Computer Vision: Homework 1

CV2022@CSU

1 Instructions

Python Environment We are using Python for this homework. You can find references for the Python standard library here: https://docs.python.org. To make your life easier, we recommend you to install Anaconda for Python (https://www.anaconda.com/download/). This is a Python package manager that includes most of the modules you need. We will make use of the following package: Numpy, SciPy, Matplotlib.

2 Image Patches

A patch is a small piece of an image. Sometimes we will focus on the patches of an image instead of operating on the entire image itself.

Task 1: Complete the function `image_patches` in `filters.py`. This should divide a grayscale image into a set of non-overlapping 16×16 pixel image patches. Normalize each patch to have zero mean and unit variance. Plot and put in your report three 16×16 image patches from `grace_hopper.png` loaded in grayscale.

3 Image Filtering

There's a difference between convolution and cross-correlation: in cross-correlation, you compute the dot product (i.e., `np.sum(F * I[y1:y2, x1:x2])`) between the kernel/filter and each window/patch in the image; in convolution, you compute the dot product between the flipped kernel/filter and each window/patch in the image.

Task 2:

- **Convolution and Gaussian Filter**. Complete the function `convolve()` in `filters.py`. Be sure to implement convolution and not cross-correlation/filtering (i.e., flip the kernel as soon as you get it). For consistency purposes, please use zero-padding when implementing convolution. Plot the following output and put it in your report. Load the image `grace_hopper.png` as the input and apply a Gaussian filter that is 3×3 with a standard deviation of $\sigma = 0.572$.

- **Edge detection**. When working on edge detection, we often pay a lot of attention to the derivatives. Denote the "derivatives":

$$I_x(x, y) = I(x + 1, y) - I(x - 1, y) \approx 2\frac{\partial I}{\partial x}(x, y)$$
$$I_y(x, y) = I(x, y + 1) - I(x, y - 1) \approx 2\frac{\partial I}{\partial y}(x, y)$$

  where I_x is the twice the derivative and thus off by a factor of 2. This scaling factor is not a concern since the units of the image are made up and anyway scale the derivative and so long as you are consistent, things are fine. Derive the convolution kernels for derivatives: (i) $k_x \in \mathbb{R}^{1\times3} : I_x = I * k_x$; (ii) $k_y \in \mathbb{R}^{3\times1} : I_y = I * k_y$.

  Follow the detailed instructions in `filters.py` and complete the function `edge_detection()` in `filters.py`, whose output is the gradient magnitude. Use the original image and the Gaussian-filtered image as inputs respectively and use `edge_detection()` to get their gradient magnitudes. Plot both outputs and put them in your report.

Task 3: The Sobel filters S_x and S_y are given below and are related to a particular Gaussian kernel G_S :

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad G_S = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Complete the function `sobel_operator()` in `filters.py` with the kernels/filters given above. Plot the following and put them in your report: $I * S_x, I * S_y$, and the gradient magnitude with the image '`grace_hopper.png`' as the input image I.

4 Submission

将所有的代码、实验图像和结果报告打包成zip文件(班级-学号-姓名-HW1.zip)通过学校可视化教学平台提交，代码中所有文件路径请使用相对路径(切勿使用绝对路径)