

# Image Processing and Computer Vision - Lab 5



Roberta Macaluso  
Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)  
Torino - Italy

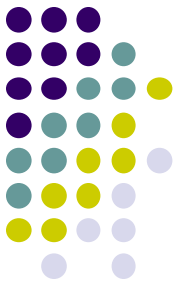
[roberta.macaluso@polito.it](mailto:roberta.macaluso@polito.it)



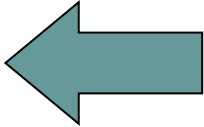
This work is licensed under the Creative Commons (CC BY-NC-SA)

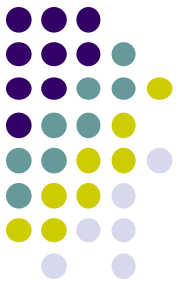


License. To view a copy of this license, visit  
<http://creativecommons.org/licenses/by-nc-sa/4.0/>



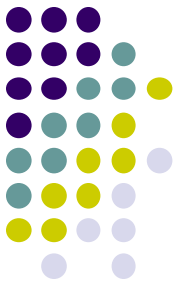
# The Plan

1. Intro to Image Processing
2. CCD, CMOS, and Optical Systems
3. Intro to OpenCV
4. Fourier Transform (and Friends)
5. Image Segmentation 
  - today (07/05)
6. Car Lane Detection
7. Face Detection and Tracking
8. Neural Network Introduction



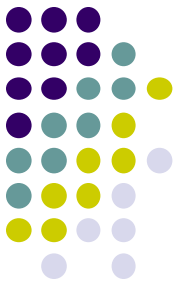
# Image Segmentation

- Today
  - 1,5 hours
- Text of the exercises/tasks
  - on the Teaching Portal
- You need a webcam and a still image
  - the image is on the Teaching Portal
- Goal
  - Experiment with common image segmentation algorithms (like Canny) and the morphological operators



# Image Segmentation

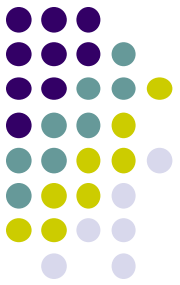
- Two exercises:
  1. Edge detection
  2. Dilation and erosion
- It is a good idea to “*remove some noise*” before applying the image segmentation methods



# Edge Detection: Canny

```
edges = cv2.Canny(image, min_threshold,  
                  max_threshold)
```

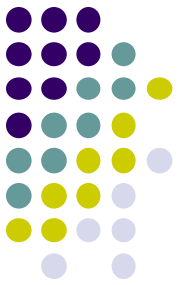
- where `min_threshold` **and** `max_threshold` usually have a 1:3 ratio
- Any edges with intensity gradient more than `max_threshold` **are sure to be edges** and **those below** `min_threshold` **are sure to be non-edges**, so discarded.



# Edge Detection: Sobel

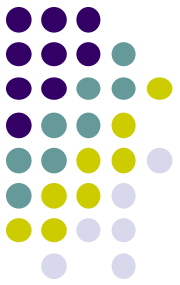
```
x = cv2.Sobel(image, cv2.CV_16S, 1, 0,  
              ksize)  
y = cv2.Sobel(image, cv2.CV_16S, 0, 1,  
              ksize)
```

- you can compute Sobel along x and/or along y



# Edge Detection: Sobel

- After computing Sobel, to show the result of both operations on screen, you need to:
  - `x = cv2.convertScaleAbs(x)`  
[repeat for y]
  - perform a linear blending between `x` and `y`
  - display the result of the previous step



# Morphological Operators

- Dilate

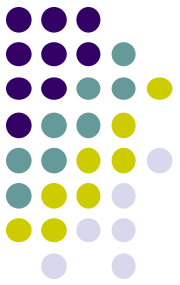
```
ris = cv2.dilate(src, kernel, iterations)
```

- Erode

```
ris = cv2.erode(src, kernel, iterations)
```

- kernel can be a matrix of 1s  
(`np.ones(m, n)`), e.g., 2x2 or 3x3

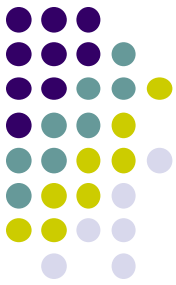




# Image Segmentation

- Hints, insights, links, etc. are in the text of the exercises
  - I am here for you...
  - ... please ask if you need any help or clarification

# License






This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike International (CC BY-NC-SA 4.0)” License.

You are free to:

- **Share** - copy and redistribute the material in any medium or format
- **Adapt** - remix, transform, and build upon the material

for any purpose, even commercially.

Under the following terms:

-  **Attribution** - You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
-  **Noncommercial** - You may not use the material for commercial purposes.
-  **Share Alike** - If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.