Software Training

Task 3

Subtask 1

We have talked about convolution during our session, implement a function that takes an image and a kernel as an input, then performs convolution to obtain an output image with the desired effect. Your function should be able to take a an image and a kernel of any size, just check that the kernel is valid at the beginning – *a kernel needs to have odd rows and columns to be valid.*

Test your code by applying the following filters: *box*, *horizontal sobel* and *vertical sobel*. Bonus points for implementing the *gaussian* and *median* filters.

Use an image of your own, but choose the image wisely – i.e don't use *sobel* filters with an image without easily identifiable edges.

Requirements

Imports

```
# Setup Commands: (inside VSCode terminal)
## (one-time) python -m venv .venv
## (Windows: every re-open) ./.venv/Scripts/activate.bat
## (Other systems: every re-open) ./.venv/Scripts/activate
## (one-time) pip install matplotlib opencv-python numpy
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

convolve(image, kernel)

```
def convolve(image, kernel):

    """
    Apply a convolution to an image using a given kernel.

    Your code should handle different kernel sizes - not necessarily 3x3 kernels
    """

    # Start by flipping the kernel horizontally then vertically (related to the mathematical proof of convolution)
    # Pad the image
    # Perform convolution using sliding window method, dot product is not used here
    ... # remove comments and implement functionality
```

TURN OVER THE PAGE

Testing Code

```
# Take notice that OpenCV handles the image as a numpy array when opening it
img = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
fig, axes = plt.subplots(2, 2, figsize=(8, 8))

axes[0, 0].imshow(img, cmap='gray')
axes[0, 0].set_title('Original Image')
axes[0, 0].axis('off')

axes[0, 1].imshow(convolve(img, np.ones((5, 5)) / 25), cmap='gray')
axes[0, 1].set_title('Box Filter')
axes[0, 1].axis('off')

axes[1, 0].imshow(convolve(img, np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])),
    cmap='gray')
axes[1, 0].set_title('Horizontal Sobel Filter')
axes[1, 0].axis('off')

axes[1, 1].imshow(convolve(img, np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])),
    cmap='gray')
axes[1, 1].set_title('Vertical Sobel Filter')
axes[1, 1].axis('off')
plt.show()
```

Subtask 2

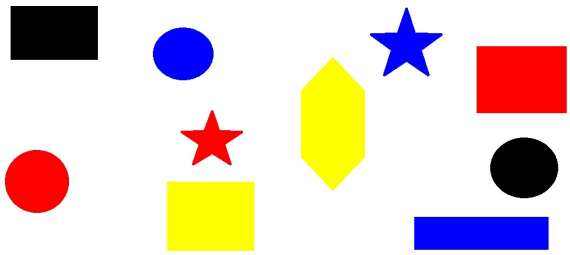
M.K was having some trouble editing a photo, he wanted to change the blue parts to black, the red parts to blue and the black parts to red. He went to ask Samy for advice, but Samy being Samy (3ammy w 3am 3eyali) said: “Fakess photoshop, el3elm nour. Gah elwa2t ely asta5dem fih el7agat ely et3alemtaha fi el CV”.

After saying that, Samy realised that he is very busy with preparing the upcoming sessions and doesn’t have enough time, but he remembered that thresholding in numpy is mentioned in one of the videos that were sent before, so he is asking you to do it instead.

Requirements

Imports & Preamble

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
# Take notice that OpenCV handles the image as a numpy array when opening it
img = cv2.imread('shapes.jpg')
out = img.copy()
```



Required & Testing Code

```python
# Make a mask for each color (red, blue, black)
# Take care that the default colorspace that OpenCV opens an image in is BGR not RGB

# Change all pixels that fit within the blue mask to black
# Change all pixels that fit within the red mask to blue
# Change all pixels that fit within the black mask to red

fig, axes = plt.subplots(1, 2)
axes[0].imshow(img)
axes[0].set_title('Original Image')
axes[0].axis('off')

axes[1].imshow(out)
axes[1].set_title('Processed Image')
axes[1].axis('off')

plt.show()
```

Submission Guidelines

```
AUR-Training-25 (repo name)
└─ Phase 2
   └─ Session 3
      ├─ Subtask 1.py
      └─ Subtask 2.py
```

END OF DOCUMENT