Comp Photography Assignment #4

Ngoc Tran

Fall 2015

Part 1:Programming Gradients

1) **imageGradientX:** This function takes in a grayscale image, and return the difference of image in the X direction

- Initialize the output array to zero the same size as image passed as argument input array subtract by 1 in column array.
- Iterated through each pixel using two for loop to compute the absolute value of the different using formula F(x, y) = F(x+1, y) - F(x, y)
- Casted the gradientX return value as type float b/c we try to minimize information loss

```
def imageGradientX(image):
    gradientX = np.zeros((image.shape[0], image.shape[1]-1), dtype= np.float)
    for row in range (gradientX.shape[0]):
        for col in range (gradientX.shape[1]):
            gradientX[row,col] = abs( np.float(image[row,col+1]) - image[row,col] )
    gradientX = gradientX.astype (np.float)
    return gradientX
```

Part 1:Programming Gradients

2) **imageGradientY:**This function takes in a grayscale image, and return the difference of image in the Y direction

- Initialize the output array to zero the same size as image passed as argument input array subtract by 1 in row array.
- Iterated through each pixel using two for loop to compute the absolute value of the different using formula F(x, y) = F(x, y+1) - F(x, y)
- Casted the gradientY return value as type float b/c we try to minimize information loss

```
def imageGradientY(image):
    gradientY = np.zeros((image.shape[0]-1, image.shape[1]),dtype= np.float)
    for row in range (gradientY.shape[0]):
        for col in range (gradientY.shape[1]):
            gradientY[row,col] = abs( np.float(image[row+1,col]) - image[row,col] )
    gradientY = gradientY.astype (np.float)
    return gradientY
```

Part 1:Programming Gradients

3) **computeGradient:** This function applies a 3x3 input kernel to the input image. The output is the computed gradient for the image depending on the kernel used.

- Initialize the output array to zero the same size as image passed as argument input array subtract by 2 in both rows and columns of input array.
- Iteration through each pixel using two for loops and use image correlation that take out from neighborhood, and multiplied that with kernel
- then added all the elements of the matrix to a total value and take absolute value

```
def computeGradient(image, kernel):
 result = np.zeros((image.shape[0]-2, image.shape[1]-2), dtype= np.float)
 for row in range( 1, image.shape[0]-2 ):
     for col in range ( 1, image.shape[1]-2 ):
        image_correlation = image[ row-1:row+2, col-1:col+2 ]
         result[row-1, col-1] = abs(np.sum( np.multiply(image_correlation, kernel), axis=(0,1) ))
  result = result.astype (np.float)
  return result
```

Part 2:Edge Detection

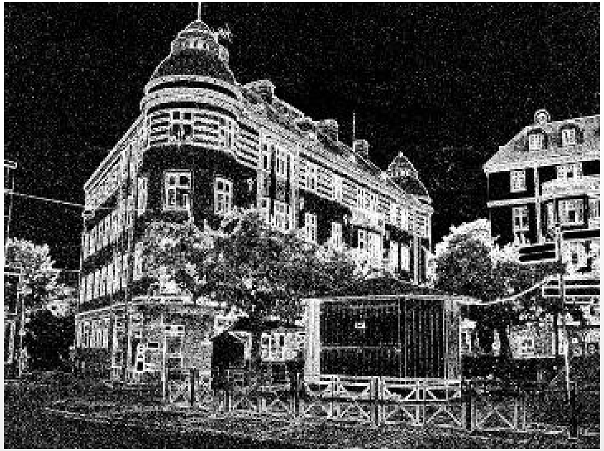
- I use 3 methods (Sobel, Prewitt, and Laplacian) for computing edges using kernels
- Here is a original picture I use for this test. I took from Linkoping, Sweden when i was on a business trip.

Part 2: Sobel Operator

- The Sobel operator is a discrete differential operator. The operator utilizes two 3x3 kernels: one estimates the gradient in the x-direction, and one estimates the gradient in the y-direction.

sobelX- x-direction

sobelY - y-direction

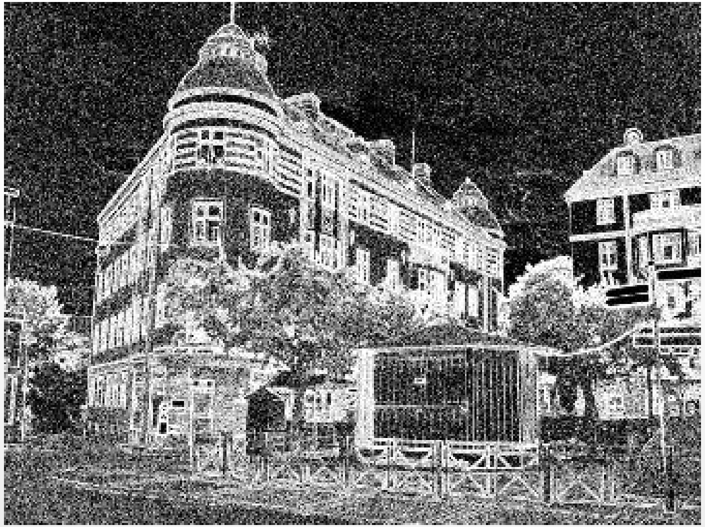


SobelXY- Gradient X and Y direction
& play with threshold value

Part 2: Prewitt Operator

PrewittX- X direction

PrewittY- Y-direction



PrewittXY- Sumed the result of X and Y direction & play with threshold value

Part 2: Laplacian Operator

Unlike the Sobel edge detector, the Laplacian edge detector uses only one kernel. It calculates second order derivatives in a single pass.

Gradients comput use laplacian kernel

Laplacian kernel and play with threshold value

Part 2: Canny Edge

- The advance of Sobel operator is its simple, provides an approximation to the gradient magnitude, and it can detect edges and their orientations, but it is sensitive to noise
- The advance of Canny operator is smoothing effect to remove noise. Good localization and response. Enhances signal to noise ratio. But it's complex to implement.

The original picture