Exercise 2: Convolution & Edge Detection

Due date: 4/8/2019

The purpose of this exercise is to help you understand the concept of the convolution and edge detection by performing simple manipulations on images. This exercise covers:

- Implementing convolution on 1D and 2D signals
- Performing image derivative and image blurring
- Edge detection

1 Convolution

Write two functions that implement convolution of 1 1D discrete signal and 2D discrete signal. The two functions should have the following interfaces:

```
def conv1D(inSignal:np.ndarray,kernel1:np.ndarray)->np.ndarray:
def conv2D(inImage:np.ndarray,kernel2:np.ndarray)->np.ndarray:
```

where: `inSignal`,`kernel1` and `outSignal` are row vectors.
`inImage`,`kernel2` and `outImage` are are 2D arrays.

The result of your convolution functions should be identical to the python functions np.convolve (here is the link) and cv2.filter2D (here is the link) with option 'same'.

2 Image derivatives & blurring

2.1 Derivatives

Write a function that computes the magnitude of an image derivative. You should derive the image in each direction separately (rows and column) using simple convolution with $[1;0;-1]$ and $[1,0,-1]$ to get the two image derivatives. Next, use these derivative images to compute the magnitude image. The function should also display the result. The function should have the following interface:

```
def convDerivative(inImage:np.ndarray) -> np.ndarray:
```

2.2 Blurring

You should write two functions that performs image blurring using convolution between the image f and a gaussian kernel g. The functions should also display the result. The functions should have the following interface:

```
def blurImage1(inImage:np.ndarray,kernelSize:np.ndarray)->np.ndarray:
def blurImage2(inImage:np.ndarray,kernelSize:np.ndarray)->np.ndarray:
```

where:
`inImage` - is the input image to be blurred.
`kernelSize` - is the size of the gaussian in each dimension (squared kernel).
`blurImage` - is the output blurry image.

blurImage1 should be fully implemented by you, using your own implementation of convolution (filter2D) and gaussian kernel. blurImage2 should be implemented by using pythons internal functions: filter2D and getGaussianKernel.

Comments:

- In your implementaion, the gaussian kernel g should contain approximation of the gaussian distribution using the binomial coefficients. A consequent 1D convolutions of [1 1] with itself is an elegant way for deriving a row of the binomial coefficients. Explore how you can get a 2D gaussian approximation using the 1D binomial coefficients. Pay attention that the kernel elements should be summed to 1.
- The border of the images should be padded with zeros.
- The size of the gaussian, $kernelSize$, will always be an odd number.

3 Edge detection

You should implement the following functions:

- `def edgeDetectionSobel(I:np.ndarray)->(np.ndarray,np.ndarray)`
- `def edgeDetectionZeroCrossingSimple(I:np.ndarray)->(np.ndarray,np.ndarray)`
- `def edgeDetectionZeroCrossingLOG(I:np.ndarray)->(np.ndarray,np.ndarray)`
- `def edgeDetectionCanny(I:np.ndarray)->(np.ndarray,np.ndarray)`

I is an intensity image and edgeImage is binary image (zero/one) with ones in the places the function identifies edges. Each function implements edge detections accroding to a different method. edgeImage1 is the edge image of your own implementation and edgeImage2 is the edge image returned by pythons's edge function with appropriate parameters. You can find the description of each of the methods at `https://docs.opencv.org/4.0.0/`

4 Hough Circles

You should implement the Hough circles transform.

```
def houghCircle(I:np.ndarray,minRadius:float,maxRadius:float)->np.ndarray
```

I is the intensity image, min/maxRadius should be positive numbers and minRadius < maxRadius. The functions should return an array of size nX3 (n-number of circles), each line will hold: (x,y,radius).

* This function is costly in run time, be sure to keep the min/maxRadius values close and the images small.

5 Important Comments

- The input of all the above functions will be grayscale images.

6 Submission

Submission instructions will be published in the site.
Good luck and enjoy!