UTSouthwestern
Medical Center
Lyda Hill Department of Bioinformatics

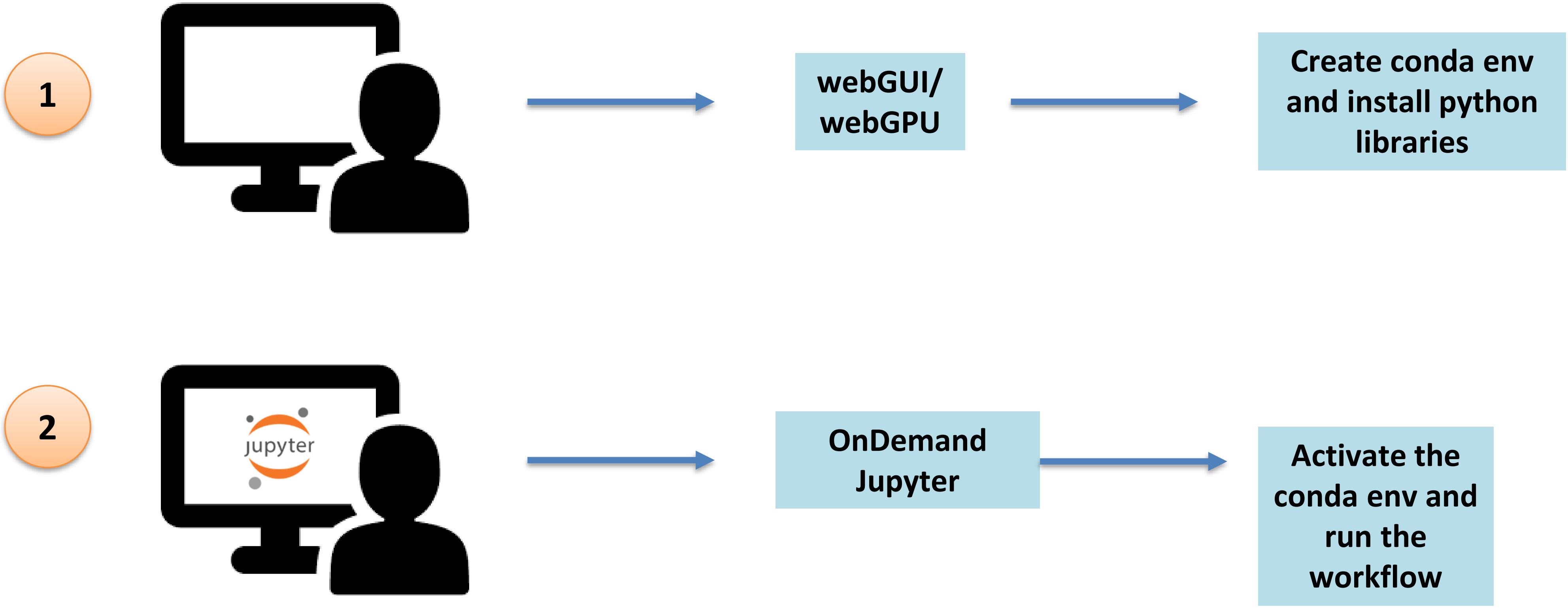
BioHPC

Image processing with Python

[web] portal.biohpc.swmed.edu
[email] biohpc-help@utsouthwestern.edu

Updated for 2022-10-12

A Python image processing workflow on BioHPC



1

webGUI/ webGPU

Create conda env and install python libraries

2

OnDemand Jupyter

Activate the conda env and run the workflow

https://portal.biohpc.swmed.edu/media/filer_public/33/6e/336ee943-7532-42ec-86a1-3931e562f988/2022_09_21_software_installation_on_biohpc.pdf

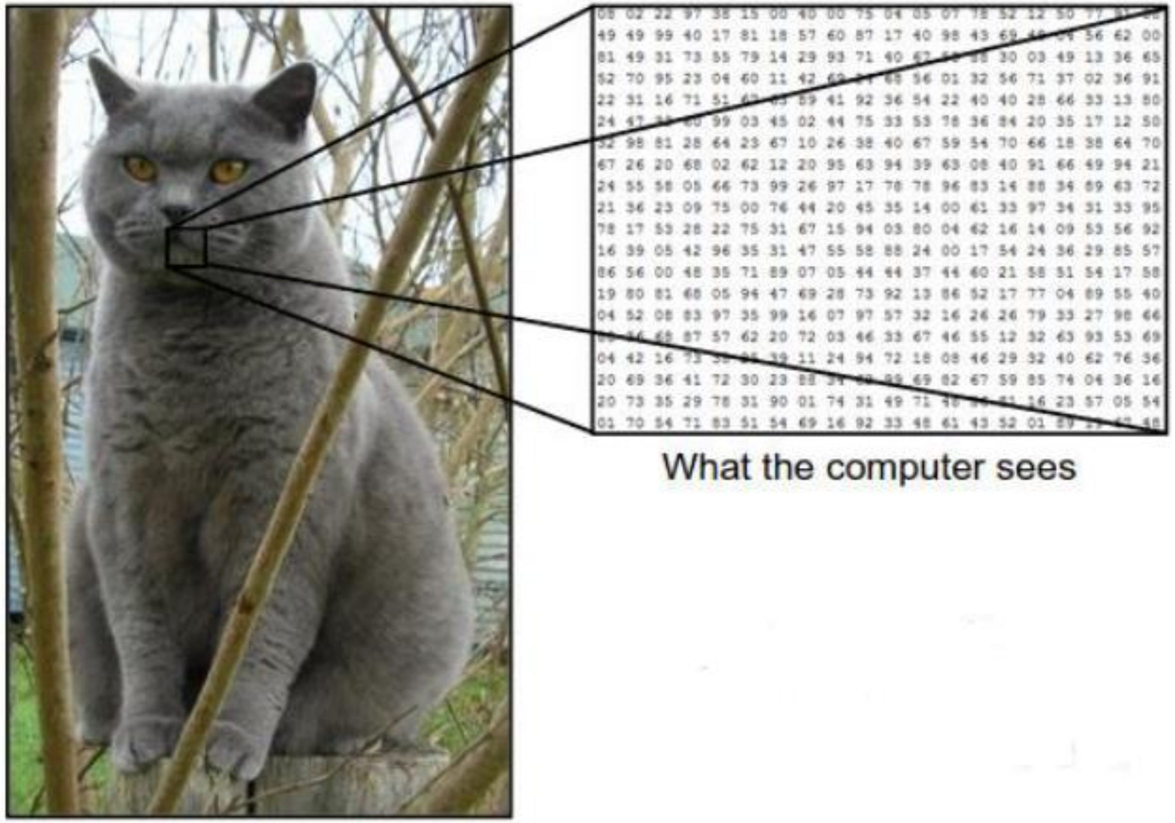
Python Libraries + Modules used in this training

All should already be installed with JupyterLab OnDemand.

Docs:

- os : https://docs.python.org/3/library/os.html
- matplotlib : https://matplotlib.org/
- scipy :
    - General : https://docs.scipy.org
    - ndimage : https://docs.scipy.org/doc/scipy/reference/ndimage.html
- skimage : https://scikit-image.org/
- sklearn : https://scikit-learn.org/stable/
- numpy :
    - General : https://numpy.org/doc/stable/index.html
    - ndarrays : https://numpy.org/doc/stable/reference/arrays.ndarray.html#id1

How a digital image is stored on a computer



```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What the computer sees

Source - http://cs231n.github.io/

Images as Arrays

Different Python libraries have different array implementations

- array
- **numpy**
  - ndarray
- **openCV**
  - cv::Mat

Common data types for image pixels:

- **bool** (binary)- [0,1]
- **int8** (signed integer 8 bit) – numbers in the range: [-128 : 127]
- **float** (double-precision floating point) – Decimal numbers (e.g. 2.2251e-308, 0.4, 0.333333...)
- **uint8** (unsigned 8-bit) – [0,255]
- **uint16** (unsigned 16-bit) – [0,65535]

Python Array Indexing

- Python starts counting indexes from 0, and arranges coordinates like C does (row-major)
- Array elements can be access in two ways:
  - By forward indexing
  - By backward indexing

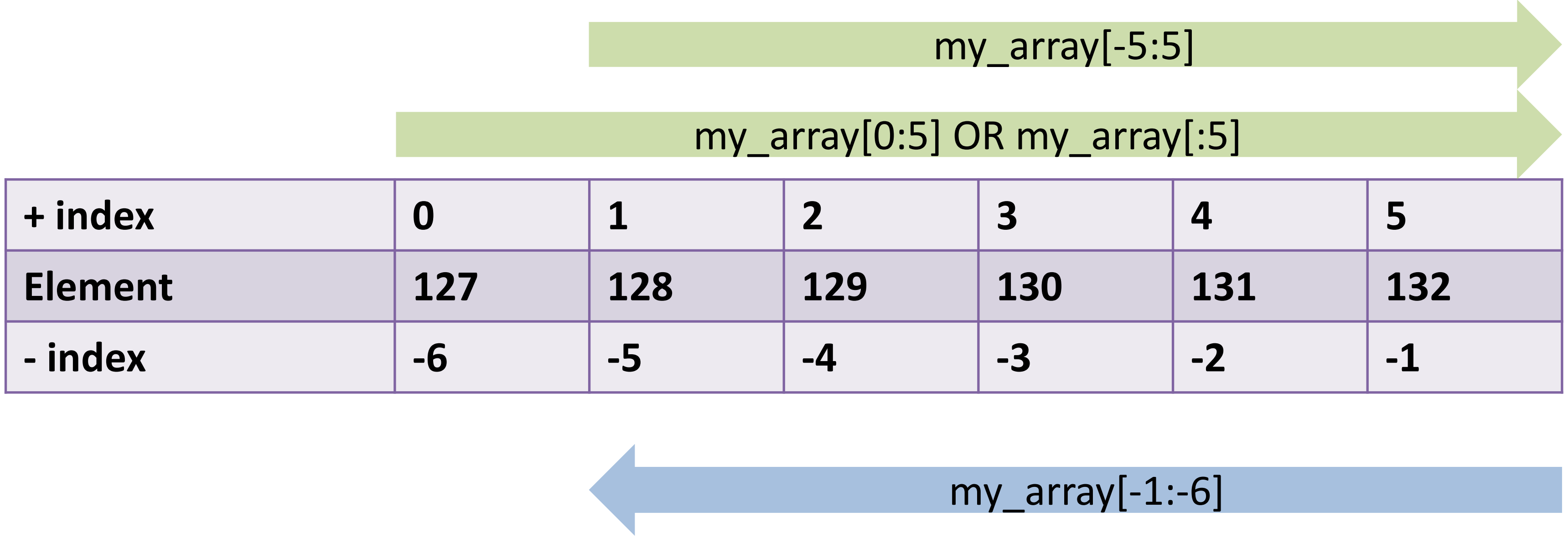
```
my_array = numpy.array([127, 128, 129, 130, 131, 132], dtype=np.int8)
```

+ index	0	1	2	3	4	5
Element	127	128	129	130	131	132
- index	-6	-5	-4	-3	-2	-1

Slice indexes are defined by [START:STOP] or [START:STOP:STRIDE]

Python Array Indexing

```
my_array = numpy.array([127, 128, 129, 130, 131, 132], dtype=np.int8)
```



my_array[-5:5]

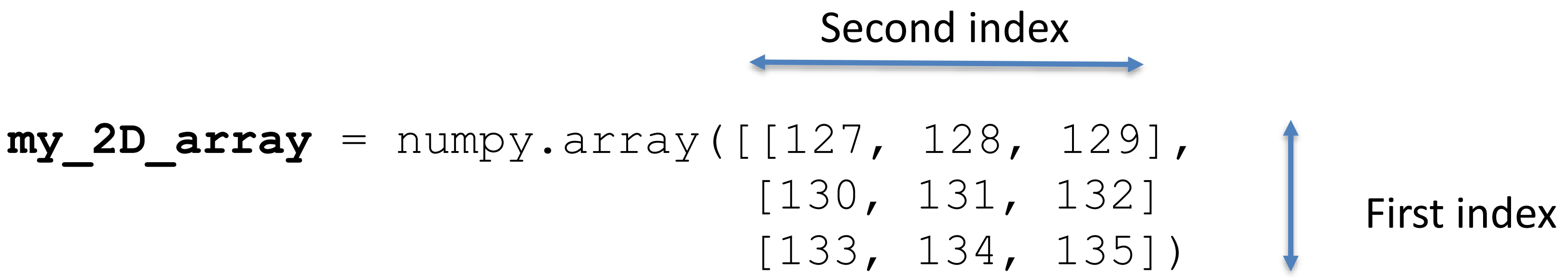
my_array[0:5] OR my_array[:5]

+ index	0	1	2	3	4	5
Element	127	128	129	130	131	132
- index	-6	-5	-4	-3	-2	-1

my_array[-1:-6]

Multi-dimensional arrays – numpy arrays

Python counts in ‘row-major’ ordering, and orders dimensions like C does.
- Multidimensional arrays are ‘lists of lists



Second index

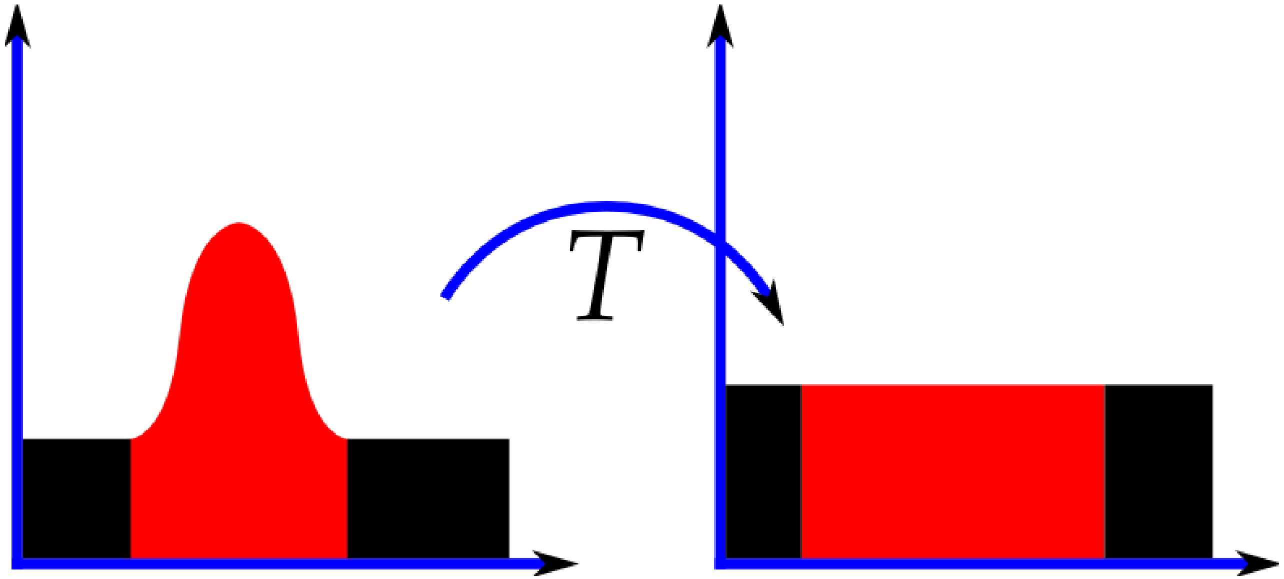
```
my_2D_array = numpy.array([[127, 128, 129],
                           [130, 131, 132]
                           [133, 134, 135])
```

First index

```
my_2D_array[1][0:1] = [133,134]
my_2D_array[-1][:] = [133, 134, 135]
```

Intensity enhancement

- Contrast stretching
- Histogram equalization
- Adaptive equalization

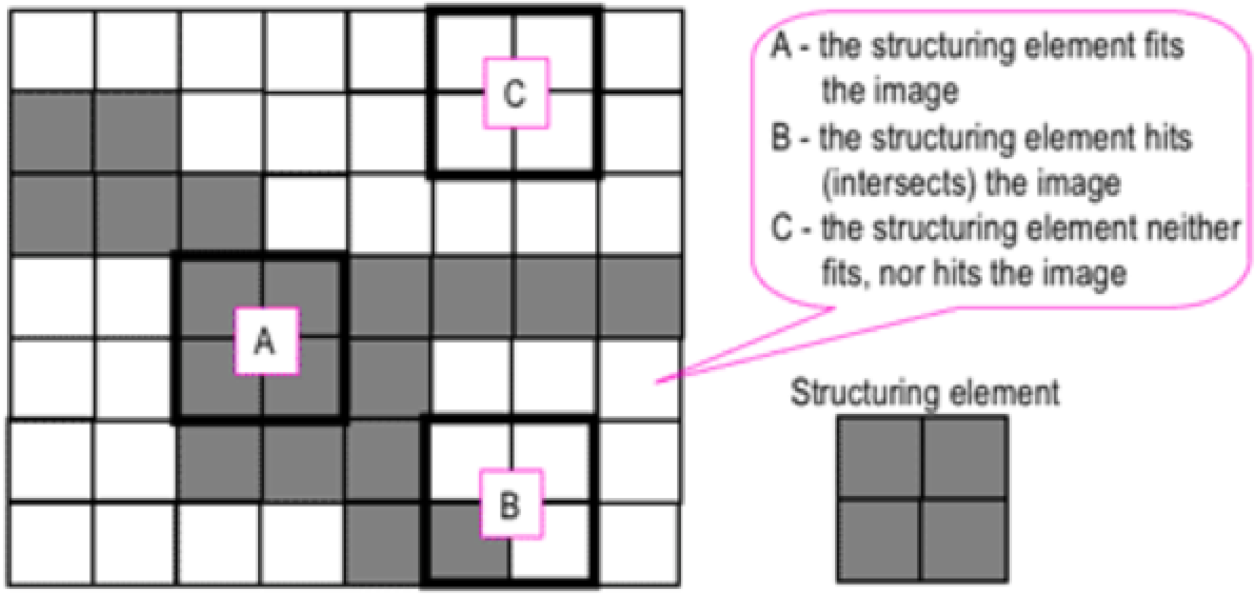


https://en.wikipedia.org/wiki/Histogram_equalization

Morphological operations: Structuring element

The structuring element is a small binary image or matrix such that:

- The matrix dimensions specify the size of the structuring element.
- The pattern of ones and zeros specifies the shape of the structuring element.

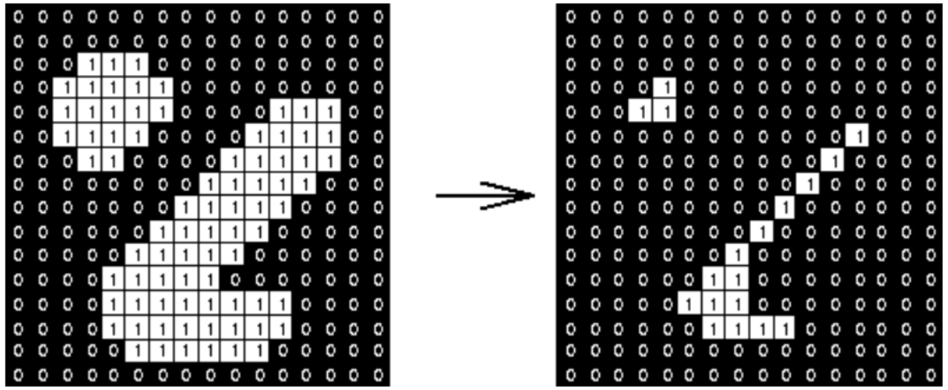

C
A
B
A - the structuring element fits the image
B - the structuring element hits (intersects) the image
C - the structuring element neither fits, nor hits the image
Structuring element

Probing of an image with a structuring element
(white and grey pixels have zero and non-zero values, respectively).

https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

Morphological operations: Dilation and Erosion

Erosion:



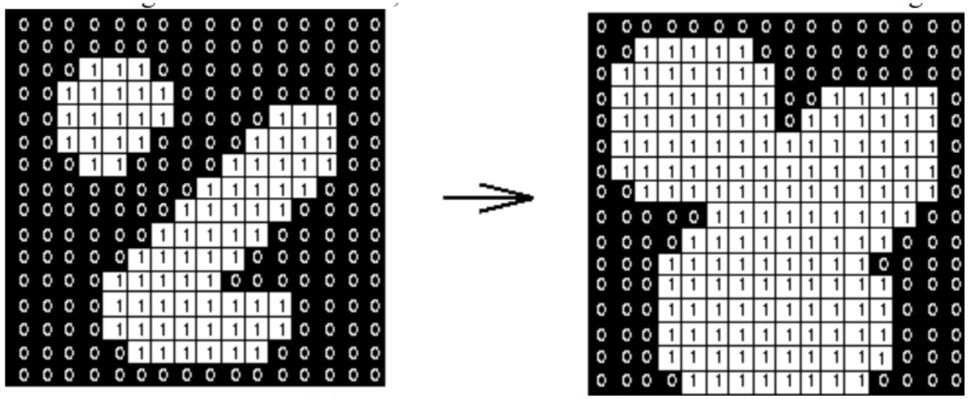
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1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

Erosion: a 3×3 square structuring element
(www.cs.princeton.edu/~pshilane/class/mosaic/).

Dilation:



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

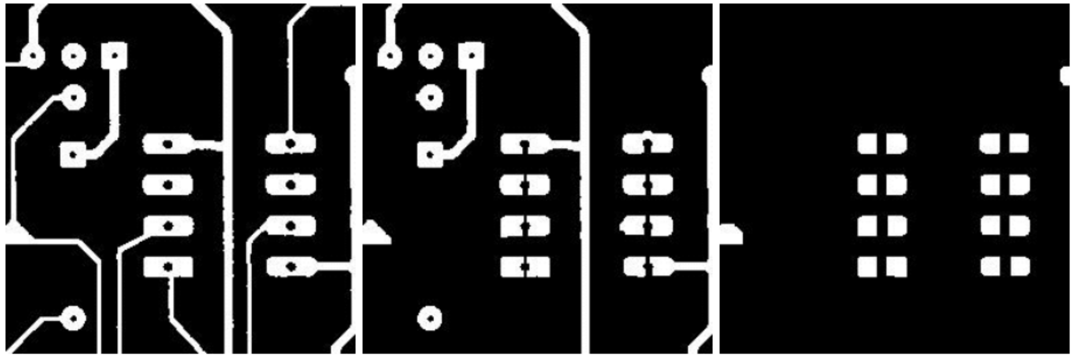
→

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

Dilation: a 3×3 square structuring element
(www.cs.princeton.edu/~pshilane/class/mosaic/).

Morphological operations: Open and Close

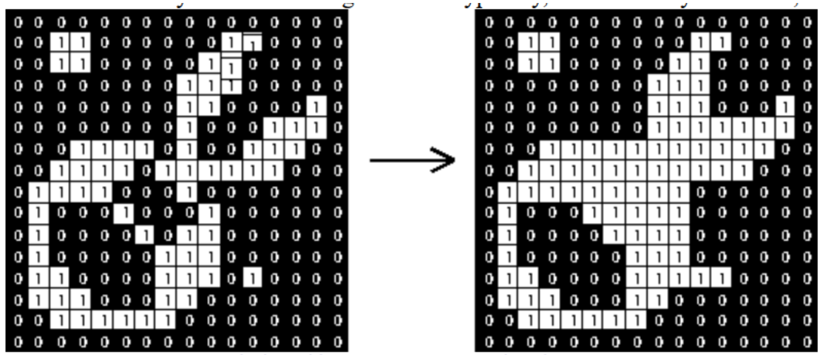
Opening: erosion followed by a dilation



Binary image *f* *f* ∘ *s* (5×5 square) *f* ∘ *s* (9×9 square)

Results of opening with a square structuring element (www.mmorph.com/html/morph/mmopen.html/).

Closing: dilation followed by a erosion



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

Closing with a 3×3 square structuring element
(www.cs.princeton.edu/~pshilane/class/mosaic/).

Mathematical Morphology - Grayscale

- Grayscale images can be treated similarly, but with a slightly modified interpretation of ‘hit or miss’
- Dilation will result in a pixel taking on the max value defined by the moving window of the strel.
- Erosion will result in a pixel taking on the min value defined by the moving window of the strel.

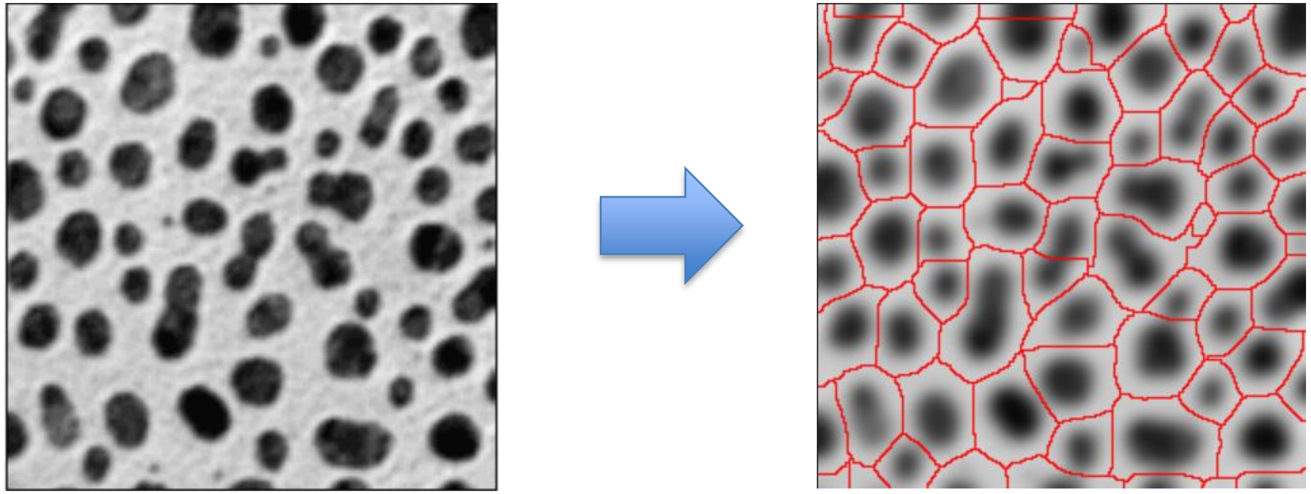
Segmentation – Separating an image into parts

Most basic: Manual thresholding

- Bright or dark background with a dark or bright foreground, respectively.
- Choose a cutoff value, threshold.
- Global thresholds can work, but can miss important elements

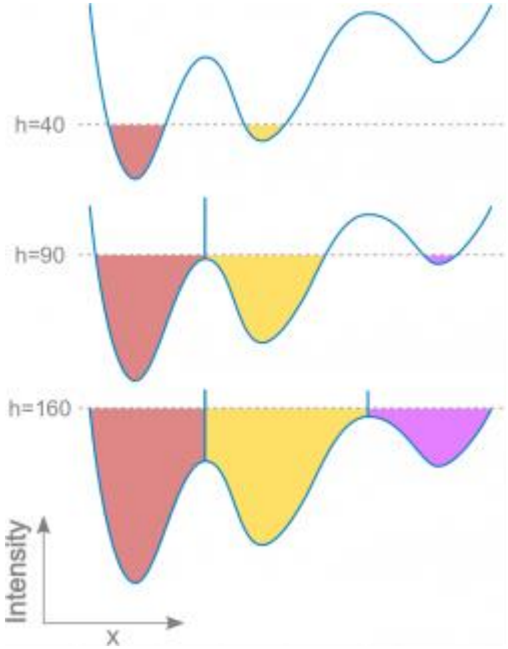
More complex:

- Adaptive thresholds
- Morphological segmentation
- Clustering
- Machine learning methods



Watershed Segmentation

- Consider grey levels as altitudes
- Identify local minima
- Flood basins starting from minima
- Separate the basins by a “dam” → the watershed



Steps for performing the watershed method:

1. Segment objects of interest
2. Convert the mask into an intensity profile using the distance transform
3. Run the watershed algorithm