Module 7

Edge Detection

- What can edge detection be used for?
  - reducing the amount of pixel data for an algorithm to process while maintaining the structural aspect of the image
  - key features can be extracted from an image
    - features like these are used by higher-level computer vision applications such as object recognition, document analysis, horizon detection, line following robots, etc.
- What is an edge?
  - **a significant local change of intensity levels in an image**
  - edges typically occur on the boundary between two different regions in an image
- What is Canny edge detection?
  - a multistage algorithm
  - it implements noise reduction, finds the edge gradient and direction for each pixel, applies non-maximum suppression for each pixel, and utilizes thresholding with hysteresis
  - edge gradients are computed by the Sobel operator using kernels of size 1, 3, 5, or 7
- What is Sobel edge detection?

- Sobel edge detection is a gradient based method that uses first order derivatives.
- It calculates the first derivatives of the image separately for the x- and y-axis.
- The operator uses two kernels (one for horizontal changes and one for vertical changes) of size 1, 3, 5, or 7 which are convolved with the original image to calculate approximations of its derivatives.

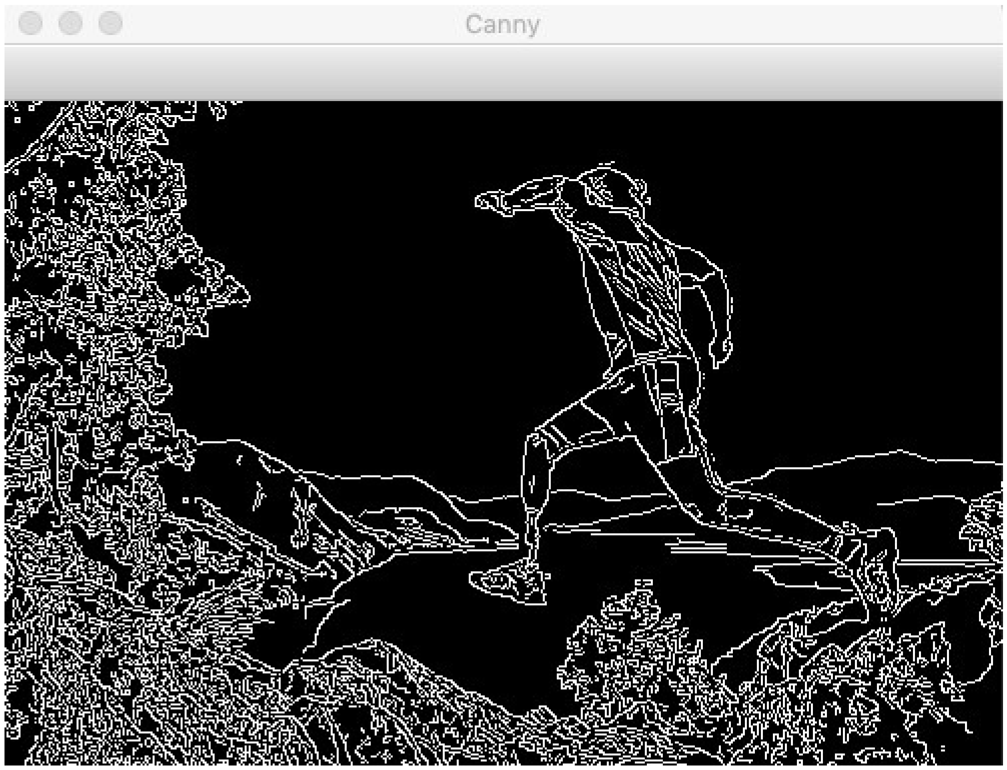
▼ What is Laplacian edge detection?

- Laplacian edge detection is a gradient based method.
- However, unlike the Sobel edge detection, it uses only one kernel.
- The operator uses one 3x3 kernel that is convolved with the original image to calculate approximations of its second order derivatives in a single pass.
- Laplacian is a high-pass filter technique. For application of this filtering technique in edge-detection, the output of the Laplacian operator is subtracted from the original image to produce edge enhancement.

▼ OpenCV example: Canny edge detection

- The following code applies Canny edge detection to the *man_jumping* image using 100 as the minimum threshold for hysteresis and 200 as the maximum threshold.
- The kernel sizes for edge gradient calculations are set to 3 by default.

```
import cv2

img = cv2.imread('man_jumping.jpg')

edges = cv2.Canny(img, 100, 200)

cv2.imshow('Canny', edges)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

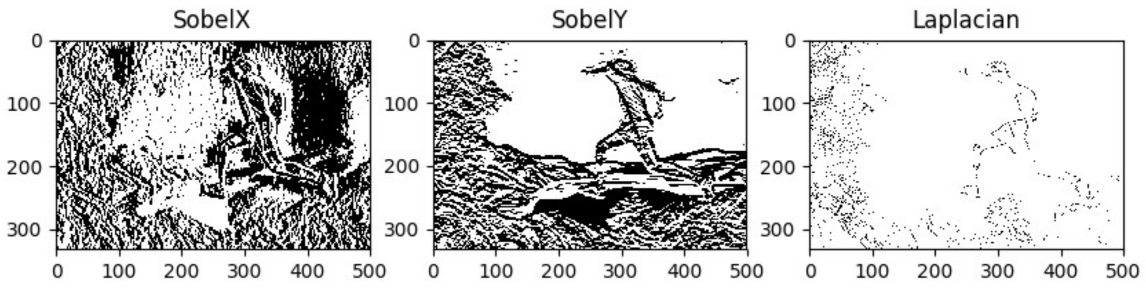


▼ OpenCV example: Sobel and Laplacian edge detection

- Recall that when utilizing these methods, to minimize the detection of “false edges”, Gaussian smoothing should be applied first in order to reduce the noise. Though it is entirely optional, converting the filtered image to grayscale before edge detection may produce better results.
- The following code first applies Gaussian smoothing with a kernel size of 3 to the man jumping image and then converts the filtered image to grayscale.
- Next, Sobel and then Laplacian operators are applied.
- To calculate SobelX (the first derivatives of the image for the x-axis), you set the parameter xorder = 1 and yorder = 0 whereas for SobelY, the parameter values are reversed.
- The Sobel kernels are of size 5 and both output data types are **cv2.CV_32F**.
- The output of the Laplacian operator is subtracted from the grayscale image to produce edge enhancement.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('man_jumping.jpg')

# noise removal with Gaussian smoothing
blur = cv2.GaussianBlur(img, (3,3), 0)

# converting to grayscale is optional but good idea
gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)

# apply sobel and then laplacian
sobelx = cv2.Sobel(gray, cv2.CV_32F, 1, 0, ksize=5)  # kernel for x-axis
sobely = cv2.Sobel(gray, cv2.CV_32F, 0, 1, ksize=5)  # kernel for y-axis

laplacian = cv2.Laplacian(gray, cv2.CV_32F)
laplacian_edge = gray - laplacian

fig, (ax1, ax2, ax3) = plt.subplots(nrows = 1, ncols = 3, figsize = (10,5))

# when showing images in matplotlib, convert image from BGR to RGB
ax1.imshow(cv2.cvtColor(sobelx, cv2.COLOR_BGR2RGB))
ax1.set_title('SobelX')
ax2.imshow(cv2.cvtColor(sobely, cv2.COLOR_BGR2RGB))
ax2.set_title('SobelY')
ax3.imshow(cv2.cvtColor(laplacian_edge, cv2.COLOR_BGR2RGB))
ax3.set_title('Laplacian')
```

```
plt.show()
```


SobelX
SobelY
Laplacian