Project 2

Out: Oct. 09, 2023
Due: Nov. 06, 2023 (deadline: 11:55PM)
Please note that this Project extends over 4 weeks since the week of the mid-term exam is included.

Late submissions: Late submissions result in 10% deduction for each day. The assignment will no longer be accepted 3 days after the deadline.
Grading: 25% for implementations and 75% for experiments and written report including graphs, results, and critical discussion.

Office hours:

		Mon	Tue	Wed	Thur	Fri
Guido Gerig	gerig@nyu.edu				2-3pm ZOOM	
Pragnavi Ravuluri Sai	pr2370@nyu.edu	4-6pm				
Sai Rajeev Koppuravuri	rk4305@nyu.edu			4-5pm		

Please remember that we also use campuswire for communication on homeworks.

Please read the instructions carefully. Note: we will not be running code. Rather, we will check your code to make sure your implementation is your own, and it matches your results. Your grade is primarily based on your written report. This means going beyond just showing results, but also describing them. You should produce a standalone lab report, describing results in enough detail for someone else (outside of class) to follow. Please read every page in this assignment. **Please submit a single PDF/HTML with all code included as an appendix.**

1) Convolution and Derivative Filters

Correlation/Convolution: Implement 2D convolution (denoted *) **without the use of built in functions**. Your function will take as input two 2D arrays, a filter *f* and an image *I,* and return *f*∗*I*. You can handle the boundary of *I* in any reasonable way of your choice, such as padding with zeros based on the size of the filter *f*. You are free to make use of any part of your previous indexing implementation.
Make the convolution module flexible to receive 1D line filters (used for separable filtering) and also masks representing image templates (used in the second question below).

Please note that all processing is **done in floating point** and not on byte images. As a first step, convert input images into float arrays, and also use float arrays for all the processing steps. Would you not have a display of floats, you can at the very end convert the results back to byte or integers.

```
def convolution(f, I):
      # Handle boundary of I, e.g. pad I according to size of f

      # Compute im_conv = f*I
      return im_conv
```

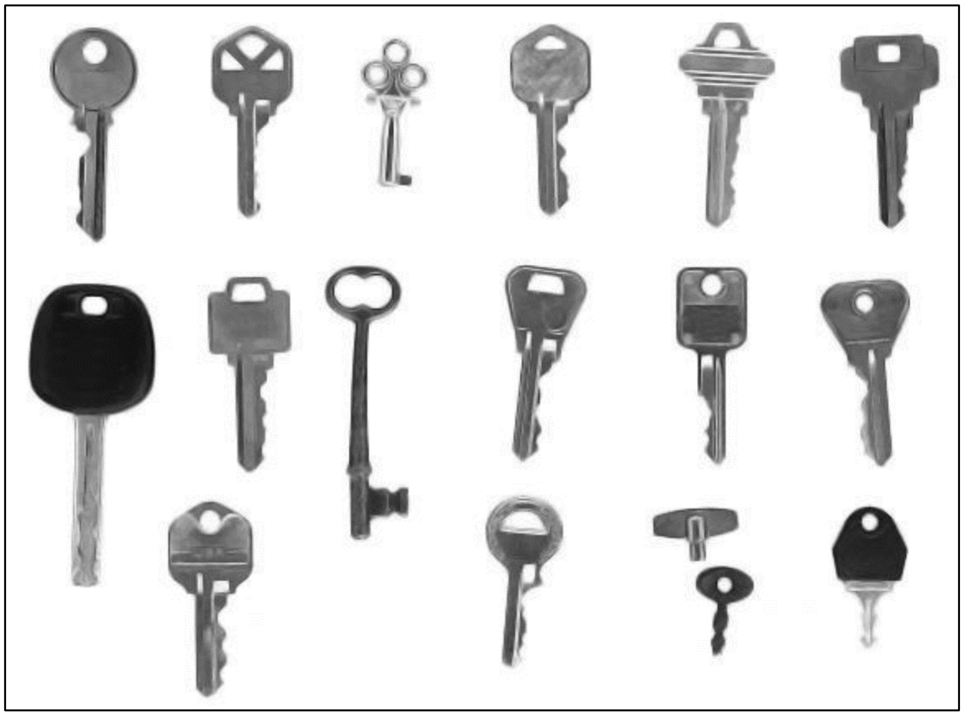
Edge Detection Filtering: Using the example image ‘cameraman.png or zebra.png.

- Denoise the image with a Gaussian filter. Please remember the in class demo and pdf description (uploaded to Brightspace) to create a 1D Gaussian filter, but <u>you must use your own implementation of convolution</u>. Use separability to consecutively filter first with a horizontal 1D Gaussian Gx and second a vertical 1D Gaussian Gy.
- Compute derivative images (with respect to x and with respect to y) using the separable derivative filter of your choice, e.g. [-1 0 1] and [-1 0 1]^T. <u>You can hardcode the derivative filter, but use your implementation of convolution</u>.
- Compute the gradient magnitude image.

- Create binary edge images from the gradient magnitude image using a best threshold.

In your report, introduce how the derivative filter is a discrete approximation of a derivative. Show all results and discuss the outcome of various thresholds.

2) Cross-correlation and Template Matching



Template

The uploaded image '*multiplekeys.png*' is a single image containing multiple instances of keys. As an example, an isolated key to be used for template matching is shown to the right.

Perform the following steps:

- Choose your favorite key and crop with a narrow boundary. Use this image as your template.
- Prepare the template image as follows:
  - Calculate the min, max and mean value of the template image.
  - Normalize the template image by subtracting the mean value from all pixel values, so that the mean becomes 0.0. This will be your zero-mean template.
- Implement cross-correlation with the original image and the template as mask.
- Please recall that this results in a peak (maximum) if the template matches the specific image region which you selected for your template.

- Do a pass through the correlation image to detect the maximum peak value and its (x,y) location. You may mark this location with an overlay of a circle or just manually painting an arrow or else. Discuss if this location matches your expectations. Discuss the appearance and cause of other peaks, and how they compare to the global peak, you may just manually check other peaks and annotate them on the peak image.
- In your report, show the original image, peak image (output of cross-correlation), and your template which was used for correlation.
- Voluntary: Experts may overlay the peak image with the original peak image so that one sees where major peaks were found, best after thresholding the peak image to show the largest peak clusters.
- Include all results and discussion in your report. Remember your report needs to be a standalone document that someone outside of class can follow.

3) Creative Part

You may now use your filtering modules to solve problems of your choice, below just a few ideas you may want to pursue:

1. **Color-Edge-Detection**: The edge-filtering (Gaussian filtering and gradient magnitude calculation) can be extended to color images. Interesting would be a conversion of a color image into the "LCH" color shape (use python), then separate the three L,C,H channels and apply Edge Detection to each of those. You can then also calculate the sum of all these 3 edge images. In contrast, you can then convert a color image to greyscale, apply the regular edge-detection, and compare the grayscale, each color channel, and the combined color edge detection.
2. **Counting objects:** Given series of same type of objects, could you do automatic counting? (e.g. search for "google images counting objects" where you see charts used for kids). Another application may be sorting fruits on a flat surface, or else. Such a method would require thresholding of the peak image (choose manual threshold that you may think gives the best blobs of all peaks. This is then followed by connected-component analysis**(*)** which gives you labels for each connected regions. Counting those labels will finally give you the number of objects.
3. **Counting cars in a car manufacturer's lot (e.**g. link or link2). A manufacturer may want to know how many cars of a specific type are parked on its parking lot. Procedure same as .in 1).
4. **Sorting by similarity**: Peak images show different strengths of objects that deviate from the template itself. Using connected component labeling on thresholded peak images as in 1), you could detect the maximum peak value within each connected component and sort them by size. This could be interesting to be applied to family pictures (as showed in class), or the key template example, or other images of your choice.
5. **Template Contour filtering**: Instead of filtering a mask with the gray-level image template, you may consider filtering a contour image with a contour template: Apply your gradient magnitude filter to the original image, crop a small template from this processed image, normalize the template by zero-mean,

and apply your convolution as done in section 2. Comparing to the regular template convolution result's peak image, would you see a benefit?

6. **Etc**.

(*) connected component analysis tool provided by the TA's