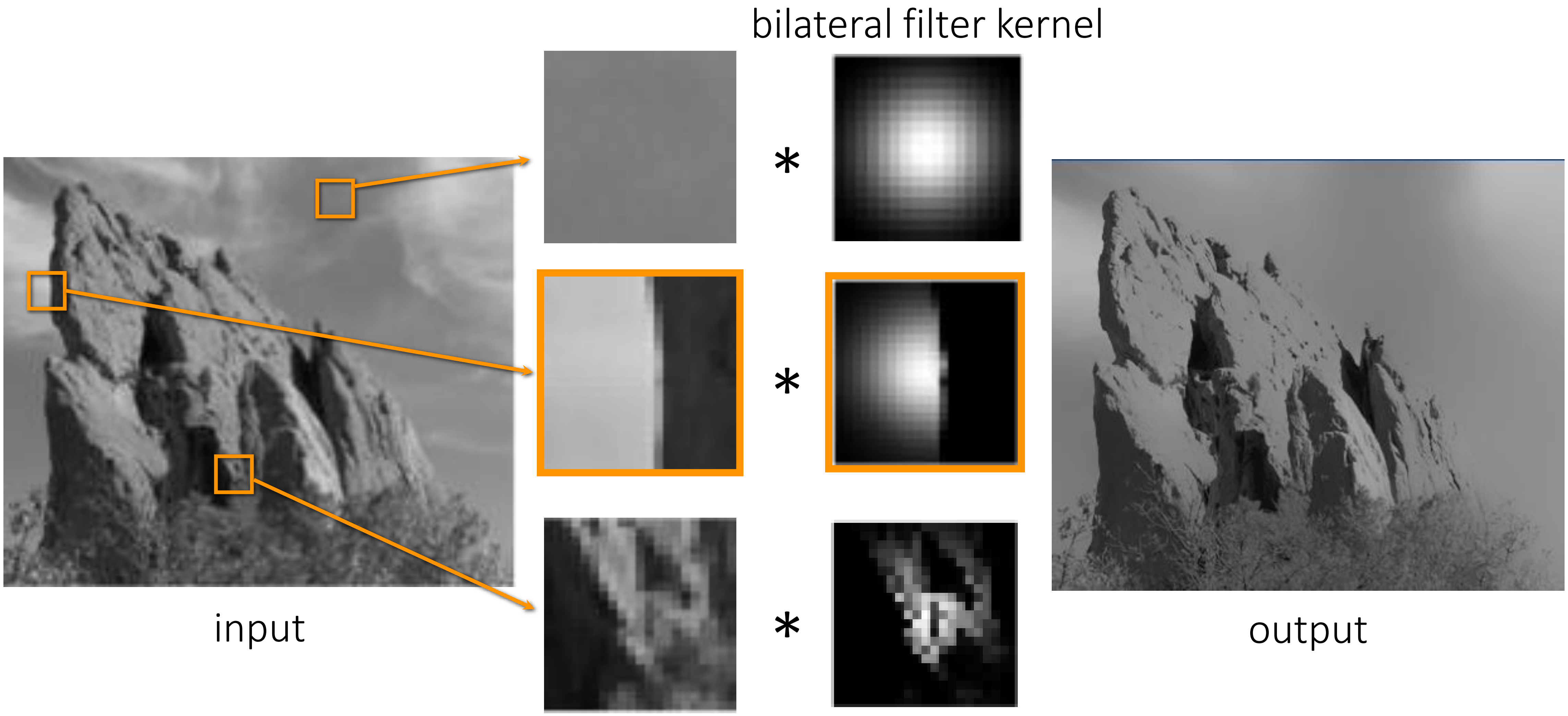
$\sigma_s = 2$

$\sigma_s = 6$

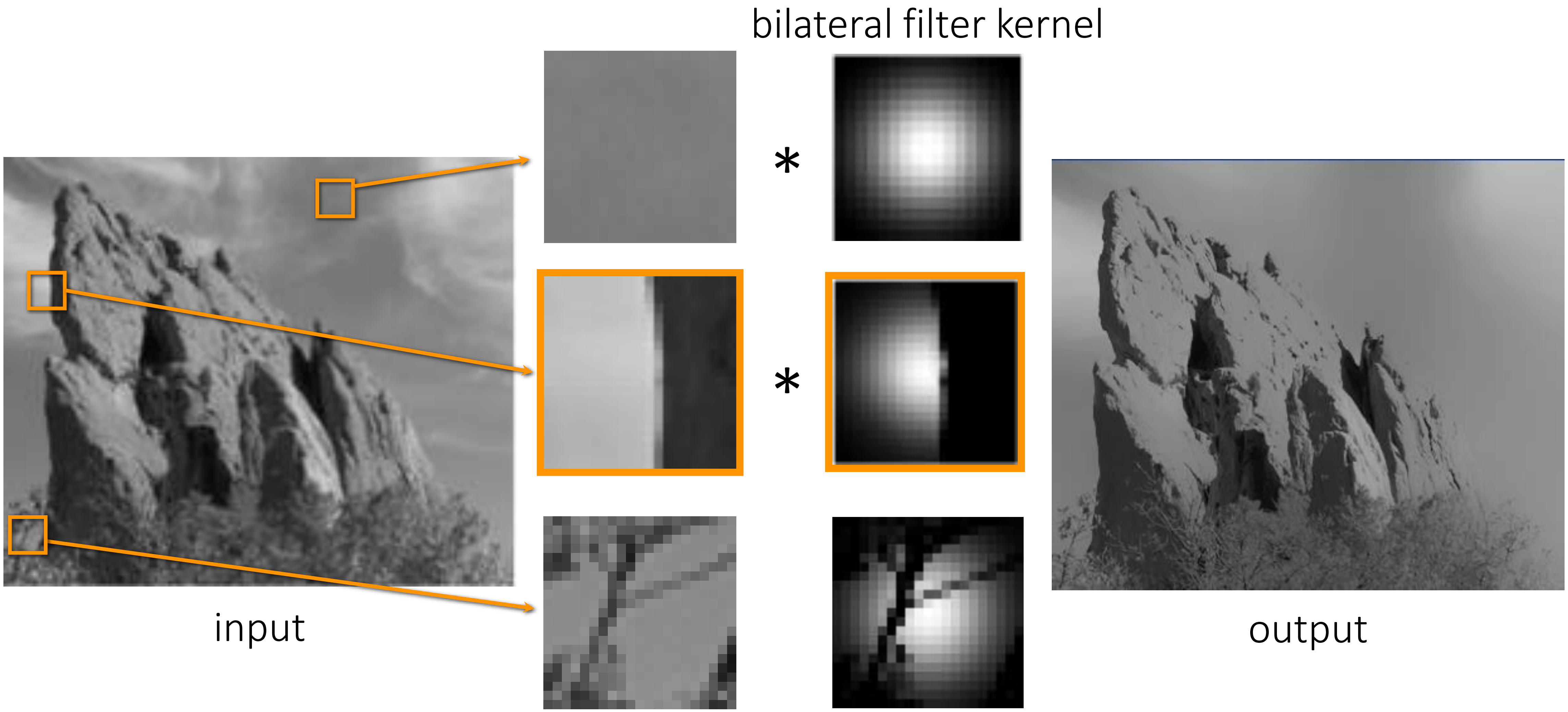
$\sigma_s = 18$



# Does the bilateral filter respect all edges?

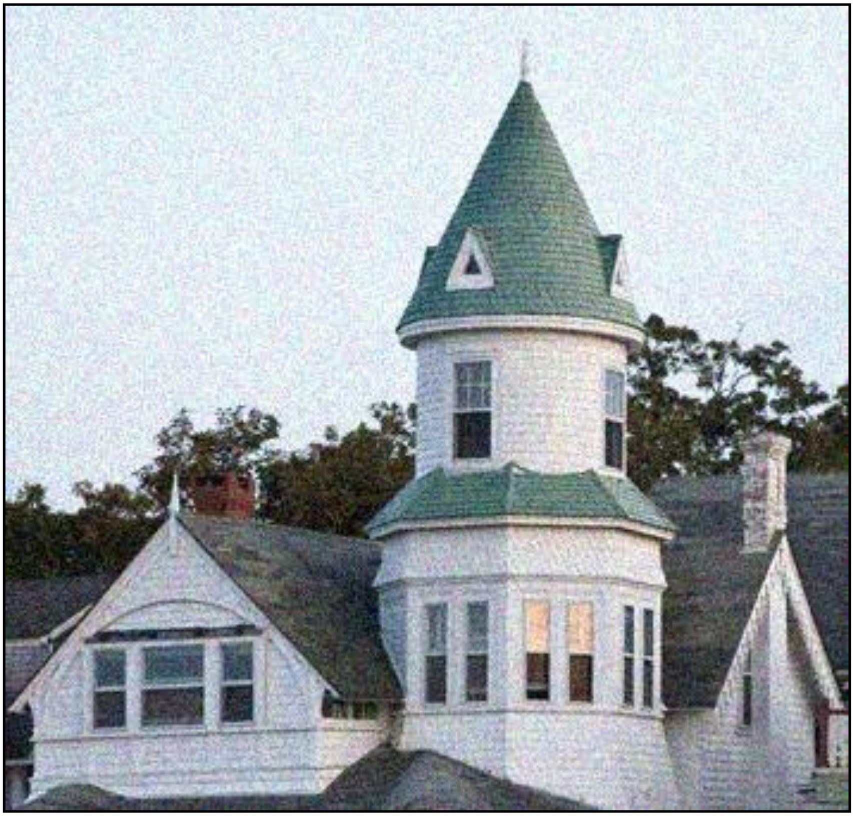


# Does the bilateral filter respect all edges?




Bilateral filter crosses (and blurs) thin edges.

# Denoising

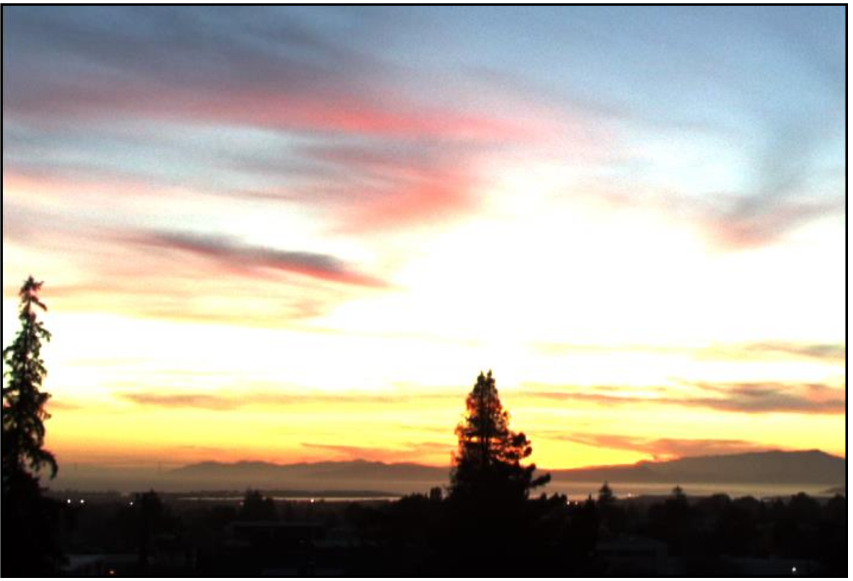


noisy input

bilateral filtering

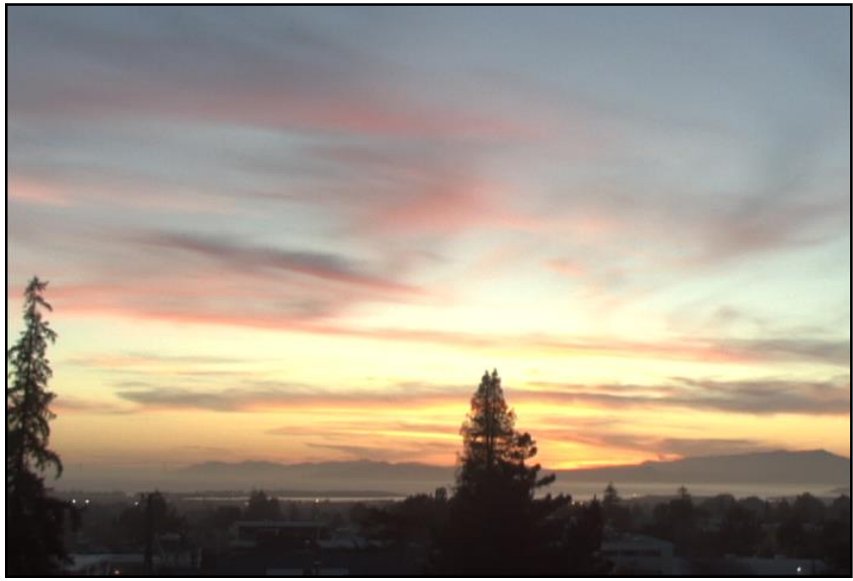
median filtering

# Tone mapping



original

bilateral filtering



simple gamma correction

# Photo retouching

original

digital pore removal (aka bilateral filtering)

# Before

# After

# Close-up comparison

original

digital pore removal (aka bilateral filtering)

# Is the bilateral filter:

Linear?

Shift-invariant?

# Is the bilateral filter:

Linear?

- No.

Shift-invariant?

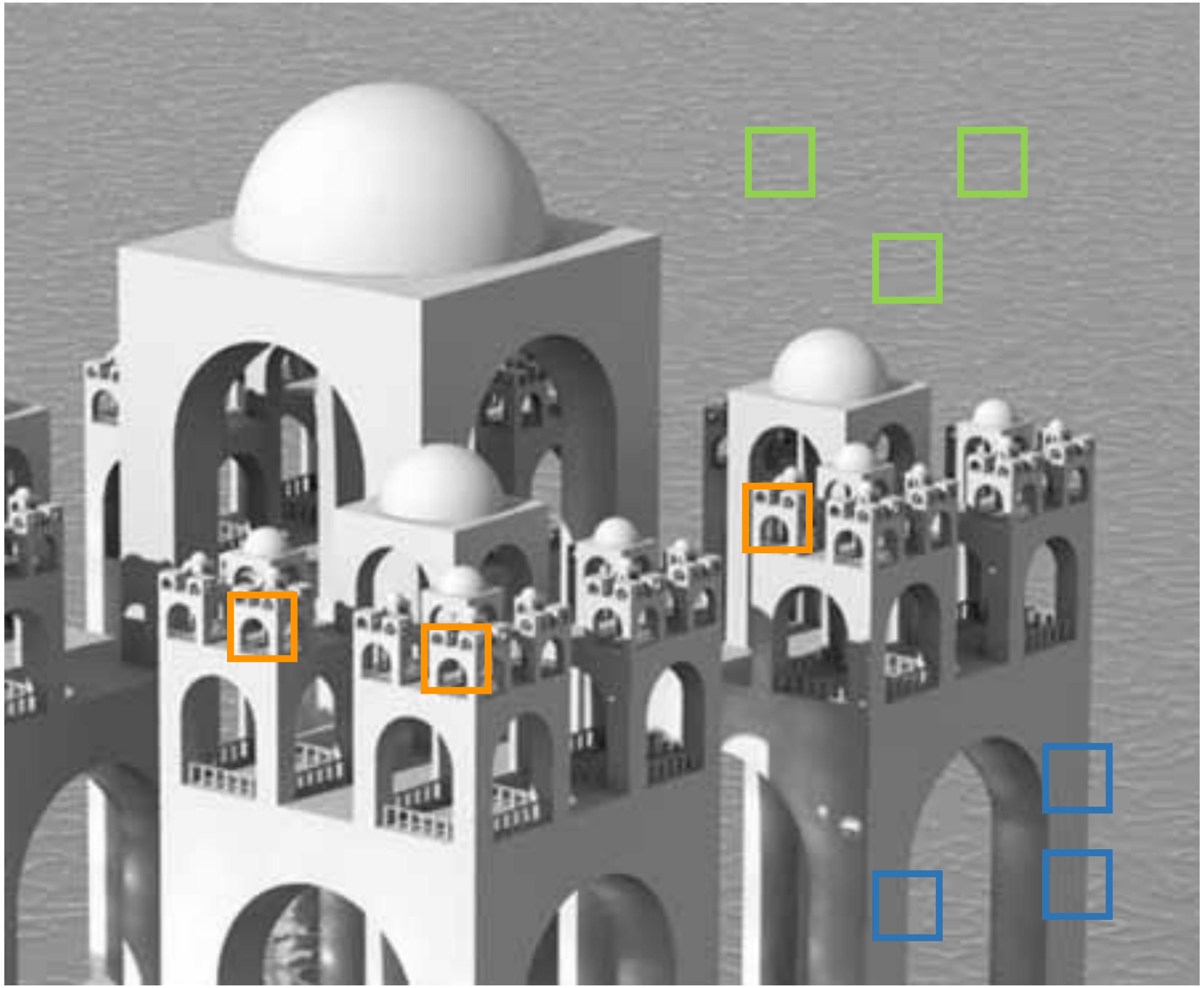
- No.

Bilateral filtering cannot be implemented as convolution. This makes naïve implementation very computationally expensive.

Efficient algorithms for bilateral filtering are an active research area.

# Non-local means

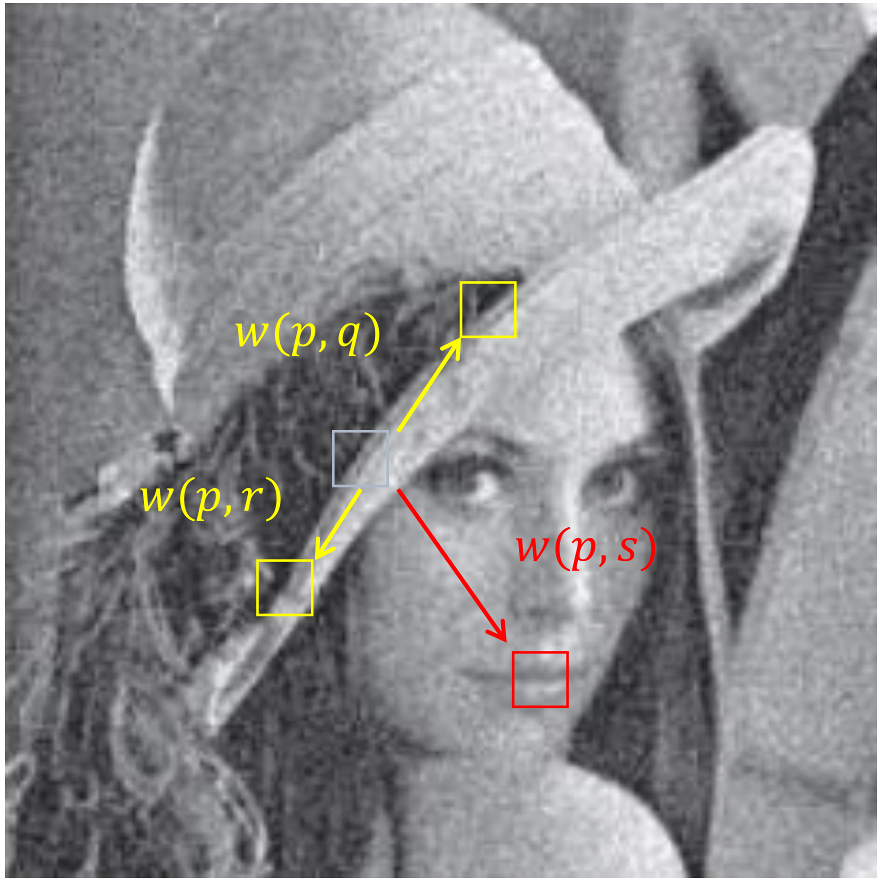
# Redundancy in natural images



# Non-local means

No need to stop at neighborhood. Instead search *everywhere* in the image.

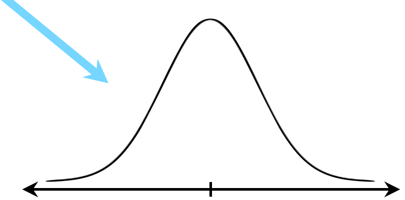
$$\hat{x}(i) = \frac{1}{C_i} \sum_j y(j) \underbrace{e^{-\frac{SSD\left(\mathrm{y}(N_i) - \mathrm{y}(N_j)\right)}{2\sigma^2}}}_{w(i,j)}$$



# Non-local means vs bilateral filtering

Non-local means filtering

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} r_{mn}[k,l] f[m+k, n+l]$$

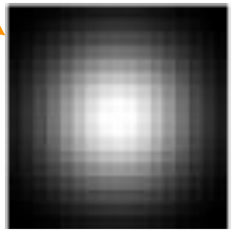


$$x = f[m,n] - f[m+k, n+l]$$

Intensity range weighting: favor *similar* pixels (patches in case of non-local means)

Bilateral filtering

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$



Spatial weighting: favor *nearby* pixels

# Everything put together

Gaussian filtering

Smooths everything nearby (even edges)
Only depends on *spatial* distance

Bilateral filtering

Smooths 'close' pixels in space and intensity
Depends on *spatial* and *intensity* distance

Non-local means

Smooths similar patches no matter how far away
Only depends on *intensity* distance

# Denoising example

noisy input

Gaussian filtering

bilateral filtering

non-local means

# Very general forms of “structural” filtering

We will see more in later lectures.

# Is non-local means:

Linear?

Shift-invariant?

# Is non-local means:

Linear?

- No.

Shift-invariant?

- No.

Non-local means is not a convolution, and is generally very very challenging to implement efficiently.

Efficient algorithms for non-local means are an active research area.

# References

Basic reading:

- Szeliski textbook, Sections 3.2 and 8.1

Additional reading:

- Serra, “Image Analysis and Mathematical Morphology,” Academic Press 1983.
  standard reference book on mathematical morphology, also available in course form
  http://cmm.ensmp.fr/~serra/cours/index.htm
- Paris et al., “A Gentle Introduction to the Bilateral Filter and Its Applications,” SIGGRAPH 2007-08, CVPR 2008
  short course on the bilateral filter, including discussion of fast implementations
  https://people.csail.mit.edu/sparis/bf_course/
- Xu et al., “Image Smoothing via $L_0$ Gradient Minimization,” SIGGRAPH 2011
  one of many works on image abstraction and cartoonization, with a good related work section
- Buades et al., “Nonlocal Image and Movie Denoising,” IJCV 2008
  the journal version of the original non-local means paper
- Felzenszwalb and Huttenlocher, “Distance Transforms of Sampled Functions,” ToC 2012
  discusses how to compute distance transforms and skeletons using morhology