AMME4710: COMPUTER VISION AND IMAGE PROCESSING WEEK 3

Dr. Mitch Bryson

School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney

Last Week

- Radiometry, Light and Shading
- Colour and Colour Image Processing

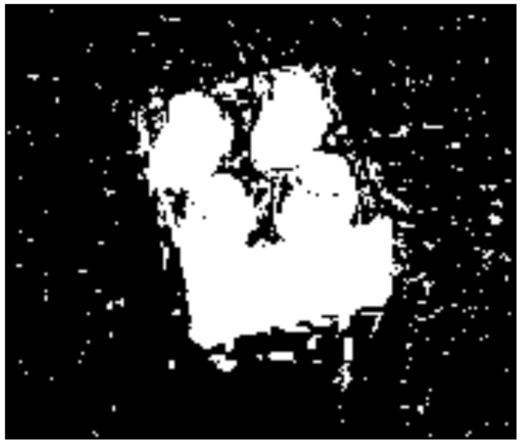
This Week's Lecture

- Basic Image Processing and Applications:
  - Morphological operations
  - Image filtering
  - Edge detection

- Learning Objectives:
  - To explore algorithms for image filtering and edge detection, which form the building blocks of higher-level algorithms for object detection, tracking and recognition

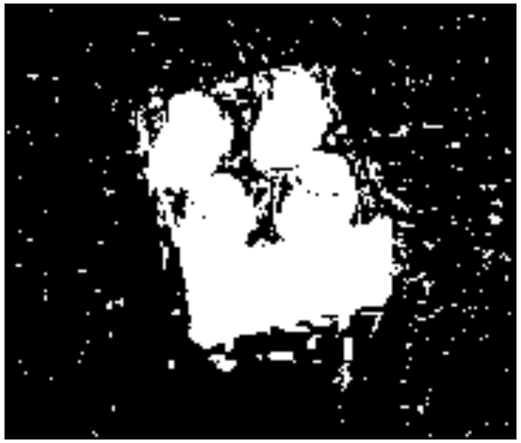
Morphological Operations on Images

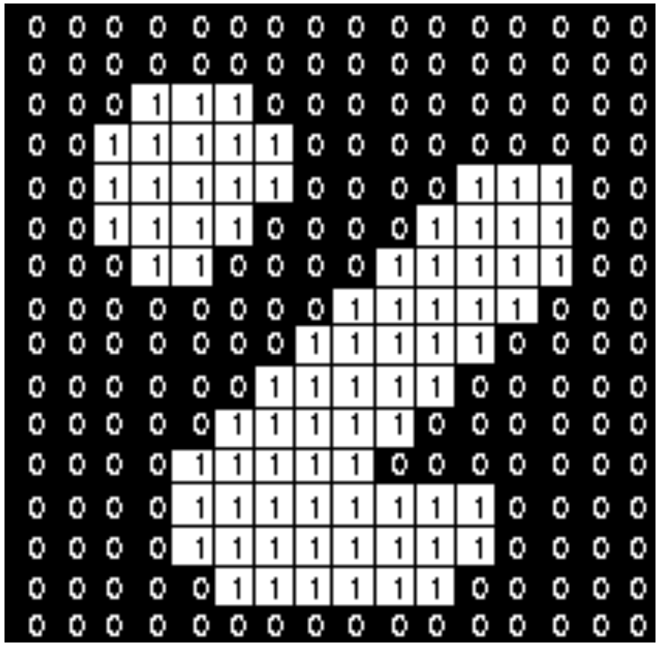
- Binary images (for example arising from thresholding) typically contain many imperfections, such as noise and clutter
- **Morphological operations** are designed to enhance the morphology (shape) of binary image data



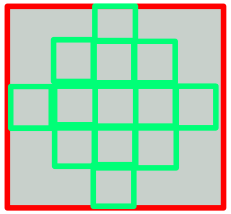
Morphological Operations on Images

- Binary images (for example arising from thresholding) typically contain many imperfections, such as noise and clutter
- **Morphological operations** are designed to enhance the morphology (shape) of binary image data
- Operations use a **structuring element**: a small raster template which is used to probe the local neighborhood of each image pixel to determine the output of a specific function



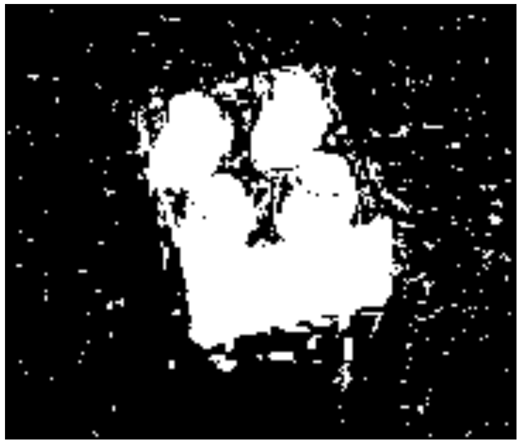


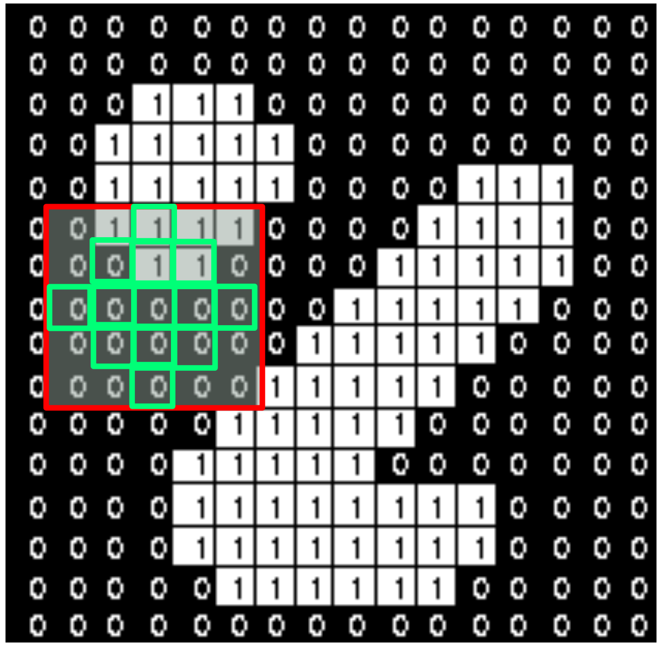
Structuring element



Morphological Operations on Images

- Binary images (for example arising from thresholding) typically contain many imperfections, such as noise and clutter
- **Morphological operations** are designed to enhance the morphology (shape) of binary image data
- Operations use a **structuring element**: a small raster template which is used to probe the local neighborhood of each image pixel to determine the output of a specific function

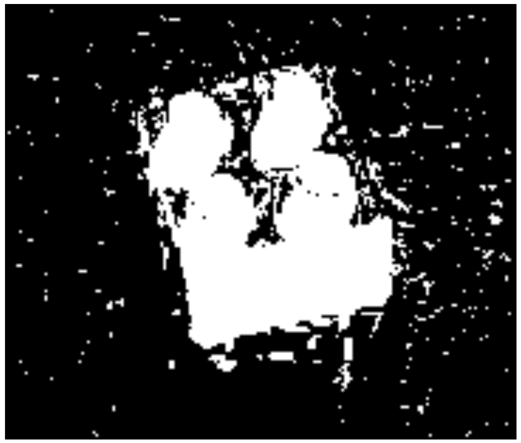


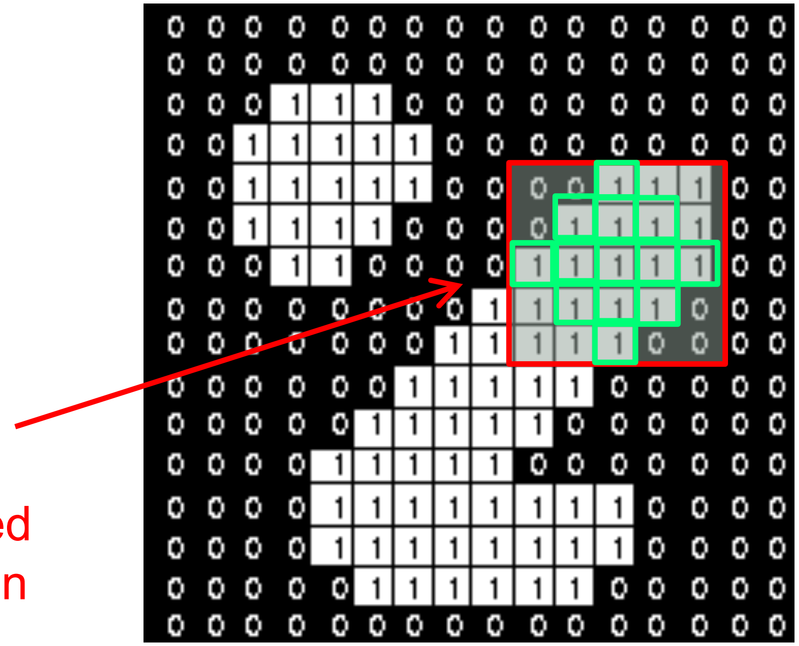


Structuring element applied to pixel location

Morphological Operations on Images

- Binary images (for example arising from thresholding) typically contain many imperfections, such as noise and clutter
- **Morphological operations** are designed to enhance the morphology (shape) of binary image data
- Operations use a **structuring element**: a small raster template which is used to probe the local neighborhood of each image pixel to determine the output of a specific function





Structuring element applied to pixel location

Morphological Operations on Images

- Morphological operations on images are typically denoted using set notation

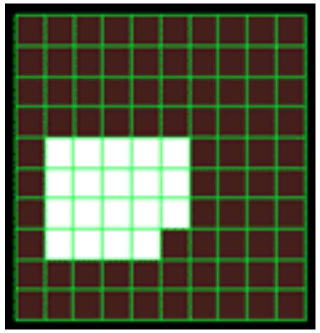


Image A

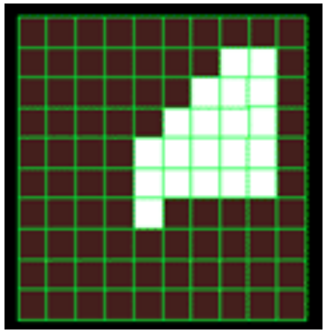
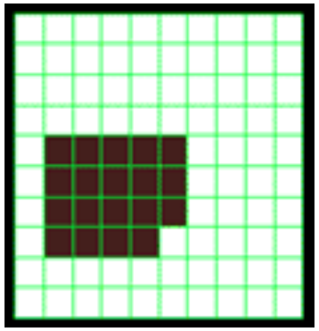
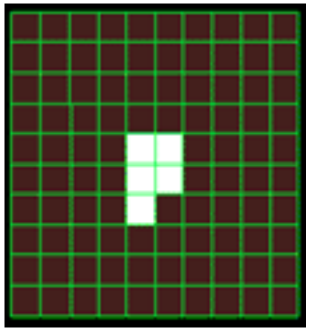
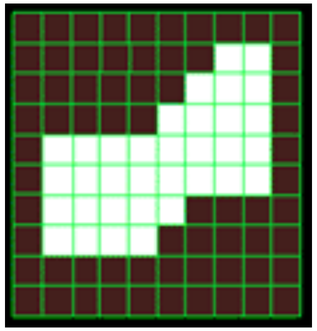


Image B


Compliment of A
$(A)^c$


Intersection
$A \cap B$


Union
$A \cup B$

Morphological Operations on Images

- Morphological operations on images are typically denoted using set notation
- For binary raster data, the output of each pixel in a set operation is akin to logical operations between two pixel values:

p	q	P AND q	P OR q	NOT p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

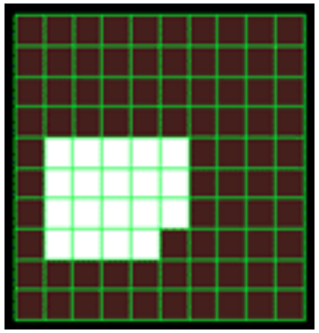


Image A

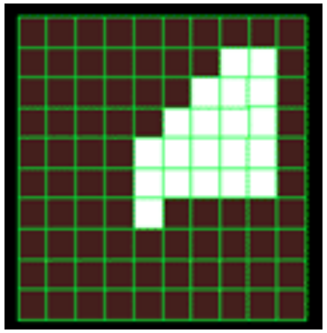
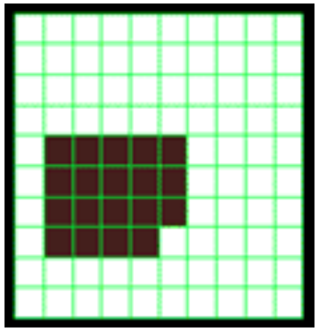
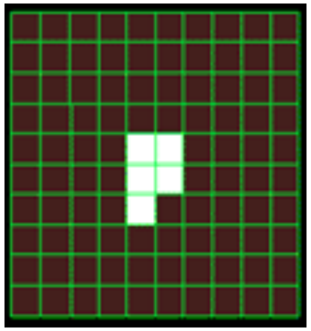
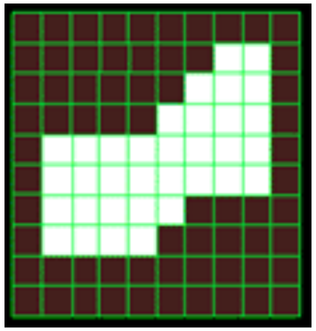


Image B


Compliment of A
$(A)^c$


Intersection
$A \cap B$

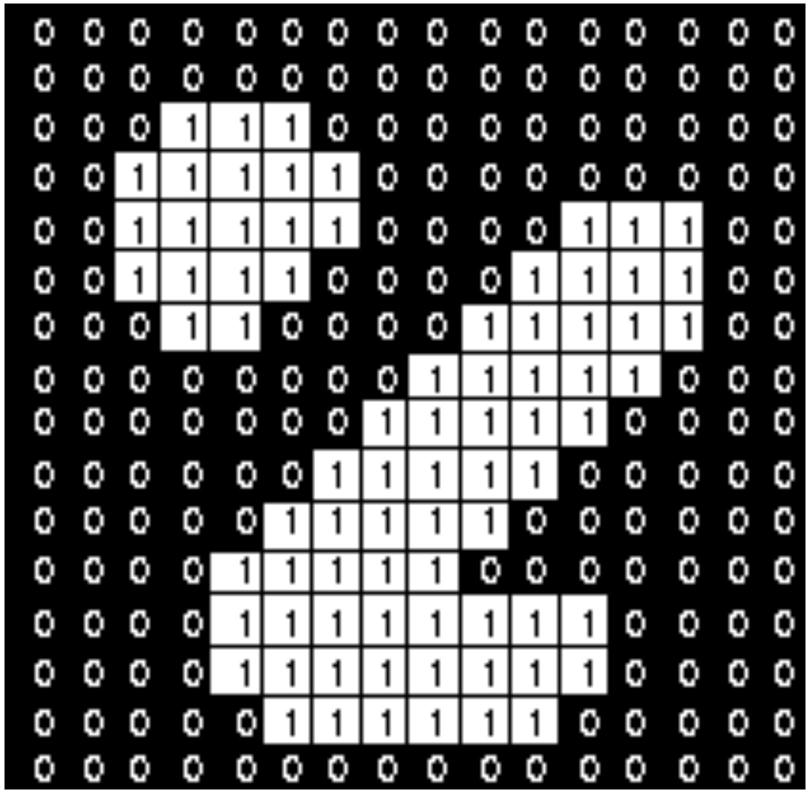

Union
$A \cup B$

Morphological Operations on Images

- A basic morphological operation is a **dilation**:
  - Dilation of an image A by a structuring element B is:

$$A \oplus B = \{z | (B)_z \cap A \neq \emptyset\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection of A and B is not the empty set (i.e. there are at least some true values: some points where both A and B are true), then this pixel z is true in the output image



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

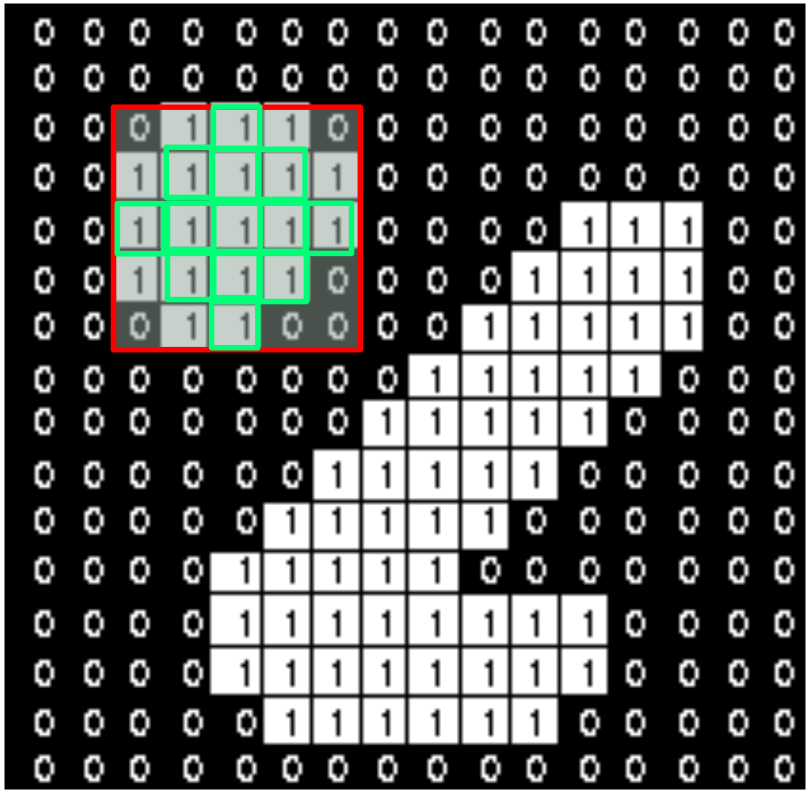
Morphological Operations on Images

- A basic morphological operation is a **dilation**:
  - Dilation of an image A by a structuring element B is:

$$A \oplus B = \{z | (B)_z \cap A \neq \emptyset\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection of A and B is not the empty set (i.e. there are at least some true values: some points where both A and B are true), then this pixel z is true in the output image

Pixel value in output: True



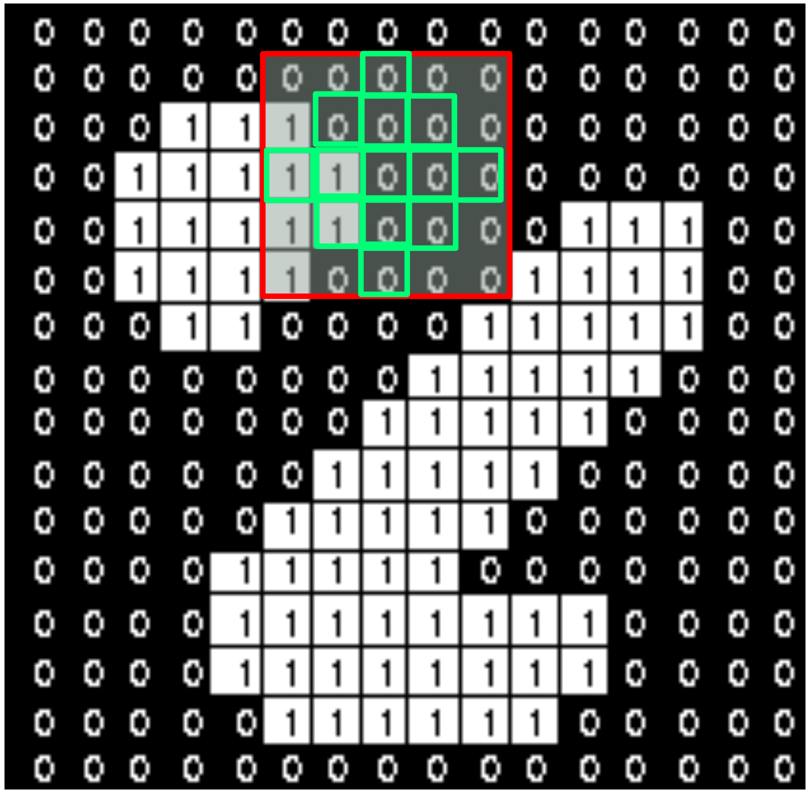
Morphological Operations on Images

- A basic morphological operation is a **dilation**:
  - Dilation of an image A by a structuring element B is:

$$A \oplus B = \{z | (B)_z \cap A \neq \emptyset\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection of A and B is not the empty set (i.e. there are at least some true values: some points where both A and B are true), then this pixel z is true in the output image

Pixel value in output: True



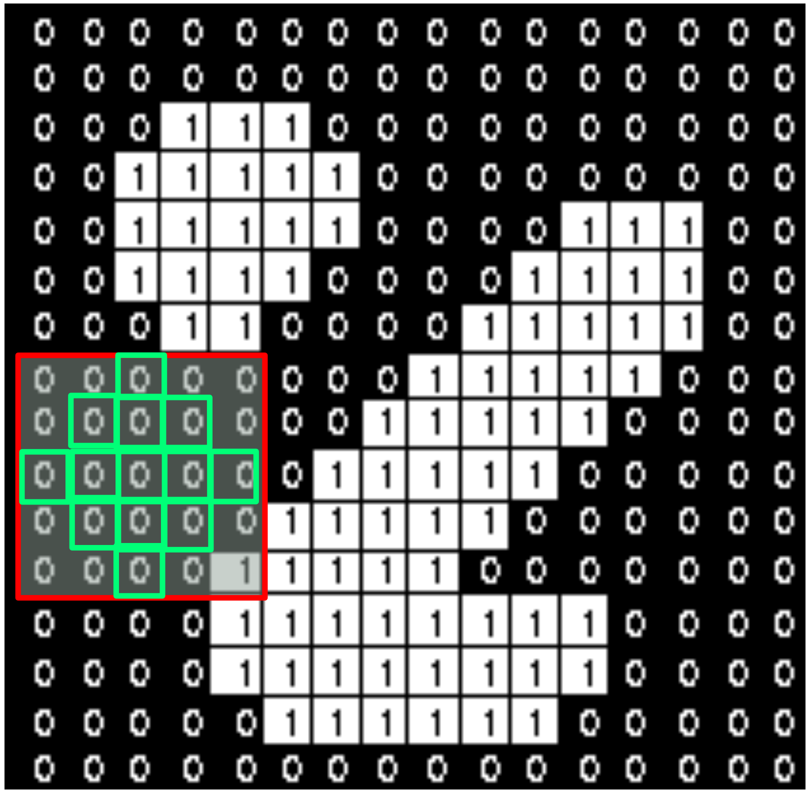
Morphological Operations on Images

- A basic morphological operation is a **dilation**:
  - Dilation of an image A by a structuring element B is:

$$A \oplus B = \{z | (B)_z \cap A \neq \emptyset\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection of A and B is not the empty set (i.e. there are at least some true values: some points where both A and B are true), then this pixel z is true in the output image

Pixel value in output: False



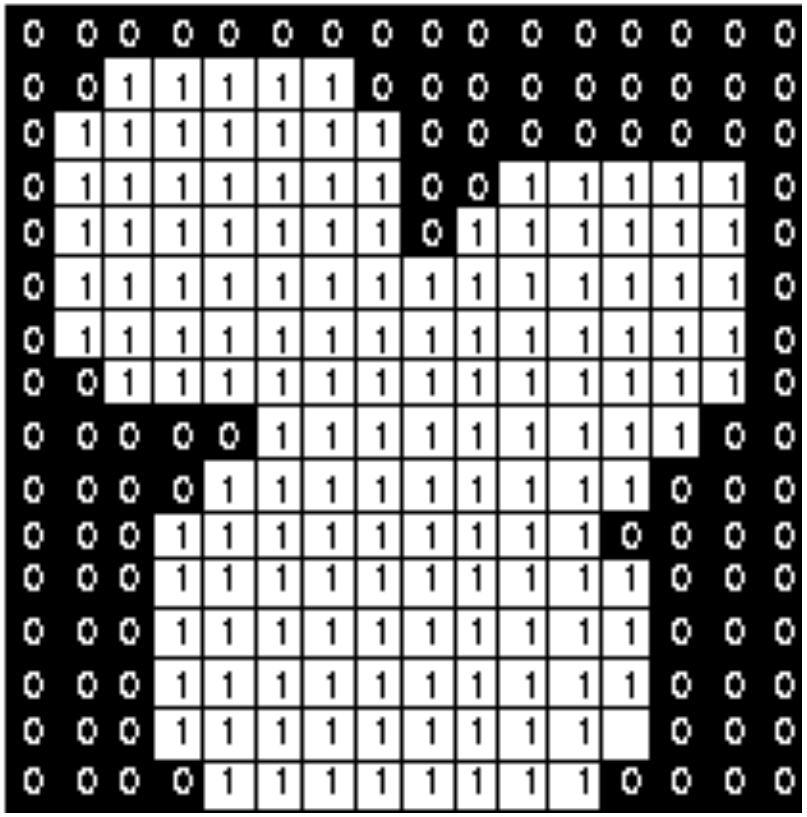
Morphological Operations on Images

- A basic morphological operation is a **dilation**:
  - Dilation of an image A by a structuring element B is:

$$A \oplus B = \{z | (B)_z \cap A \neq \emptyset\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection of A and B is not the empty set (i.e. there are at least some true values: some points where both A and B are true), then this pixel z is true in the output image

Output image



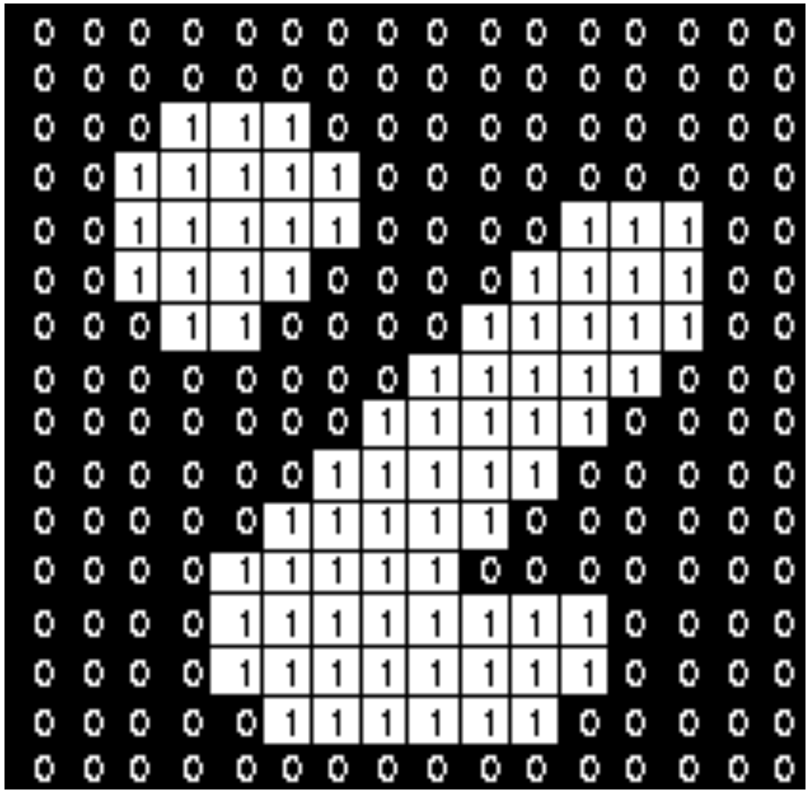
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0
0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1		0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

Morphological Operations on Images

- A dual operation of dilation is an **erosion**:
  - Erosion of an image A by a structuring element B is:

$$A \ominus B = \{z | (B)_z \cap A = (B)_z\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection between A and B is the same as the set B (i.e. for all B is true, A is also true), then this pixel z is true in the output image



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

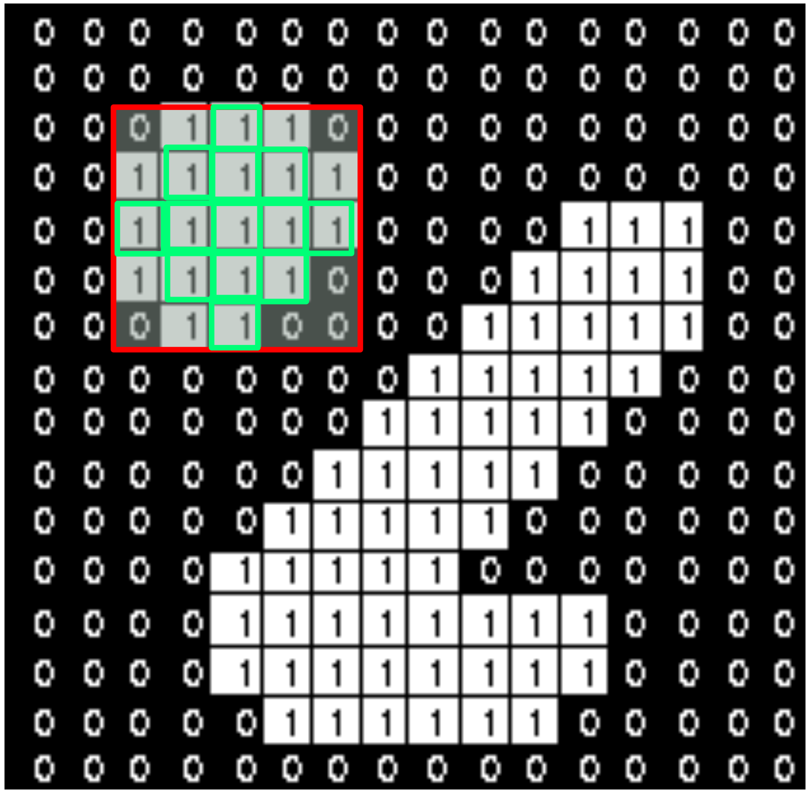
Morphological Operations on Images

- A dual operation of dilation is an **erosion**:
  - Erosion of an image A by a structuring element B is:

$$A \ominus B = \{z | (B)_z \cap A = (B)_z\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection between A and B is the same as the set B (i.e. for all B is true, A is also true), then this pixel z is true in the output image

Pixel value in output: True



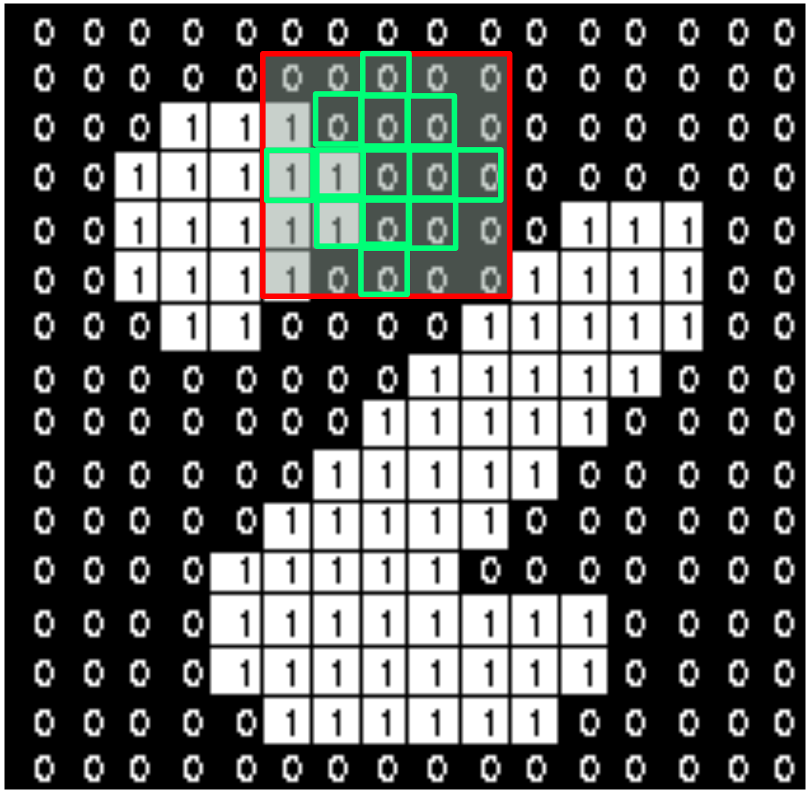
Morphological Operations on Images

- A dual operation of dilation is an **erosion**:
  - Erosion of an image A by a structuring element B is:

$$A \ominus B = \{z | (B)_z \cap A = (B)_z\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection between A and B is the same as the set B (i.e. for all B is true, A is also true), then this pixel z is true in the output image

Pixel value in output: False



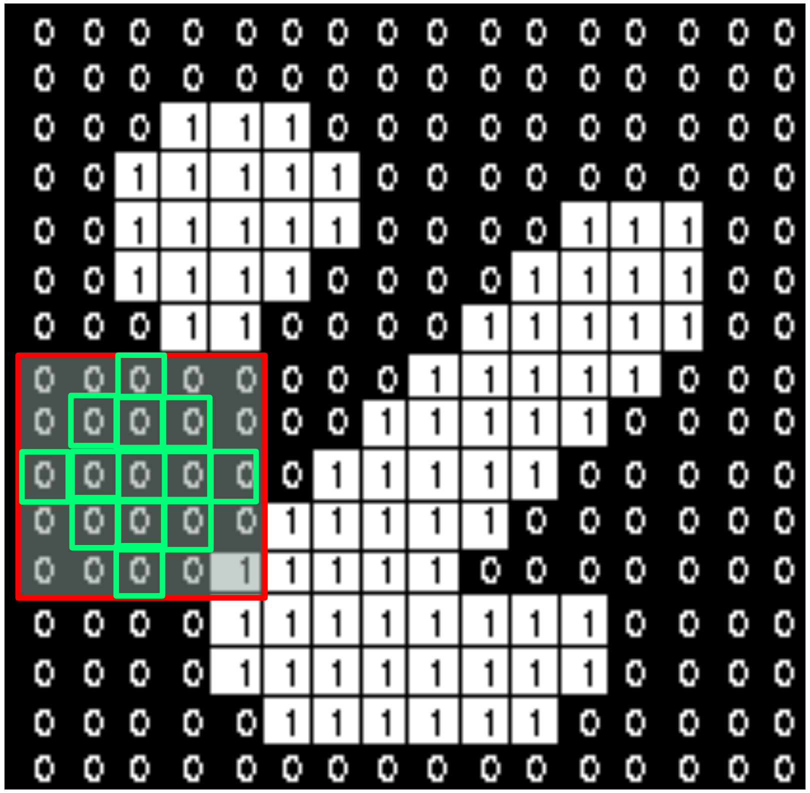
Morphological Operations on Images

- A dual operation of dilation is an **erosion**:
  - Erosion of an image A by a structuring element B is:

$$A \ominus B = \{z | (B)_z \cap A = (B)_z\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection between A and B is the same as the set B (i.e. for all B is true, A is also true), then this pixel z is true in the output image

Pixel value in output: False



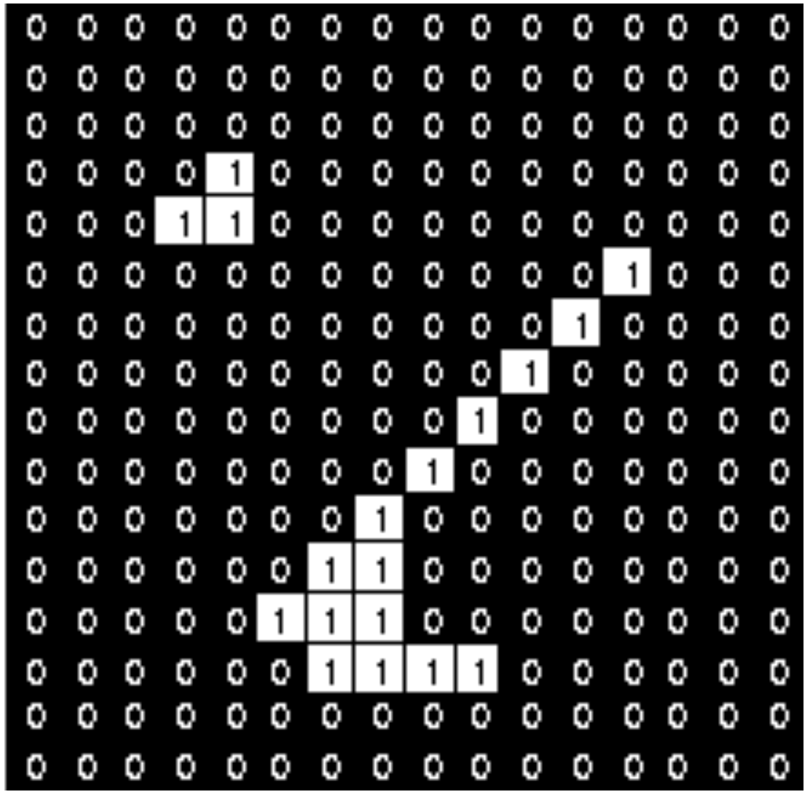
Morphological Operations on Images

- A dual operation of dilation is an **erosion**:
  - Erosion of an image A by a structuring element B is:

$$A \ominus B = \{z | (B)_z \cap A = (B)_z\}$$

- This means for each pixel location z we apply the template B, centered at this location: if the intersection between A and B is the same as the set B (i.e. for all B is true, A is also true), then this pixel z is true in the output image

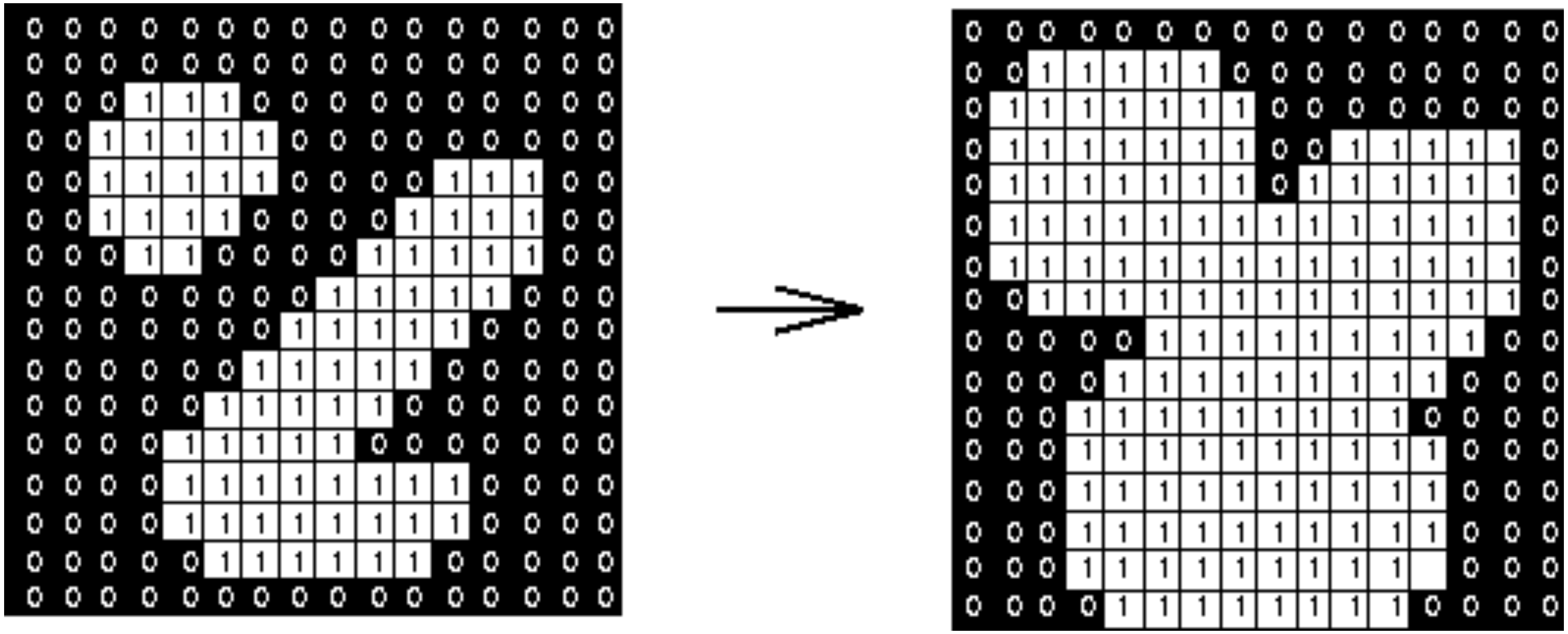
Output image



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Dilation and Erosion

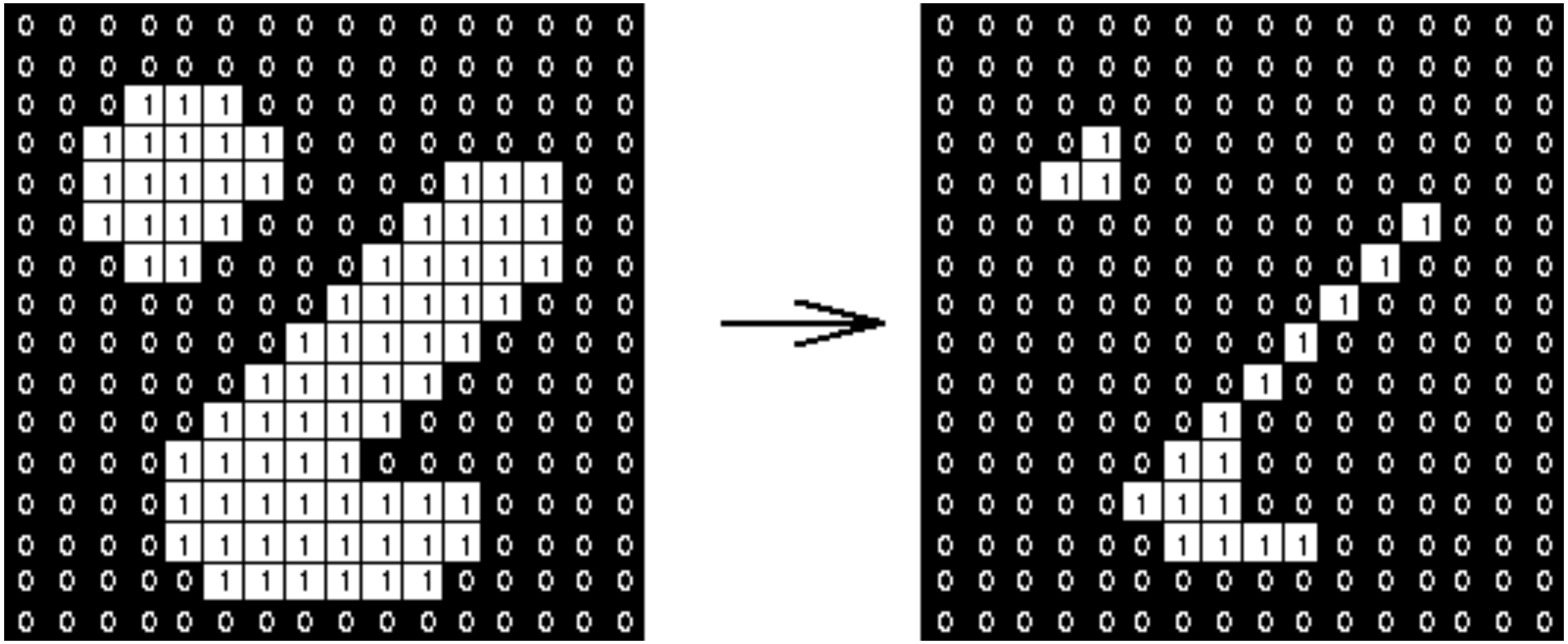
- Dilation:
  - each pixel is set to true if any of the pixels in the neighborhood is true



https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

Dilation and Erosion

- Erosion:
  - each pixel is set to false if any of the pixels in the neighborhood is false



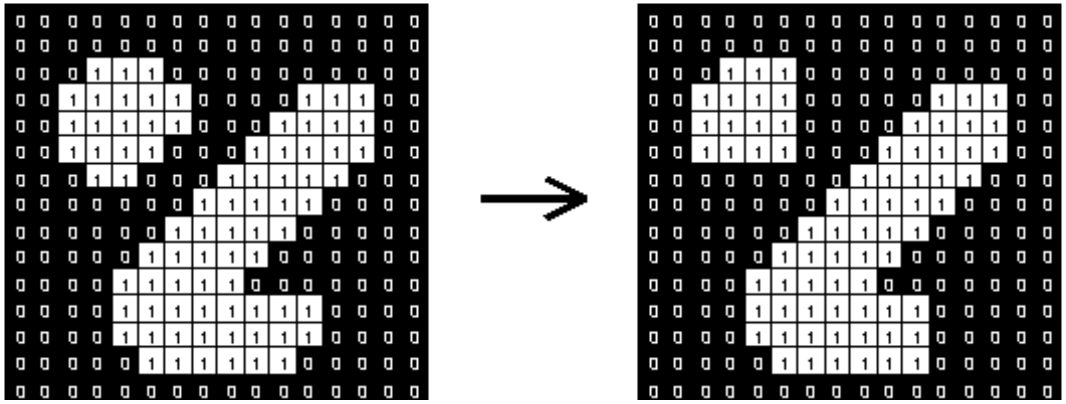
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

Opening and Closing

- Opening: $(A \ominus B) \oplus B$
  - erosion of an image followed by a dilation with the same structuring element
  - Removes thin/small regions and maintains the size of surviving objects

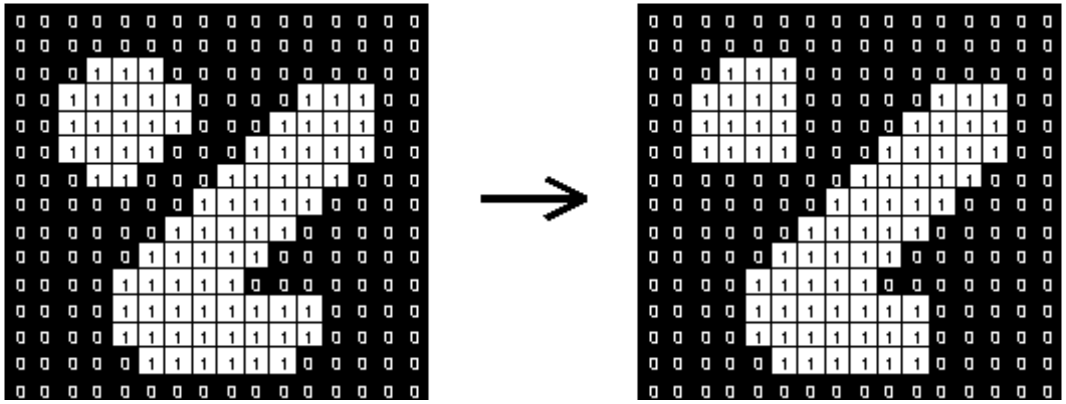


- Closing: $(A \oplus B) \ominus B$
  - dilation of an image followed by erosion with the same structuring element
  - Fills holes in regions while maintaining their original size

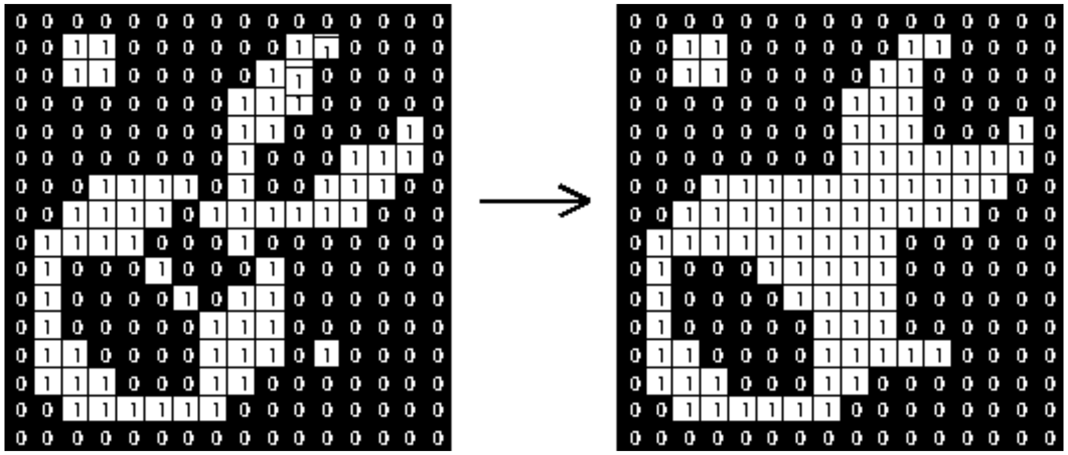
https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

Opening and Closing

- Opening: $(A \ominus B) \oplus B$
  - erosion of an image followed by a dilation with the same structuring element
  - Removes thin/small regions and maintains the size of surviving objects

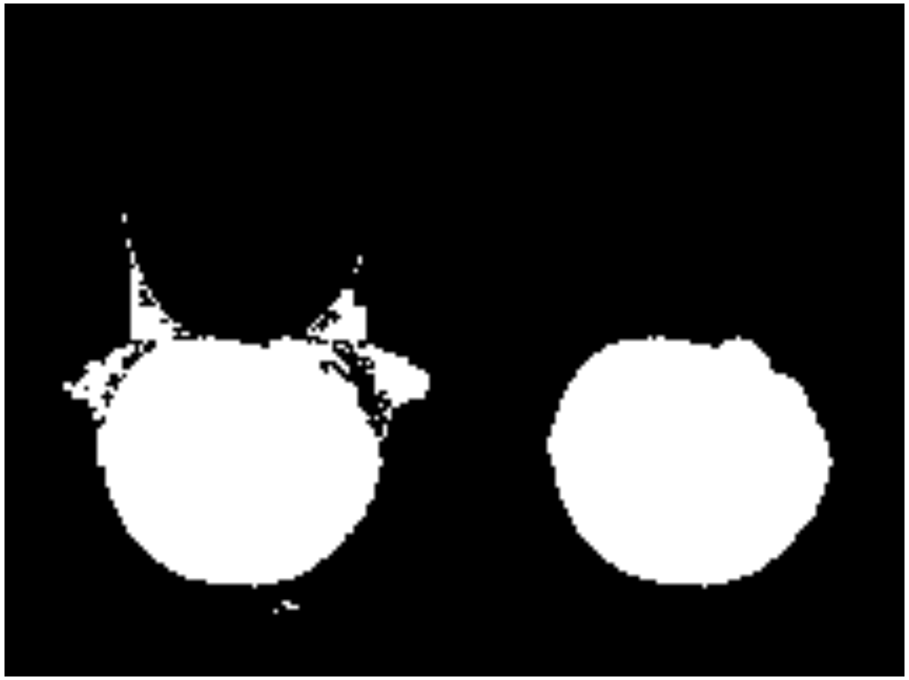
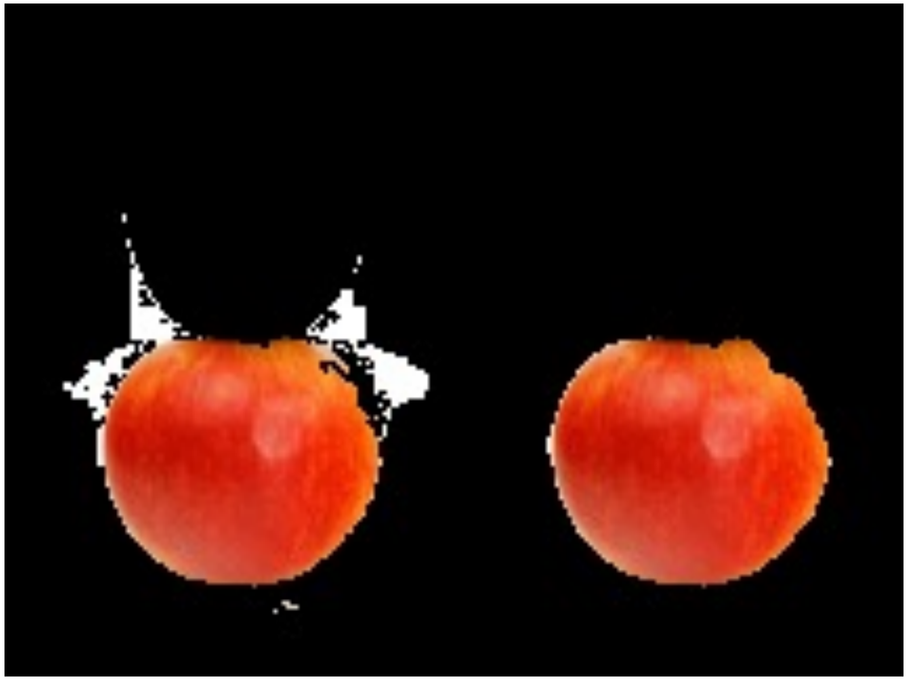


- Closing: $(A \oplus B) \ominus B$
  - dilation of an image followed by erosion with the same structuring element
  - Fills holes in regions while maintaining their original size



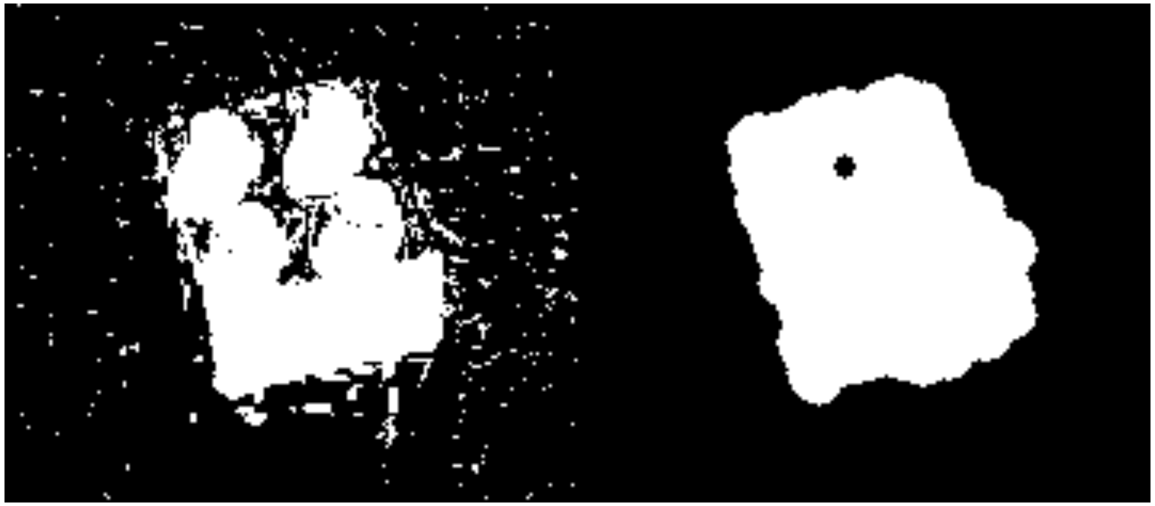
https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

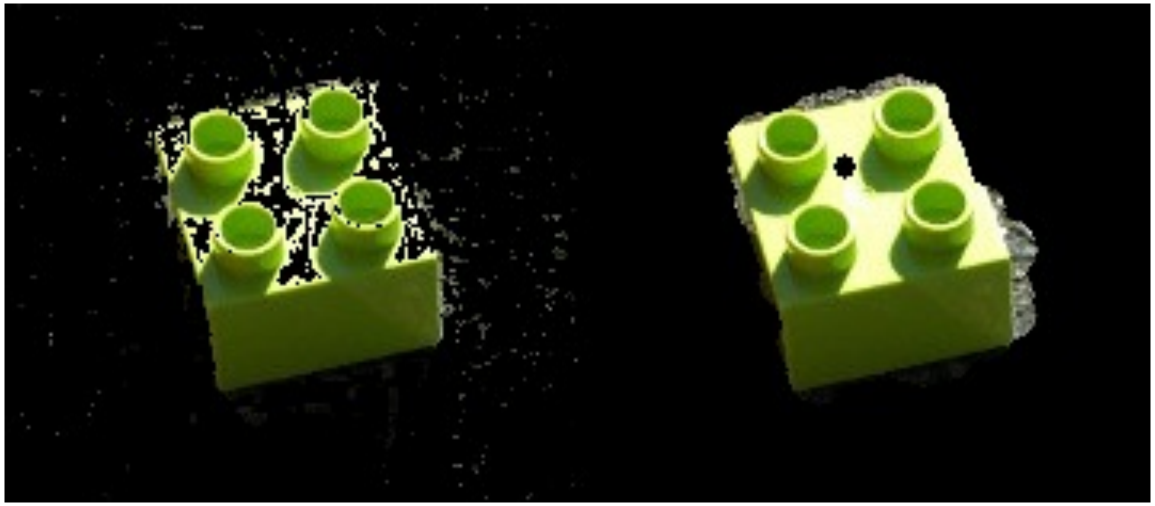
Opening and Closing

Applying image opening using a 4x4 disk structuring element

Opening and Closing



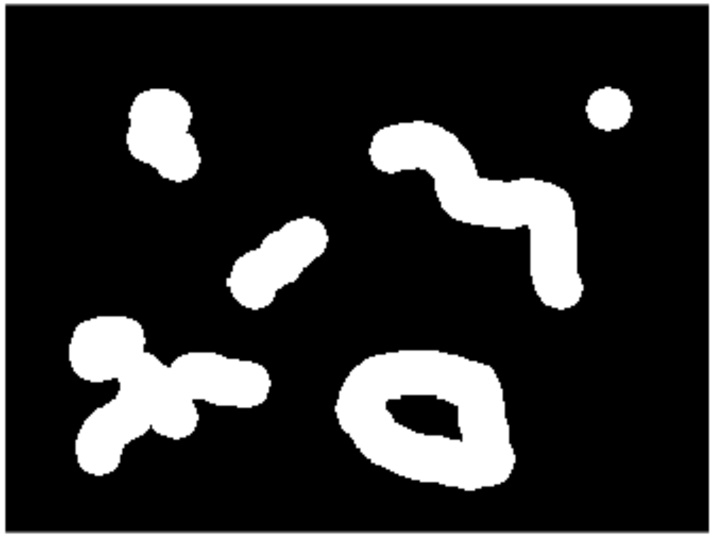


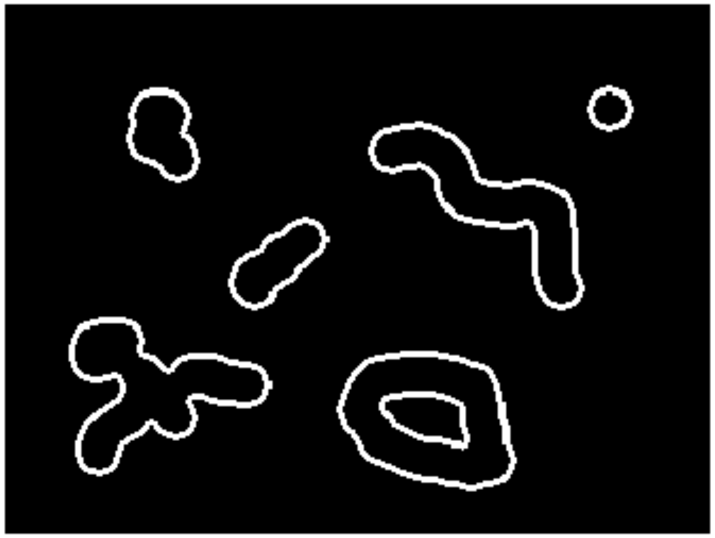
Closing using a 3x3 disk structuring element followed by opening using a 10x10 disk structuring element

Boundary Extraction

- The extraction of boundary pixels around a region can be achieved by taking the difference between the image and the image eroded by a structuring element B:

$$A - (A \ominus B) = A \cap (A \ominus B)^c$$

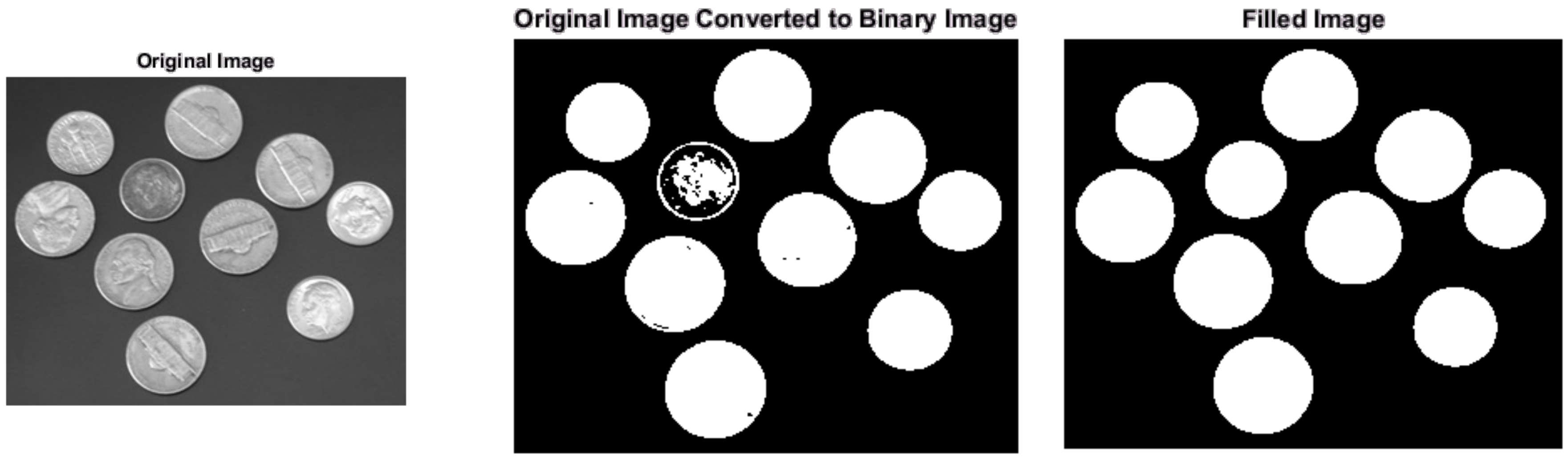

Original Image


Boundary extraction with 3x3 disk

Region Filling

- Region filling across an image A can be achieved by starting from a set of pixels X_k inside a set of boundaries, and iteratively performing a dilation using B while maintain output pixels that intersect with the compliment of A (Iteration ceases when $X_{k+1} = X_k$):

$$X_{k+1} = (X_k \oplus B) \cap (A)^c$$

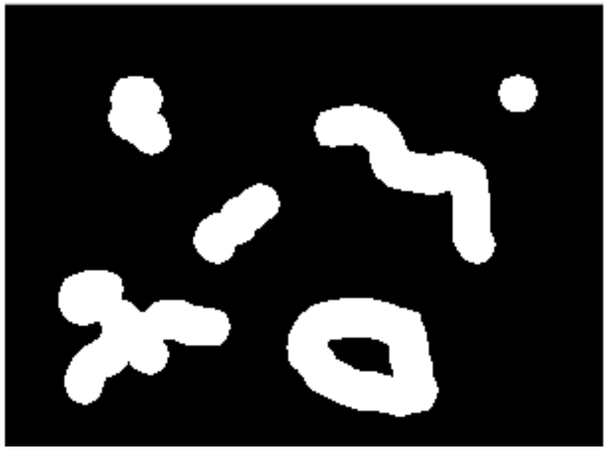
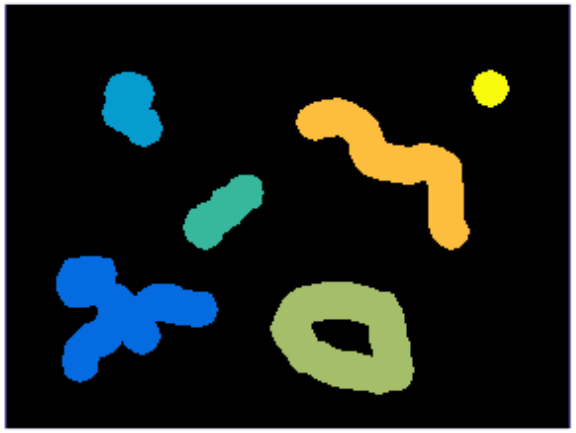


Two-stage hole filling: (1) region fill starting with pixels on edges on image, (2) region filling starting with the compliment of (1) unioned with all true pixels in A

Connected Components

- All pixels contained in a connected component of an image A can be found by starting from some subset of the component and iteratively taking the intersection between dilations with A itself (until $X_{k+1} = X_k$):

$$X_{k+1} = (X_k \oplus B) \cap (A)$$

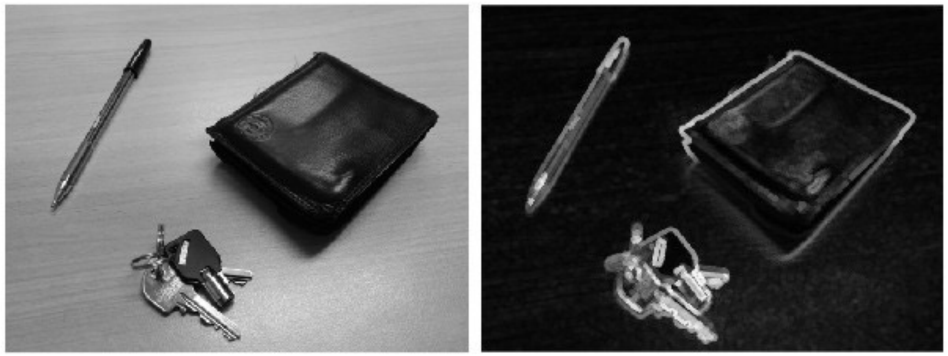



- The number of connected components and their pixels in an image can be calculated by:
  - (1) apply the above algorithm to the first true pixel in the image until convergence
  - (2) record the component and label all connected pixels as "searched"
  - (3) repeat (1) on next true pixel not in searched until no pixels left

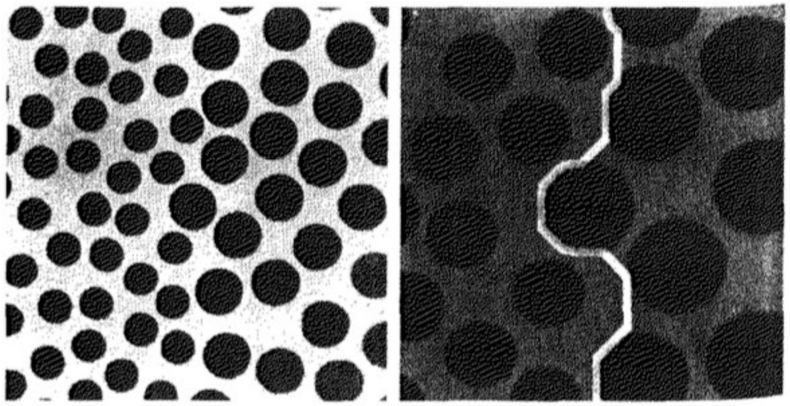
Morphological Operations on Grayscale Images

- Morphological operations can be applied to grayscale images:
  - Dilation: each pixel equals the **maximum** of the input pixel's neighborhood
  - Erosion: each pixel equals the **minimum** of the input pixel's neighborhood
- Morphological operations on grayscale images can be used for example for finding edges and distinguishing between textures

Intensity gradient computed using the difference between dilation and erosion



Texture boundary computed by progressively closing the image using larger structuring elements until one texture disappears



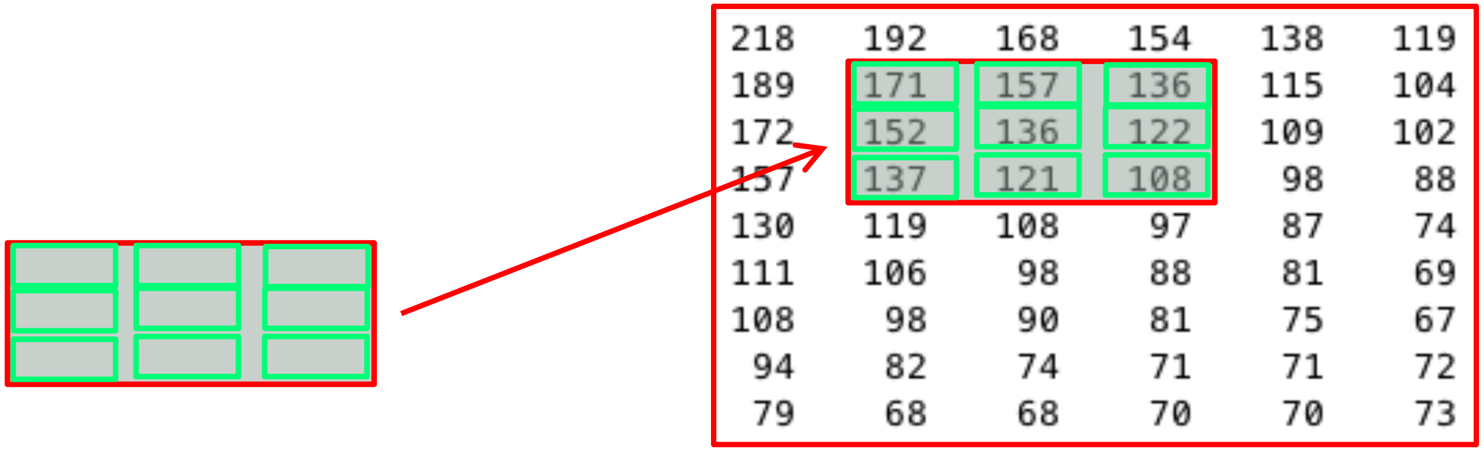
Assignment 1

- Currently available on Canvas, due 28th August (end of week 4)
- Two Questions:
  - Photometric stereo
  - Colour-based object tracking
- Make sure to follow submission instructions: two separate portals, one for report, other for code

5 minute break

Spatial Filtering of Images

- Morphological operations involve a structuring element which applies logical/set operations on the underlying image data
- **Image filtering** involve operations that use a type of structural element (often referred to as a *kernel* or *template*) that applies numerical calculations:
  - Image kernels typically have numeric values associated with each element of their raster data

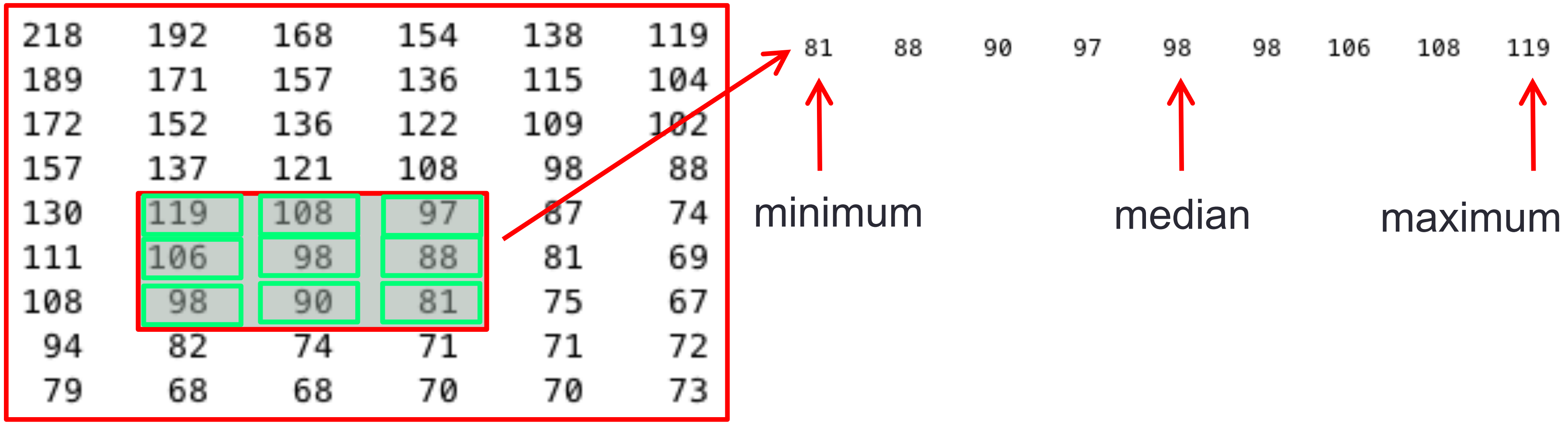


218	192	168	154	138	119
189	171	157	136	115	104
172	152	136	122	109	102
157	137	121	108	98	88
130	119	108	97	87	74
111	106	98	88	81	69
108	98	90	81	75	67
94	82	74	71	71	72
79	68	68	70	70	73

- Filters can be used for a range of tasks including noise removal, image enhancement, edge detection and the description of textural characteristics in an image

Order Statistics Filters

- Order-statistic filters are a family of non-linear filters that compute the order-statistics of pixels within a given template region
- Pixels within a given template are ordered (sorted) and an output returned based on the value at a specific ranking level (i.e. minimum, maximum, median, 90% percentile etc.)
- Median filters are one example of order-statistic filters, and are effective at removing salt-and-pepper noise (i.e. noise of high/low intensity values)



218	192	168	154	138	119
189	171	157	136	115	104
172	152	136	122	109	102
157	137	121	108	98	88
130	119	108	97	87	74
111	106	98	88	81	69
108	98	90	81	75	67
94	82	74	71	71	72
79	68	68	70	70	73

```
81   88   90   97   98   98   106   108   119
```

Order Statistics Filters: Median Filter

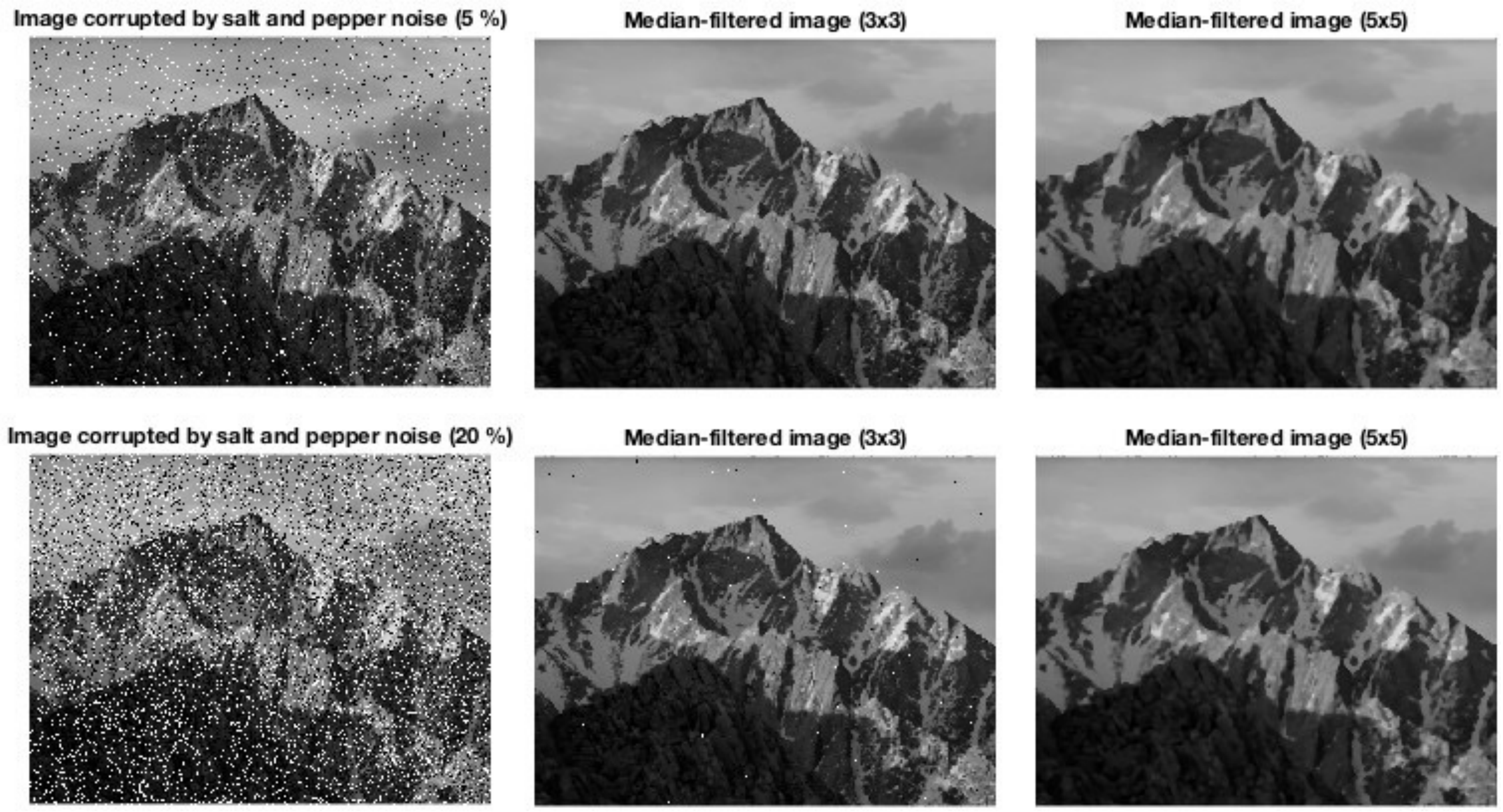


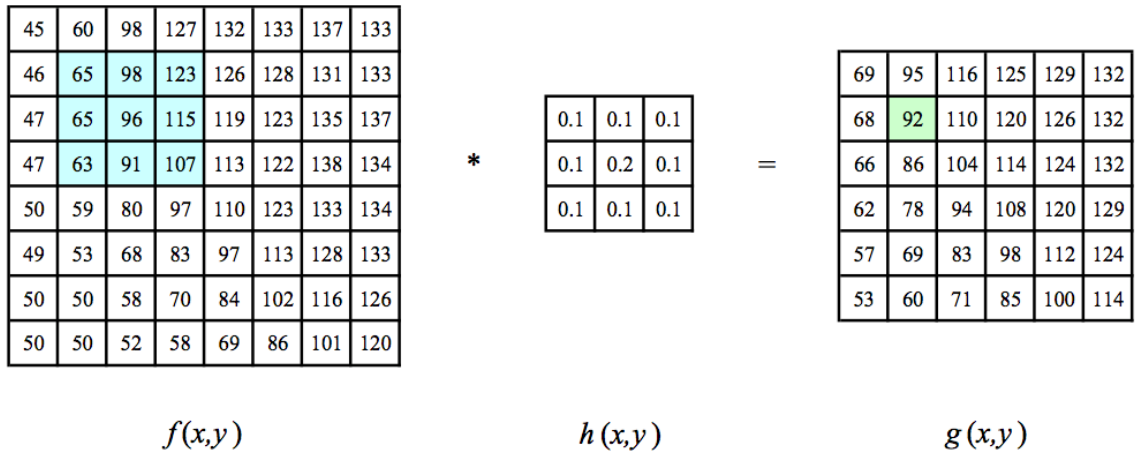
Image corrupted by salt and pepper noise (5 %)
Median-filtered image (3x3)
Median-filtered image (5x5)
Image corrupted by salt and pepper noise (20 %)
Median-filtered image (3x3)
Median-filtered image (5x5)

Linear filters

- For linear filters, the value of an output pixel is determined as a weighted sum of the input pixel values:

$$g(i,j) = \sum_k \sum_l f(i+k, j+l)h(k,l) \qquad \text{or} \qquad g = f \otimes h$$

Where f is the input image, h is the kernel and i,j,k,l are pixel coordinates



$f(x,y)$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

*

$h(x,y)$

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

=

$g(x,y)$

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

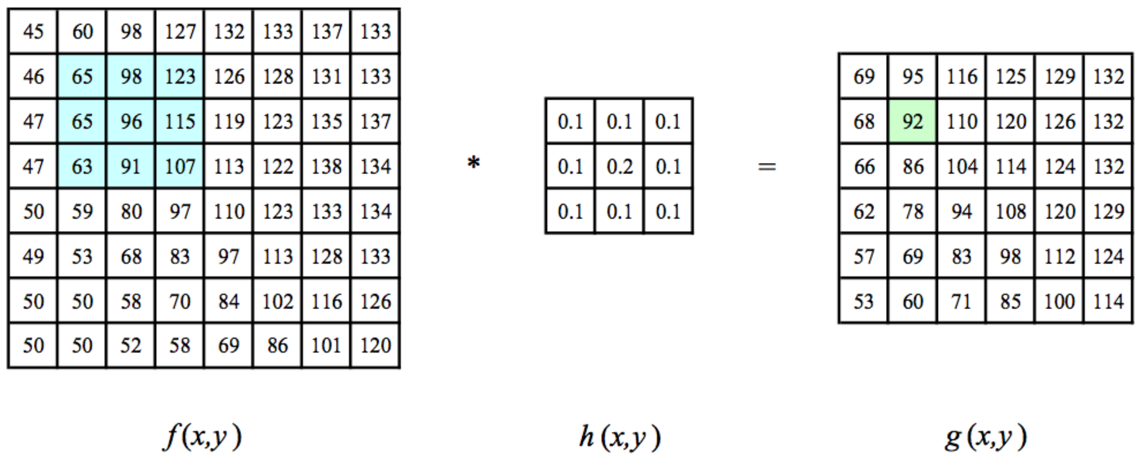
Linear filters

- For linear filters, the value of an output pixel is determined as a weighted sum of the input pixel values:

$$g(i,j) = \sum_k \sum_l f(i+k, j+l) h(k,l) \qquad \text{or} \qquad g = f \otimes h$$

Where f is the input image, h is the kernel and i,j,k,l are pixel coordinates

- Operations in this form are called **Linear Shift-Invariant (LSI)** operators which obey both the super position and shift-invariance:



45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x,y)$

$$h \otimes (f_o + f_1) = h \otimes f_o + h \otimes f_1$$

$$g(i,j) = f(i+k, j+l) \leftrightarrow (h \otimes f)(i,j) = (h \otimes f)(i+k, j+l)$$

Image Smoothing Filters

- The most straightforward application of linear filters is in image smoothing:
  - An **average filter** takes an equally weighted sum of the surrounding pixel values in a local mask region:

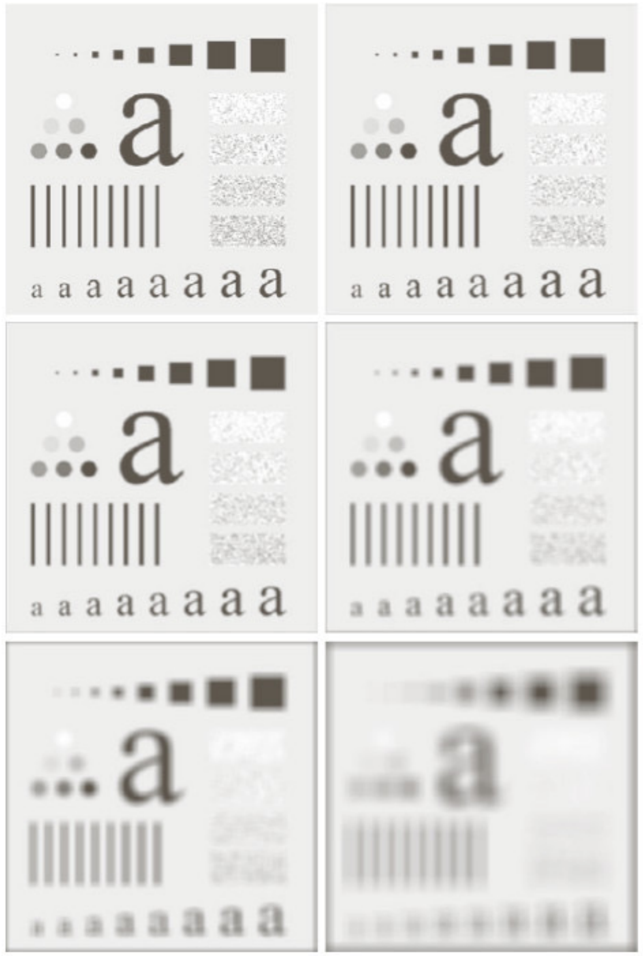
3x3 averaging kernel

0.1111	0.1111	0.1111
0.1111	0.1111	0.1111
0.1111	0.1111	0.1111

0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400

5x5 averaging kernel

  - High-frequency noise (and information) is attenuated from the image, increased blur



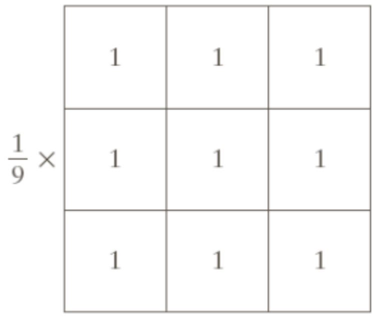
Average filtering (500x500 pixel image) with kernel sizes: 3,5,9,15,35

Image Smoothing

- Image smoothing kernels can provide different arrangements of weights depending on distance from the center of the kernel:
- A kernel based on a two-dimensional Gaussian function is a common choice for smoothing:

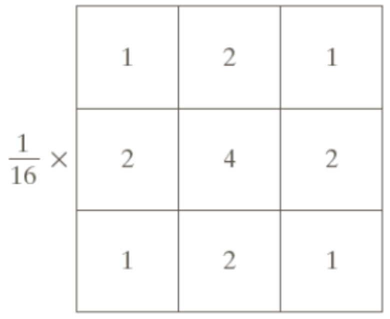
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- The size of the raster for a specified **σ** typically chosen to maintain > 99% of the probability mass of the function



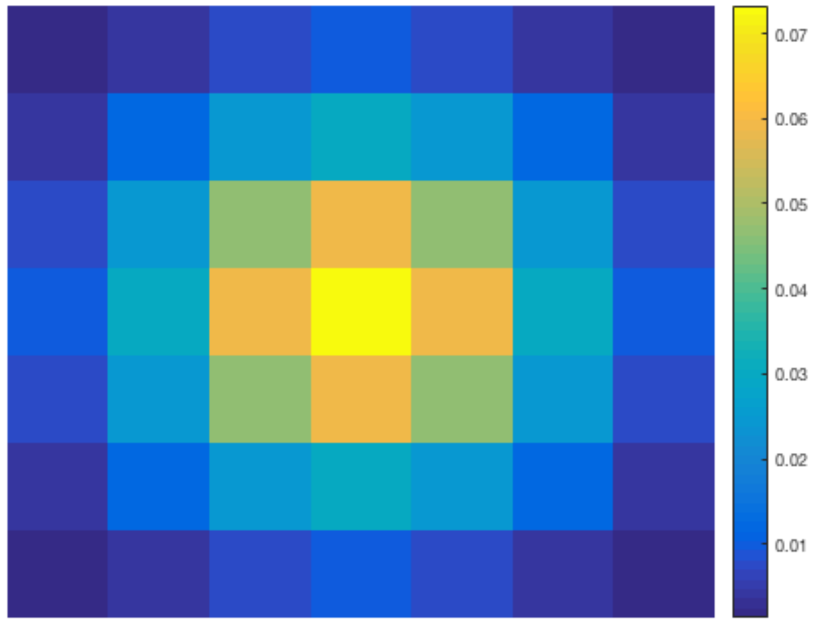
$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Averaging filter



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

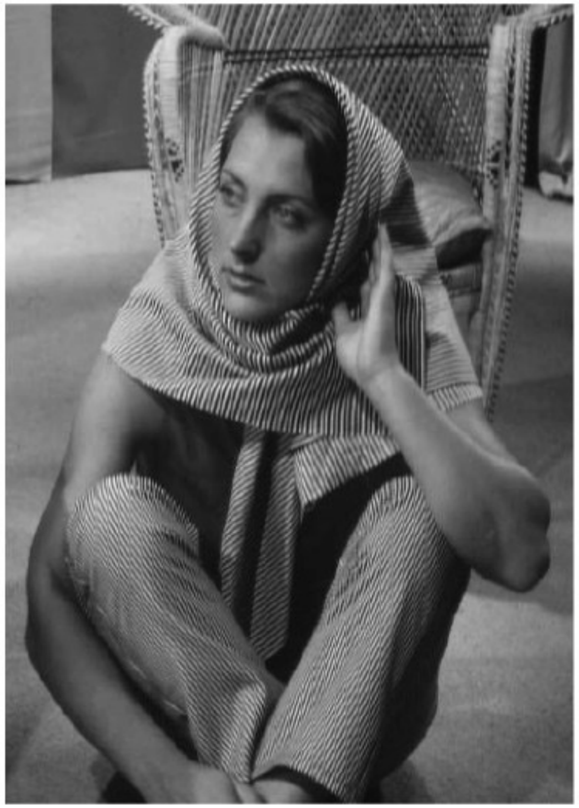
Weighted average

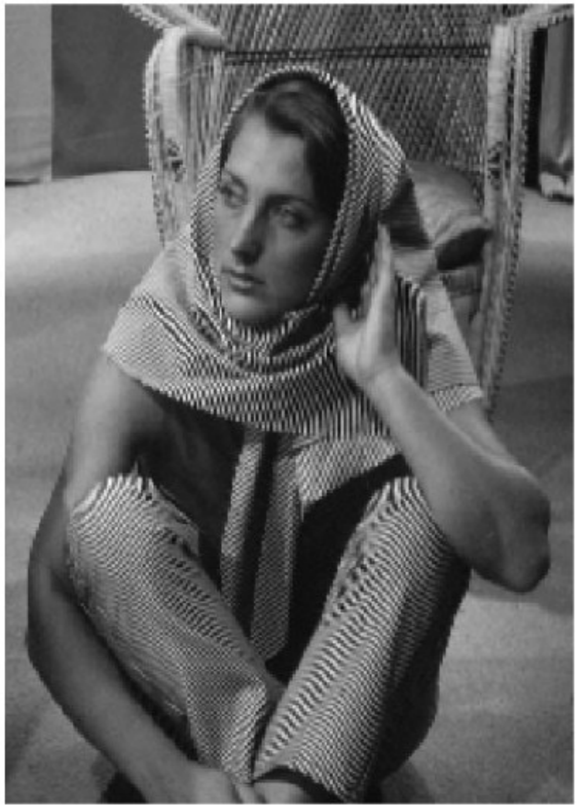


7x7 Gaussian kernel with **σ** = 1.5

Image Smoothing

- Other applications of filters: aliasing that occurs during down-sampling can be removed by applying a Gaussian filter first


Original


Down-sampled 50 %

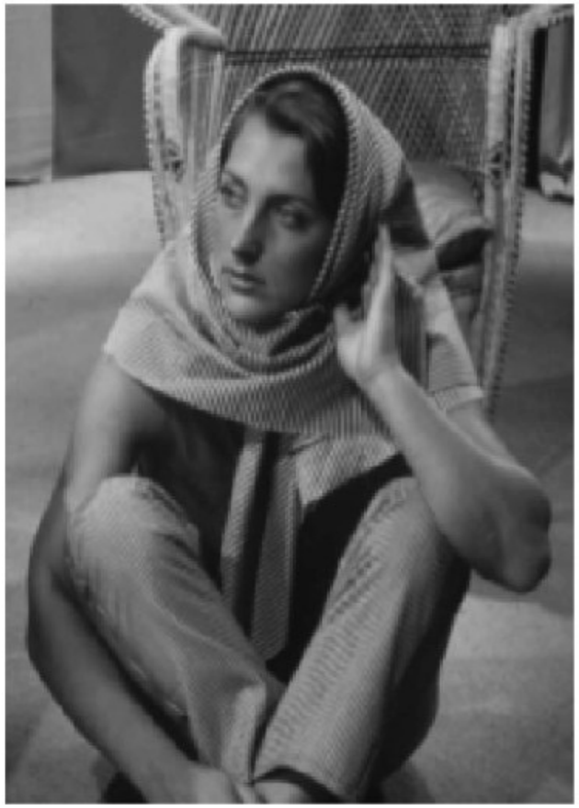

Gaussian filtered 3x3, followed by down-sample 50 %

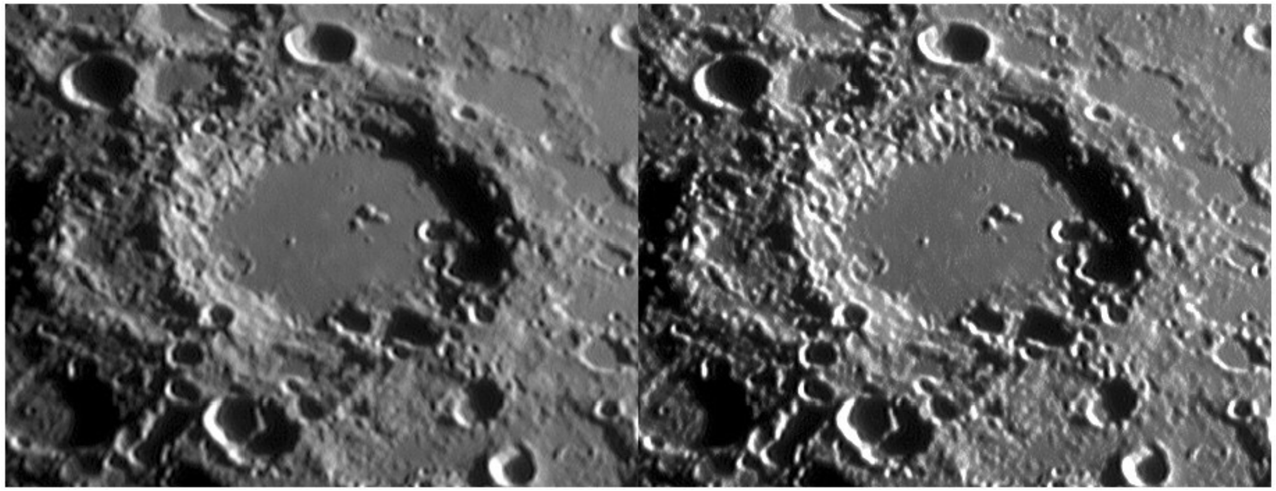
Image Sharpening Filters

- Averaging filters remove high-frequency components of the image data: By extracting the resulting low-frequency response away from the original image data, the high-frequency components are artificially amplified:

$$g = f + \gamma(f - f \otimes h)$$

Where f is the original image, g the sharpened image, h is a smoothing kernel and **γ** is a scalar parameter effecting the degree of sharpening

- The resulting filter is known as a sharpening filter and can be used to enhance the strength of edges in the image



Original | **γ** = 1.1 and **σ** = 4.0

Gradient and Laplacian Operators

- Image sharpening algorithms can also use information about the first and second derivatives in an image to accentuate high-frequency potions of the image
- For discrete image data, finite differences can be used to compute the first and second-order partial derivatives of image data in a direction x:

$$\frac{\delta f}{\delta x} = f(x+1) - f(x) \qquad \frac{\delta^2 f}{\delta x^2} = f(x+1) + f(x-1) - 2f(x)$$

Gradient and Laplacian Operators

- Image sharpening algorithms can also use information about the first and second derivatives in an image to accentuate high-frequency potions of the image
- For discrete image data, finite differences can be used to compute the first and second-order partial derivatives of image data in a direction x:

$$\frac{\delta f}{\delta x} = f(x+1) - f(x) \qquad \frac{\delta^2 f}{\delta x^2} = f(x+1) + f(x-1) - 2f(x)$$

- The Laplacian is a measure in both dimensions in the image (x,y) of second order derivative and can be computed using a single kernel function:

$$\nabla^2 f = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}$$

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

- Which can then be added back to the original image to enhance sharp features

Gradient and Laplacian Operators

- Image sharpening algorithms can also use information about the first and second derivatives in an image to accentuate high-frequency potions of the image
- For discrete image data, finite differences can be used to compute the first and second-order partial derivatives of image data in a direction x:

$$\frac{\delta f}{\delta x} = f(x+1) - f(x) \qquad \frac{\delta^2 f}{\delta x^2} = f(x+1) + f(x-1) - 2f(x)$$

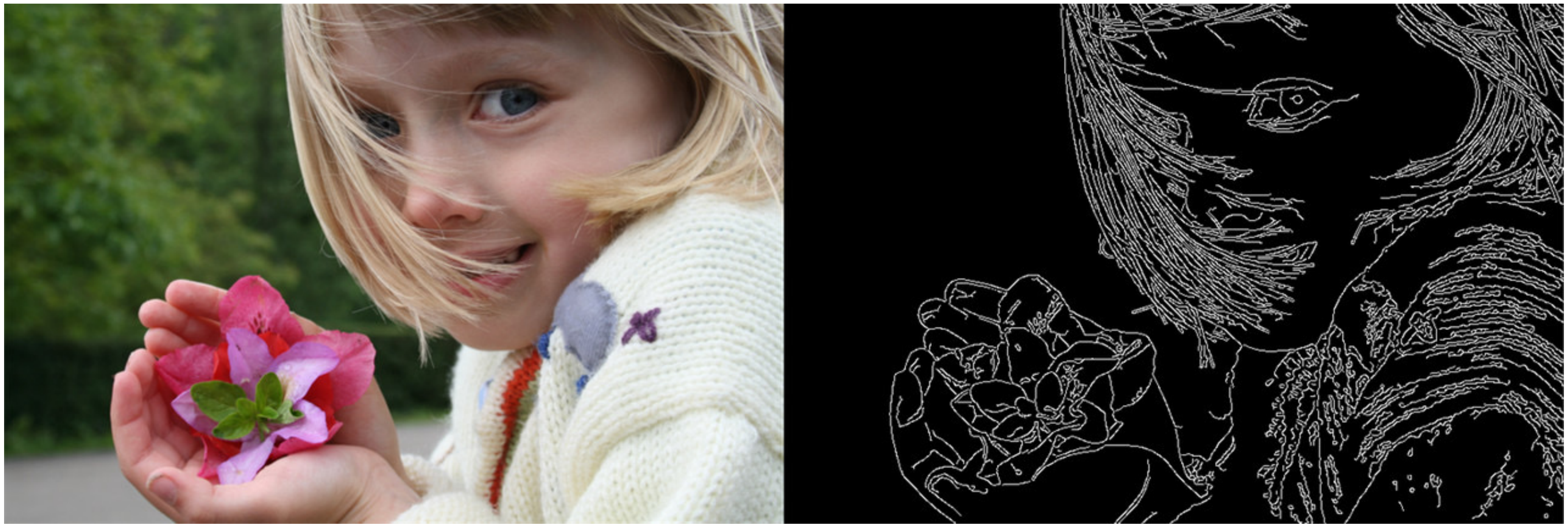
- The magnitude of the first-order derivative (gradient) of the image data can also be used to help sharpen an image:

$$\nabla \mathbf{f} = [G_x, G_y] = [\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}] \qquad |\nabla \mathbf{f}| = (G_x^2 + G_y^2)^{1/2}$$

- Which can then be added back to the original image to enhance sharp features

Edge Detection

- Gradients in the image data are also useful for finding edges in images (i.e. regions of sharp contrast from one value to another)
- Edges are typically of interest in image analysis:
  - Often correspond to the boundaries of objects
  - Correspond to occluding contours in images
  - Correspond to sudden changes in lighting on surface orientation



Jon McLoone [CC BY-SA 3.0]
(https://en.wikipedia.org/wiki/Edge_detection#/media/File:%C3%84%C3%A4retuvastuse_n%C3%A4ide.png)

Sobel Edge Detector

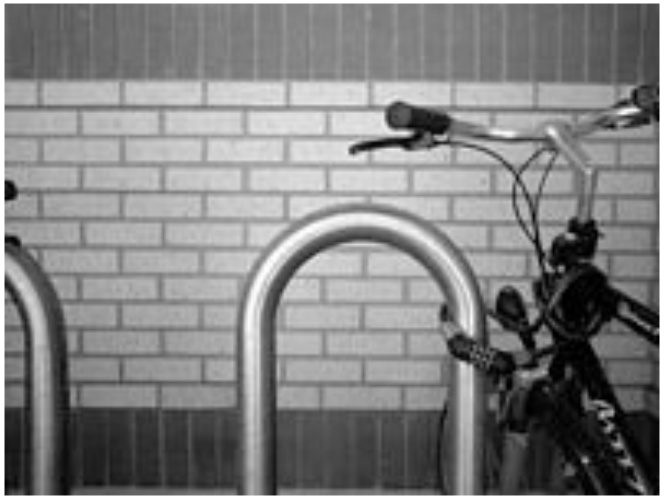
- The Sobel operator is an approximation to the gradient that is used commonly in computing simple edges in images
- From both a horizontal and vertical gradient filter, an approximation of the gradient is found (edge magnitude) and thresholded to produce an edge image

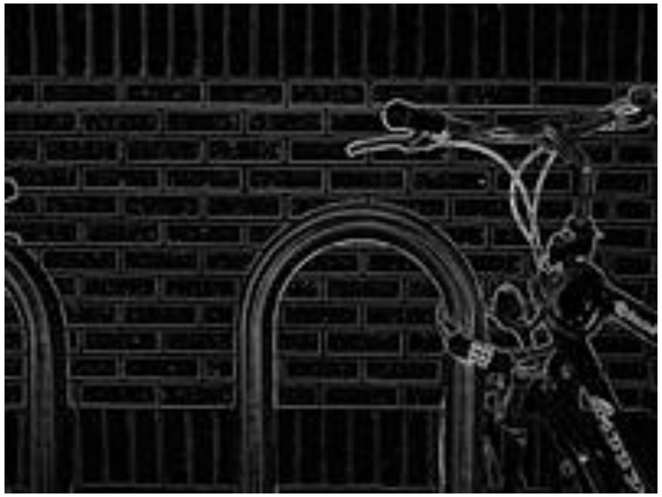
$$S_1 = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad S_2 = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

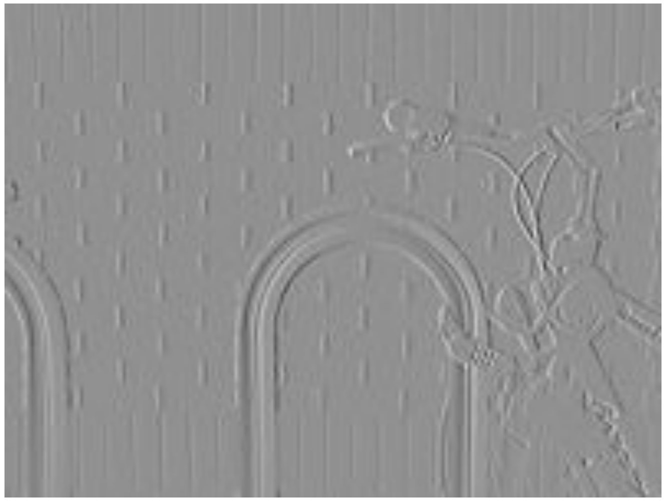
$$\text{Edge Magnitude} = \sqrt{S_1^2 + S_1^2}$$

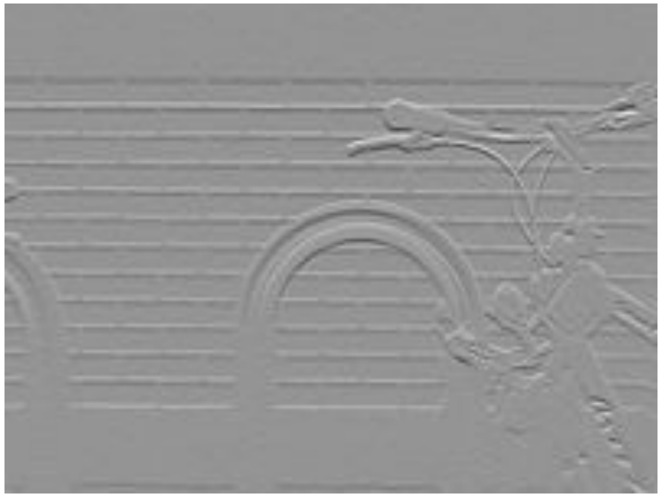
$$\text{Edge Direction} = \tan^{-1}\left(\frac{S_1}{S_2}\right)$$

Sobel Edge Detector

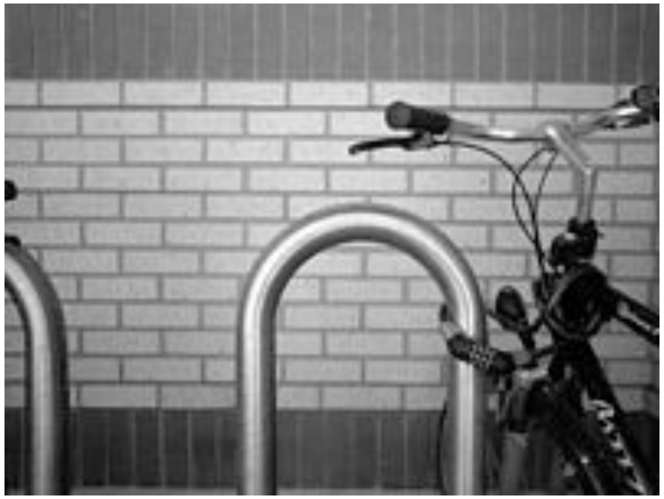

Original

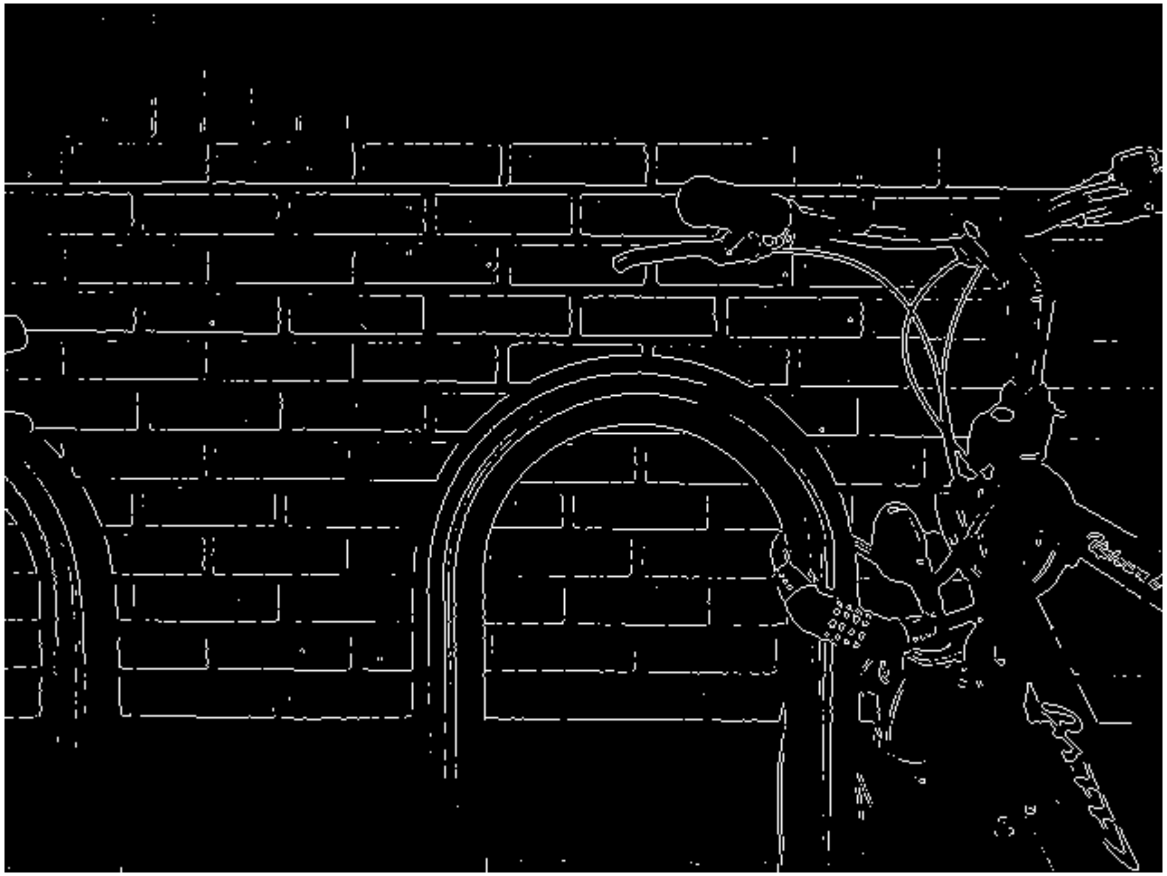

Normalized Gradient Magnitude


Normalized x-gradient


Normalized y-gradient

Sobel Edge Detector


Original


Sobel Edges

The Canny Edge Detector

- The degree of smoothing in the operator effects two aspects of edge finding:
  - The measured strength of an edge
  - The accuracy to which an edge is located (width of the response)
- In general there is a tradeoff between the quality of detected edges and how well they are localised:
  - Slight smoothing: Good localisation, but poor detection (too many edges)
  - More smoothing: Poor localisation, but good detection (only strong edges)

- The Canny Edge detector is an extended approach to edge detection that optimises the filtering kernel (using first-order derivative of a Gaussian) to balance localisation and detection performance:
  - Also incorporates post-processing steps to maximise “true” edges

The Canny Edge Detector

- The degree of smoothing in the operator effects two aspects of edge finding:
  - The measured strength of a edge
  - The accuracy to which an edge is located (width of the response)
- In general there is a tradeoff between the quality of detected edges and how well they are localised:
  - Slight smoothing: Good localisation, but poor detection (too many edges)
  - More smoothing: Poor localisation, but good detection (only strong edges)

- The Canny Edge detector is an extended approach to edge detection that optimises the filtering kernel (using first-order derivative of a Gaussian) to balance localisation and detection performance:
  - Also incorporates post-processing steps to maximise “true” edges

The Canny Edge Detector

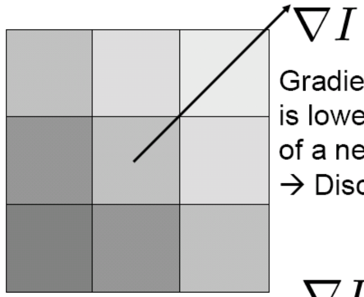
- The Canny edge detector computes edges in images using the following steps:
  - (1) Smooths the image with a Gaussian kernel
  - (2) Computes the gradient magnitude and direction (typically using a Sobel operator)
  - (3) Suppress non-maxima pixels in the direction of the gradient
  - (4) Edge tracking: uses two thresholds to keep either strong edges or weak edges that are connected to strong edges

The Canny Edge Detector

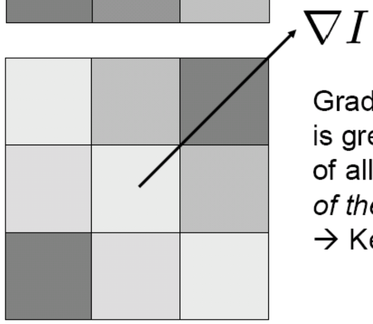
- The Canny edge detector computes edges in images using the following steps:
  - (1) Smooths the image with a Gaussian kernel
  - (2) Computes the gradient magnitude and direction (typically using a Sobel operator)
  - (3) Suppress non-maxima pixels in the direction of the gradient
  - (4) Edge tracking: uses two thresholds to keep either strong edges or weak edges that are connected to strong edges

The Canny Edge Detector

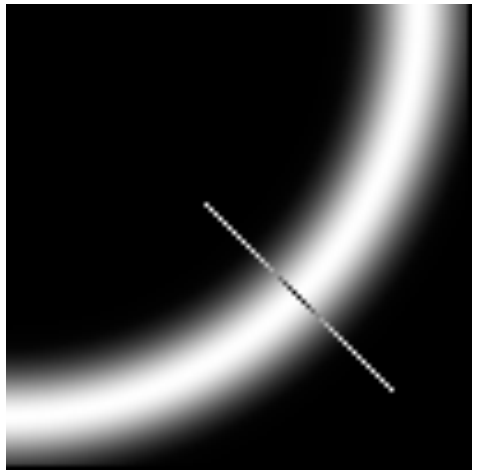
- The Canny edge detector computes edges in images using the following steps:
  - (1) Smooths the image with a Gaussian kernel
  - (2) Computes the gradient magnitude and direction (typically using a Sobel operator)
  - (3) Suppress non-maxima pixels in the direction of the gradient
  - (4) Edge tracking: uses two thresholds to keep either strong edges or weak edges that are connected to strong edges



Gradient magnitude at center pixel is lower than the gradient magnitude of a neighbor *in the direction of the gradient*
→ Discard center pixel (set magnitude to 0)

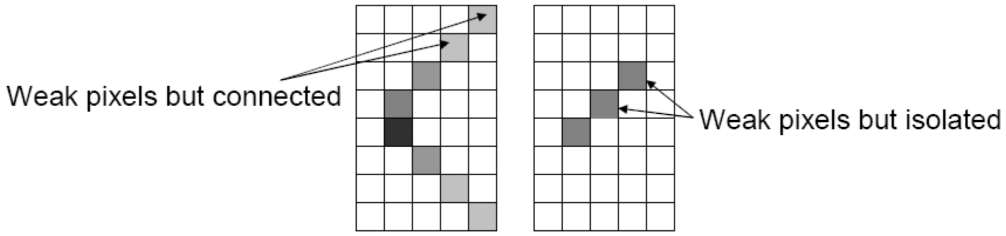


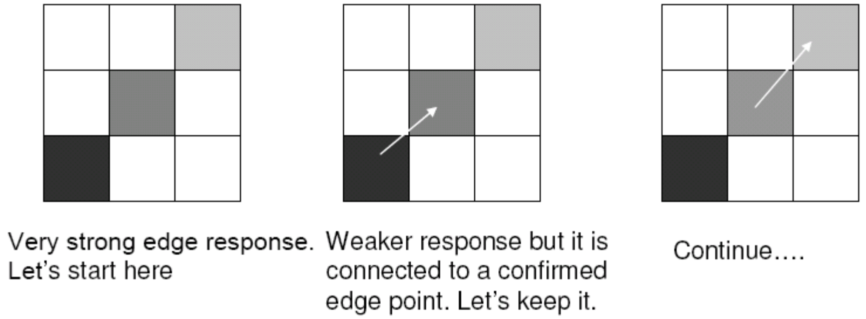
Gradient magnitude at center pixel is greater than gradient magnitude of all the neighbors *in the direction of the gradient*
→ Keep center pixel unchanged



The Canny Edge Detector

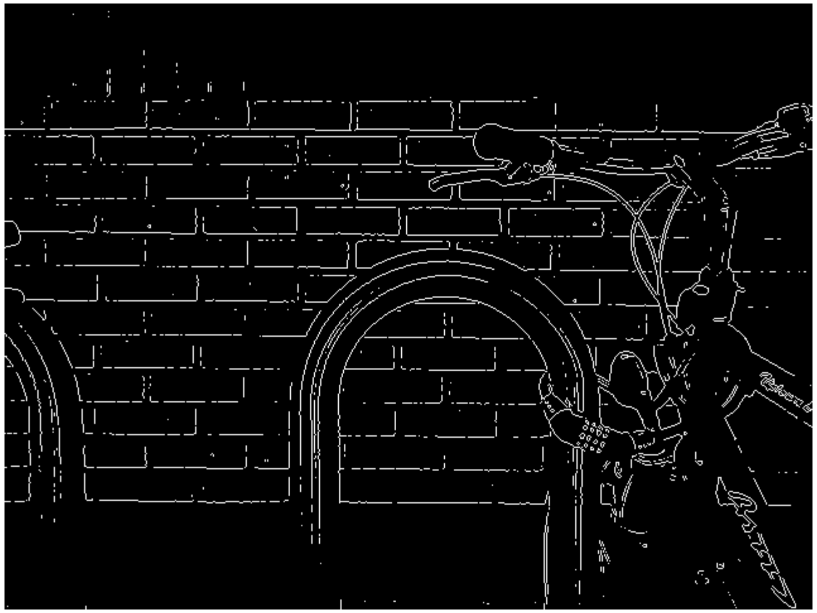
- The Canny edge detector computes edges in images using the following steps:
  - (1) Smooths the image with a Gaussian kernel
  - (2) Computes the gradient magnitude and direction (typically using a Sobel operator)
  - (3) Suppress non-maxima pixels in the direction of the gradient
  - (4) Edge tracking: uses two thresholds to keep either strong edges or weak edges that are connected to strong edges

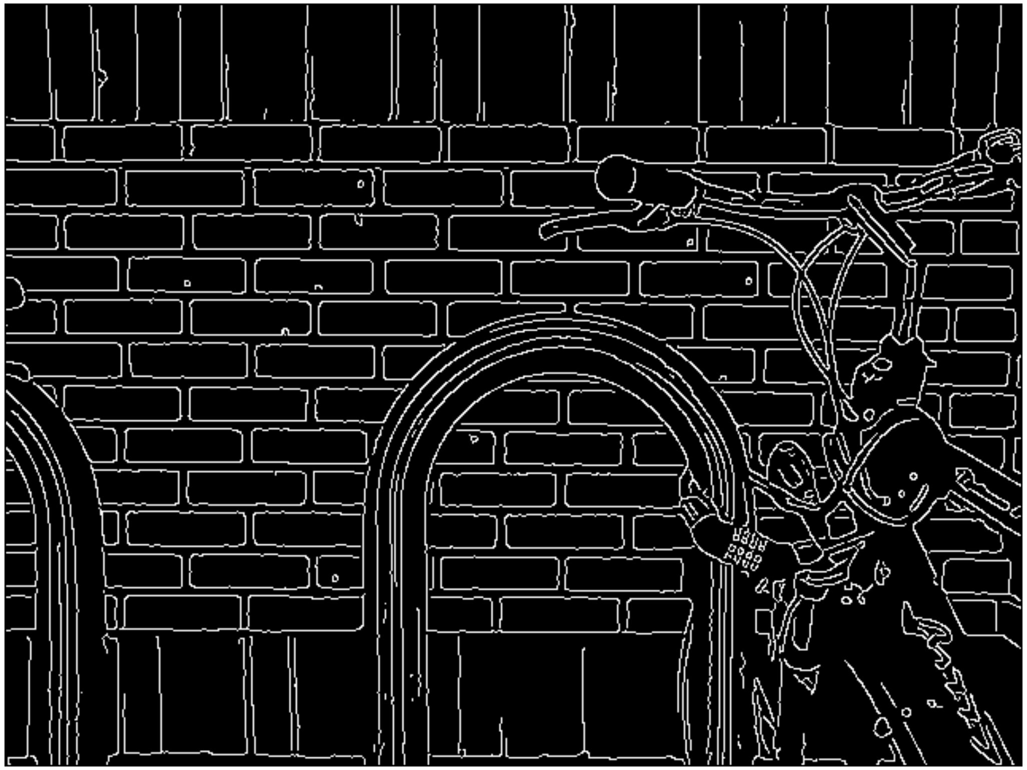



Very strong edge response. Let's start here
Weaker response but it is connected to a confirmed edge point. Let's keep it.
Continue….

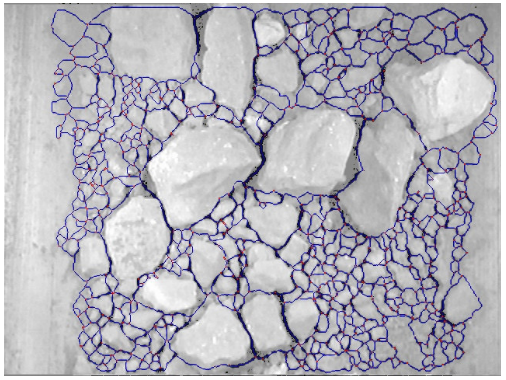
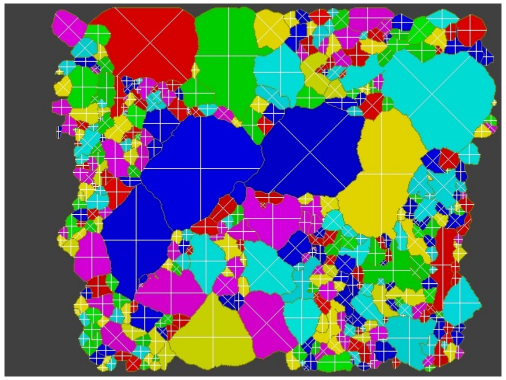
Note: Darker squares illustrate stronger edge response (larger *M*)

The Canny Edge Detector


Sobel Edges


Canny Edges

Applications of Edge Detection and Morphological Operations: Optical Granulometry

- Granulometry is the measurement of the distribution of the size of particles of grains
- Basic image processing techniques can be used to compute the distribution of particle sizes from images:
  - Compute the edges between particle grains
  - Find connected components within the regions in-between edges
  - Measure the size properties of each component and calculate distributions of size

Maerz, N. H., 1998. "Aggregate sizing and shape Determination using digital image processing." Center For Aggregates Research (ICAR) Sixth Annual Symposium Proceedings, pp. 195-203.

Further Reading and Next Week

- References:
  - D. A. Forsyth and J. Ponce, “Computer Vision - A Modern Approach”, Prentice Hall, 2002 (Section 7.1, 8.1)
  - R. Szeliski, “Computer Vision: Algorithms and Applications”, Springer, 2010 (Section 3.2, 4.2)
  - R.C. Gonzalez and R.E. Woods, “Digital Image Processing”, Prentice Hall, 2008 (Section 3.5-3.8, 9)

- Next Week:
  - Features, Detection and Matching