


浙江大学爱丁堡大学联合学院

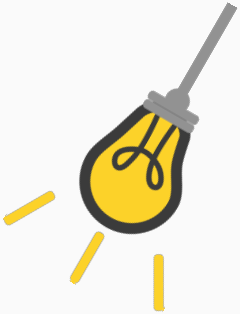
ZJU-UoE Institute

# Lecture 06 - Segmentation

Nicola Romanò - nicola.romano@ed.ac.uk

# Learning objectives

- LO 1
- LO 2
- LO 3



# Introduction

# Segmentation - definition

- Segmentation is a long-studied (and complex!) problem in computer vision.
- Process of dividing an image into sets of pixels called *segments* or *objects*
- Each pixel gets a label identifying which object it belongs to.
- The different segments can then be analysed independently.

# Application of segmentation

In biomedical imaging

- Segmentation of cells to measure their properties
- Analysis of X-ray images to identify pathologies
- Aid in surgery planning

# Application of segmentation

In biomedical imaging

- Segmentation of cells to measure their properties
- Analysis of X-ray images to identify pathologies
- Aid in surgery planning

In computer vision in general

- Car, pedestrian, break lights detection (e.g. for autonomous car navigation)
- Face detection (e.g. for facial recognition, emotion analysis etc.)
- Fingerprint recognition
- …

# Methods for segmentation

Segmentation is a difficult problem to solve, with many important practical applications, so it has been widely studied.

# Methods for segmentation

Segmentation is a difficult problem to solve, with many important practical applications, so it has been widely studied.

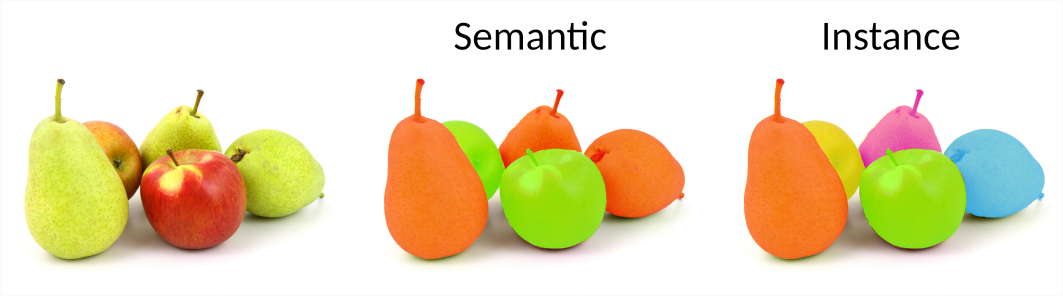
This lecture will cover some of the **traditional methods** for image segmentation. More recent **machine learning-based methods** will be covered later in the course.

# Semantic segmentation vs instance segmentation

There are two main types of segmentation:

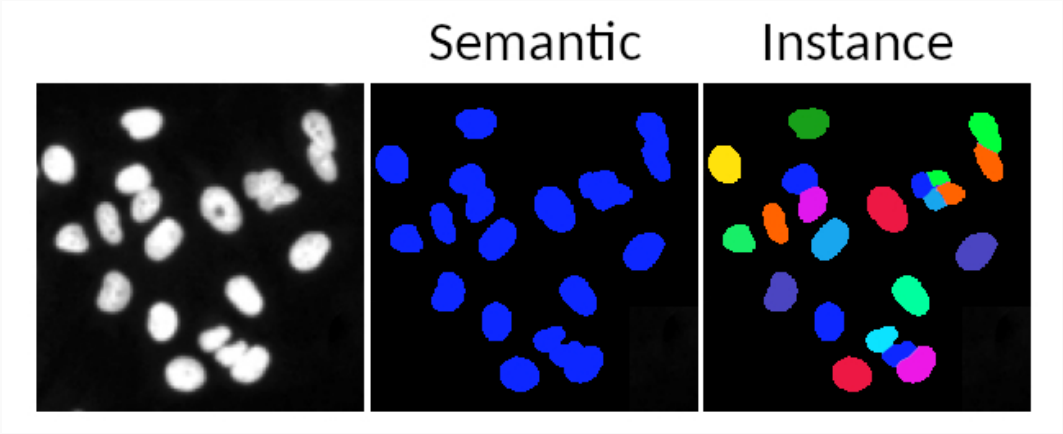
- **Semantic segmentation**: divides the image in regions, each of which belongs to a specific class. Multiple objects of the same class will be detected in the same region.
- **Instance segmentation**: divides the image in regions, each of which is an instance of a class. Multiple objects of the same class will be detected as separate regions.

# Semantic segmentation vs instance segmentation - an example




Source: Apples and pears - Petr Kratochvil - CC0

# Semantic segmentation vs instance segmentation - an example




Source: Nicola Romanò

# Semantic segmentation - thresholding methods

# Thresholding

Thresholding is the simplest way of performing semantic segmentation, when there is a clear distinction between the object(s) of interest and the background.

We define a threshold $t$ and create a mask $M$ such that:

$$M(x, y) = \begin{cases} 0 & \text{for } I(x, y) < t \\ 1 & \text{for } I(x, y) \geq t \end{cases}$$

# Thresholding

Thresholding is the simplest way of performing semantic segmentation, when there is a clear distinction between the object(s) of interest and the background.

We define a threshold $t$ and create a mask $M$ such that:

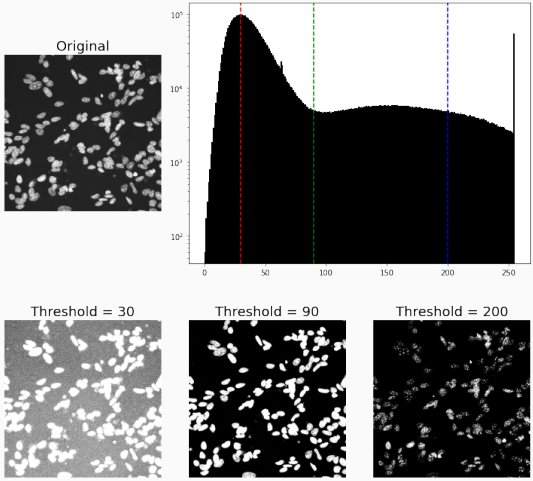
$$M(x,y) = \begin{cases} 0 & \text{for } I(x,y) < t \\ 1 & \text{for } I(x,y) \geq t \end{cases}$$

This can be extended to multi-class segmentation by chosing $t_1, t_2, \ldots, t_n$ and generating a mask

$$M(x,y) = \begin{cases} 0 & \text{for } I(x,y) < t_1 \\ 1 & \text{for } t_1 \leq I(x,y) < t_2 \\ \ldots & \\ n & \text{for } I(x,y) > t_n \end{cases}$$

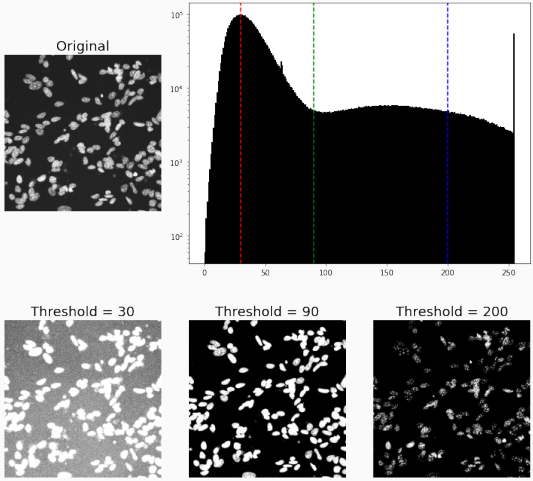
# Choosing a threshold

We can manually choose a threshold by inspecting the image histogram



# Choosing a threshold

We can manually choose a threshold by inspecting the image histogram




**Is there a better way?**

# Otsu's method

Otsu's method is a simple way to automatically choose a threshold.

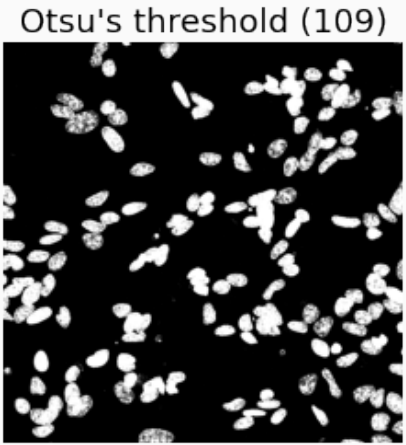
The algorithm makes an exhaustive search for the optimal threshold that maximizes the between-class variance (equivalent to minimizing the intra-class variance).

$$\sigma_\omega^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2$$

$$\omega_0 = \sum_{i=0}^{t-1} p(i) \qquad \omega_1 = \sum_{i=t}^{L-1} p(i)$$

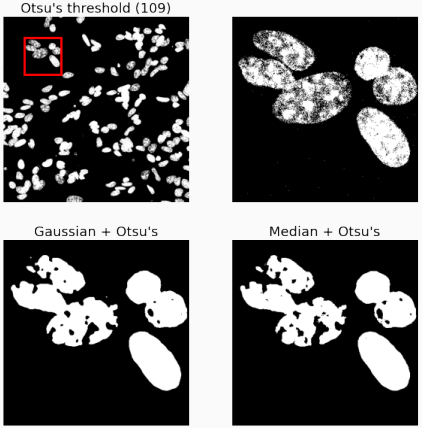
# Otsu’s method - an example

```
from skimage.filters import threshold_otsu

t = threshold_otsu(img)

plt.imshow(img > t, cmap="gray")
plt.axis("off")
plt.title(f"Otsu’s threshold ({t})",
fontsize=18)
plt.show()
```



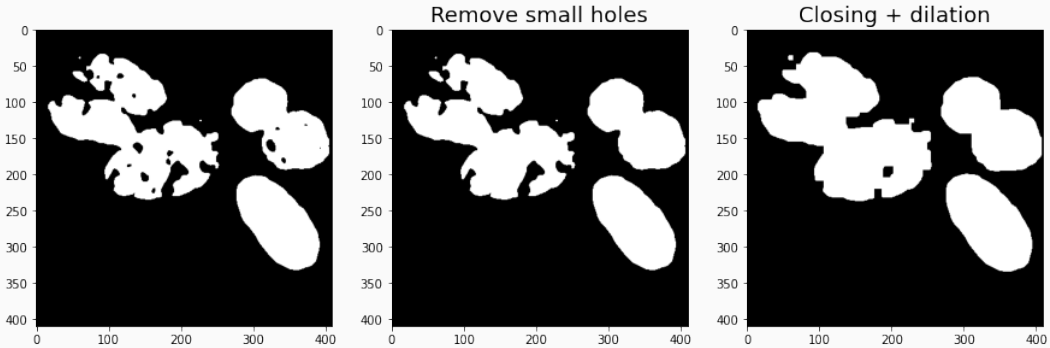
# Issues with thresholding

- Noise is a problem -> Gaussian (or median) filter helps
- Need to remove holes afterwards



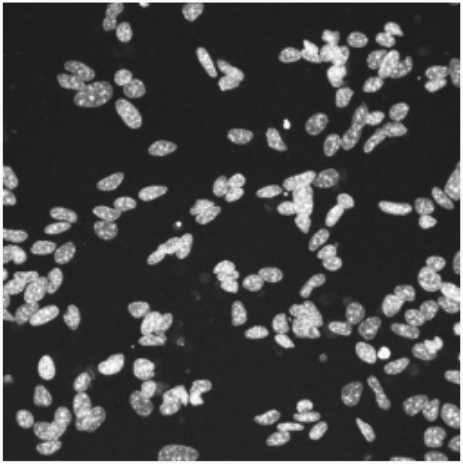
# Filling holes

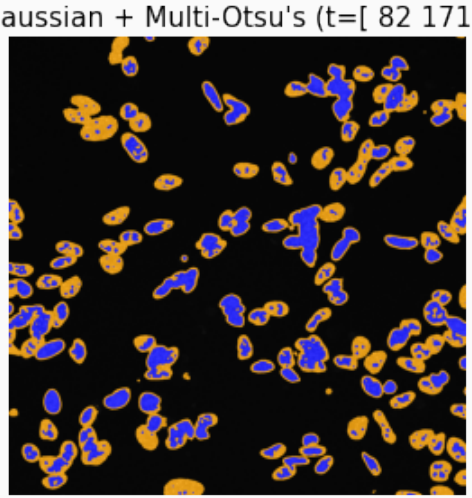
Morphological operations allow to fill small holes. The ‘skimage.morphology.remove_small_holes‘ function is an example of such an operation.



# Multi-Otsu segmentation

The Otsu's method can be extended to multi-class segmentation.





## Multi-Otsu segmentation - code

```
from skimage.filters import threshold_multiotsu, gaussian
from skimage import img_as_ubyte
import numpy as np
from skimage.color import label2rgb

t = threshold_multiotsu(img, classes=3)

# Remember to go back to unsigned 8-bit integers from float!
img_gaus = img_as_ubyte(gaussian(img, 5))
# Convert to mask with 3 levels
img_thr_gaus = np.digitize(img_gaus, t)

# Handy function to map the labels to colors
img_with_overlay = label2rgb(img_thr_gaus, image = img, bg_label=0, colors =
["orange", "blue"], alpha = .8)

# Then visualise using Matplotlib!
```

# Clustering methods

# K-means segmentation

# Voronoi segmentation

# Instance segmentation

# Instance segmentation

## Watershed

# Watershed