Computer Vision 1: Exercise Sheet 3

Summary:

1. Correlation filtering
2. Convolution filtering
3. Gaussian kernels

1 Correlation filtering

Using NumPy, create the following 3-by-5 matrix as an array:

$$I = \begin{bmatrix} 10 & 10 & 0 & 30 & 10 \\ 10 & 10 & 0 & 30 & 10 \\ 10 & 10 & 0 & 30 & 10 \end{bmatrix}.$$

Also create a 3-by-3 matrix to use as a spatial filter for the moving average:

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Ensure that the datatype of both matrices is `np.float64`.

1.1 Correlation filtering without padding

Import the function to compute a 2-dimensional cross-correlation between signals:

```
1 from scipy.signal import correlate2d
```

Use the function `correlate2d` to calculate the moving average of I. Use the input argument `mode='valid'` to disable signal padding.

- Print out the shape of the result array. Why is it `(1,3)`?
- Verify that the result equals $\begin{bmatrix} \frac{60}{9} & \frac{120}{9} & \frac{120}{9} \end{bmatrix}$.

1.2 Correlation filtering with zero padding

Use the function `np.pad` to pad the array I with two zeros on all sides to create an array

$$I_{pad} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 10 & 0 & 30 & 10 & 0 & 0 \\ 0 & 0 & 10 & 10 & 0 & 30 & 10 & 0 & 0 \\ 0 & 0 & 10 & 10 & 0 & 30 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The input argument `pad_width` should be a tuple `(h,w)` where you have to set the two numbers that specify how many elemts to pad before and after each of the two dimensions, that is, the height and the width dimension.

- Use `correlate2d` with input argument `mode='valid'` to calculate the cross-correlation of I_{pad} and K. Why does the result have shape `(5,7)`?
- Use `correlate2d` and specify the input arguments `mode`, `boundary`, and `fillvalue` so that you obtain the same result for cross-correlation of I and K as you did for I_{pad} and K above.

2 Convolution filtering

Import the function to compute a 2-dimensional convolution:

```
1 from scipy.signal import convolve2d
```

Calculate the convolution of the arrays I and K defined above. Use the function `convolve2d` and set `mode='valid'` to disable padding of the signals.

- Why is the result the same as using `correlate2d`, even though correlation and convolution are different operations?
- Come up with another 3-by-3 kernel H for which `convolve2d` and `correlate2d` produce different results.
- Calculate H_{flip} by flipping H from left-to-right and then from up-to-down using `np.fliplr` and `np.flipud`, respectively. Check that if you correlate I with H_{flip}, the result equals the convolution of I and H.

3 Gaussian kernels

The moving average or box filter is an example of a low-pass filter. The Gaussian filter is usually preferred, however.

- Create a box filter K_{box} of size 5-by-5, that is, a 5-by-5 matrix of ones normalized by multiplying it with $\frac{1}{5 \cdot 5}$.
- Create a 5-by-5 Gaussian filter kernel $K_{gaussian}$ according to the lecture slide example of a 2D Gaussian.
- Load the image `woman.png` from Moodle. Convolve the image with K_{box} and $K_{gaussian}$, respectively.
- Draw the box filtered image, the Gaussian filtered image, and the original image. Can you see differences between the two filtered outputs? Zoom in for example on the eyes or the mouth, and look for sharpness of edges.