Correlation and Convolution Filters

- Used to Filter Images
- Blurring Effects
- Remove Noise
- Prepare Images for further analysis
Cross-correlation

Let $F$ be the image, $H$ be the kernel (of size $2k+1 \times 2k+1$), and $G$ be the output image

$$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$
Convolution

- Same as cross-correlation, except that the kernel is “flipped” (horizontally and vertically)

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

This is called a \textit{convolution} operation:

\[ G = H \ast F \]

- Convolution is \textit{commutative} and \textit{associative}
2D Filter Function (Convolution)

cv2.filter2D(img, -1, kernel)

Takes input of grayscale img data filters with kernel. The kernel is an m x m array used to filter the data. Usually a Gaussian, Laplacian, Sorbel, or box filter.

Example:

# img is an existing numpy image array
kernel = np.ones((5, 5), np.float32)/25
box_blur = cv2.filter2D(img, -1, kernel)
Gaussian Blur: Role of Sigma

• We will blur images with the cv2.GaussianBlur() function.

\[
\text{cv2.GaussianBlur(img, (m x m), sigma)}
\]

Where:

- \( \text{img} \) = Image Array to Blur
- (m x m): size of ‘window’ of Gaussian (odd number)
- sigma: The spread of the blur (High for more depth)
CV2 Canny Edge Function

• Built in Edge detection function in Open CV that uses a more involved algorithm:
  • Filters image with derivative of Gaussian
  • Uses magnitude and orientation of gradients
  • Non-Maximum suppression
  • Linking and thresholding

• Result is a more refined edge image.

```python
img_edge_canny = cv2.Canny(img_gauss, 50, 100)
```
2D Gaussian Filter Image
Example:

# Blur with Gaussian
img_gauss = cv2.GaussianBlur(img, (31, 31), 1)

# Blur with Gaussian
img_gauss = cv2.GaussianBlur(img, (31, 31), 10)
Why Blur with Gaussian?

• Remove Noise from the Picture.
Code Sample: Blur with Gaussian

```python
# Gaussian Blur Sample
# Mr. Michaud

import cv2 as cv2
import numpy as np

path = 'images\\panda.jpg'

# Import as a grayscale
img = cv2.imread(path, 0)

# Blur with Gaussian
img_gauss = cv2.GaussianBlur(img, (31, 31), 10)

# Show
cv2.imshow('Gaussian Blur', img_gauss)

# Exit Sequence
cv2.waitKey(0)
cv2.destroyAllWindows()
```
Code Sample: Canny Edge

```python
# Gaussian Blur Sample
# Mr. Michaud

import cv2 as cv2
import numpy as np

path = 'images\panda.jpg'

# Import as a grayscale
img = cv2.imread(path, 0)

# Blur with Gaussian
img_gauss = cv2.GaussianBlur(img, (11, 11), 10)

# Canny Filter
# Convert to uint8 normalized to 0 to 255
img_g8 = img_gauss * 255
img_g8 = np.uint8(img_g8)
img_canny = cv2.Canny(img_g8, 20, 40)

# Show
cv2.imshow('Gaussian Blur', img_gauss)
cv2.imshow('Canny', img_canny)

# Exit Sequence
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Note: You will have to experiment with the parameters for GaussianBlur() and Canny() to get a good edge image.