CSC 249/449 Machine Vision: Homework 1

Term: Spring 2020
Instructor: Prof. Chenliang Xu
TA: Jing Shi, Zhiheng Li, Haitian Zheng, Yihang Xu, Sizhe Li, Xiaoning Gao, Tianyou Xiao, Weitao Tan
Due Date: Feb 4, 2020 11:59pm

Constraints: This assignment is to be carried out independently.

Problem 1 (30 pts): Get hands on Toolbox
Here are some simple coding questions to help you be familiar with some libraries (e.g., OpenCV, NumPy) used in computer vision (read *INSTALLATION.pdf* to setup the environment):

- Convert the color image to grayscale image with NumPy's API by the following formula:

$$GRAY = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{1}$$

  Do **NOT** use for-loop in your implementation.

- Smooth the image with box filter with cv2.filter2D (a OpenCV's implementation of correlation). Box filter is all-one filter normalized by size of filter. Here, we choose 3×3 as the size of box filter K_b:

$$K_b = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

  Then, we can obtain the smoothed image $\overline{f}(x, y)$ by:

$$\overline{f}(x, y) = K_b \star f(x, y), \tag{3}$$

  where $\star$ means convolution and $f(x, y)$ is the original image.

- Sharpen the image with laplacian with cv2.filter2D by the following equation:

$$g(x, y) = f(x, y) - f_l(x, y), \tag{4}$$

  where $g(x, y)$ is the sharpened image, $f(x, y)$ is the original image, $f_l(x, y)$ is image's laplacian. Here, the laplacian filter K_l is defined by:

$$K_l = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5}$$

- Sharpen the image with unsharp masking, defined by the following equations:

$$f_{\text{mask}}(x, y) = f(x, y) - \overline{f}(x, y), \tag{6}$$
$$g(x, y) = f(x, y) + k f_{\text{mask}}(x, y), \tag{7}$$

  First, we compute the mask image $f_{\text{mask}}(x, y)$ by subtracting the smoothed image $\overline{f}(x, y)$ from the original one $f(x, y)$. Then, the sharpened image can be obtained by adding the weighted mask image to the original one. Use $k = 1$ in your implementation.

- Detect edge of by Sobel filter with cv2.filter2D. Here, the horizontal derivative filter and vertical derivative filter are defined by

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{8}$$

  The edges can be obtained by:

$$E = \sqrt{(K_x * I)^2 + (K_y * I)^2}, \tag{9}$$

  where $*$ means cross correlation.

Requirements:

- Complete the code in *code/problem1/cv249.py* marked by **TODO**.
- Run *code/problem1/test_cv249.py* to test your code.
- Run *code/problem1/gen_imgs.py* to generate images by your implementation. You will see the output images in *data/result_imgs*.

You can read the documentation of OpenCV (https://docs.opencv.org/2.4/modules/refman.html) and NumPy (https://docs.scipy.org/doc/numpy-1.15.1/reference/) for details of API.

Problem 2 (40 pts): Implementing forward functions of 2D convolution, 2D max pooling, and 2D average pooling

You need to implement forward pass of 2D convolution, 2D max pooling, and 2D average pooling in *code/problem2/layer.py*. Complete the code marked by **TODO**.

You can refer to this note (http://cs231n.github.io/convolutional-networks/) to know more details about convolution, max pooling, and average pooling.

Correction of output size of Convolution/Pooling layers mentioned in cs231n:

$$W_2 = \lfloor (W_1 - F + 2P)/S \rfloor + 1,$$

$$H_2 = \lfloor (H_1 - F + 2P)/S \rfloor + 1, \tag{10}$$

where the difference is the floor operation which is used to avoid floating number output size.

Do **NOT** modify function interfaces. If you want to add parameters to a function, please provide default values so that the original behavior of the function is unchanged.

Do **NOT** use any additional python packages/modules other than those provided in the framework.

Do **NOT** use NumPy's convolution API.

1. (30 pts) Convolutional Layer

   The convolutional layer you will be implementing in this assignment will work for images (or feature maps) with channels. That is, the input image will be 3D of shape (C, H, W) where C denotes the number of channels. The convolutional layer should support stride and padding and have multiple neurons, i.e. output multiple activation maps. In convolutional neural network, a convolutional layer is actually implemented in correlation. In order to keep the consistency with convolutional neural network, please use correlation in your implementation as well.

   Note for this assignment, we are seeking a direct implementation of convolution with nested for loops (that is how you really understand the convolution).

   - Implement the *forward* pass for the Conv2D layer in the "layer" module.
   - Test your implementation by running *python test_layer.py TestLayer.test_conv*

2. (40 pts) Max Pooling Layer and Average Pooling Layer

   The pooling layers you will be implementing in this assignment will work for images (or feature maps) with channels. That is, the input will be 3D of shape (C, H, W) where C denotes the number of channels. The pooling layer should support stride and padding.

   - Implement the *forward* pass for the MaxPooling layer and AvgPooling layer in the "layer" module.
   - Test your implementation by running *python test_layer.py TestLayer.test_max_pool TestLayer.test_avg_pool*

Problem 3 (20 pts): Separable Filter

1. Separate each of the following filters into the product of two one-dimensional filters:

- Box filter: $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- Sobel filter (vertical derivative): $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

2. Assuming that the computational complexity of filter $\mathcal{F} \in \mathbb{R}^{k_1 \times k_2}$ convoling with image $\mathcal{I} \in \mathbb{R}^{H \times W}$ is $\mathcal{O}(k_1 k_2 HW)$. What is the computational complexity of convolution of a **separable** filter $\mathcal{F}_s \in \mathbb{R}^{k \times k}$ and image $\mathcal{I} \in \mathbb{R}^{H \times W}$?

Problem 4 (10 pts): Edges

Compared with Laplacian of Gaussian, why Laplacian filter is not a good edge detector?

Problem 5 (20 pts): 3D Corner (*CSC 449 ONLY and extra bonus for CSC 249*)

The mathematics of corner detection we discussed at length in lecture analyze the eigenvalues of the 2D gradient structure tensor. Let λ_1 and λ_2 denote the larger and smaller eigenvalues, respectively. We select feature points based on the analysis of the eigenvalues (e.g., two small eigenvalues indicate a mostly absent gradient, one large and one small eigenvalue indicate an edge, and two large eigenvalues indicate a corner).

Now, consider a video parameterized over $(x, y, t) \in \mathbb{R}^3$. The eigenvalues of the 3D gradient structure tensor will similarly stratify the pixels in the video into different types. Let $\{\lambda_1, \lambda_2, \lambda_3\}$ denote the three eigenvalues of this 3D structure tensor (sorted in decreasing order).

Complete the following questions.

1. (10) First derive the form of the 3D structure tensor from the sum of squared differences (SSD) error function (for window W):

$$E(u, v, w) = \sum_{x,y,t \in W} [\mathbf{I}(x + u, y + v, t + w) - \mathbf{I}(x, y, t)]^2 \ .$$

   Then propose a criterion to extract "3D corners."
   Hints: recall that we used first-order Taylor approximation for small motions in the 2D case.

2. (10) Describe the different types of 3D structures by analyzing how the relation between the three eigenvalues can vary. And, answer the specific question: "What is a 3D corner?" Recall that you are answering these questions for the case of a video rather than a 3D image, such as an MRI scan. Although geometrically these may be the same, consider the special cases that will be observed by objects moving in video as your answer.

Submission Process: You should prepare your submission using the *collect_submission.py* program. Please run the program and upload the generated zip file to Blackboard. Submissions without following the format will receive penalty points. Your submission should contain the following files:

- code/ - The implementation of the framework we provide you.
- homework.pdf - Your answers to Problem 3, 4, and 5. Word or LaTeXis recommended for writing your homework. For scanned file, please make sure your handwriting is legible.

Grading and Evaluation: The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given when appropriate.