CS 6643 - Computer Vision, Spring 2021 **James Fishbaugh**

Project 2

Out: 3 / 16 / 2021
Due: 4 / 05 / 2021 (deadline: 11:55PM)

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 implementation and 75% for written report

Office hours:

		Mon	Tue	Wed	Thur	Fri
James Fishbaugh	jf146@nyu.ed	4-5 PM		4-5 PM		
Michael Lally	mfl340@nyu.ed		12-1 PM		10-11 AM	
Akshat Khare	ak7674@nyu.edu	2-3 PM				10-11 AM

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 submit a single PDF/HTML with all code included as an appendix.**

Hint for implementation: Most image processing require using float or double type for image arrays. You can map those images back to integer at the very end of the processing for display purposes, and during this operation you can also optimally scale and shift the result to [0..255] after calculation of minimum and maximum intensities across the resulting image.

1) Convolution and Derivative Filters

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.

```
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
```

Derivative Filters: Using the example image 'cameraman.png'

- Denoise the image with a Gaussian filter. You may use the code from the in class demo to create the Gaussian filter, but you must use your own implementation of convolution.
- 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 $[-1\ 0\ 1]^T$. You can hardcode the derivative filter, but use your implementation of convolution.
- Compute the gradient magnitude image.
- Create binary edge images from the gradient magnitude image using several thresholds

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

The uploaded image '*multiplekeys.png*' is a single image containing multiple instances of keys.

Perform the following steps:

- Threshold the original key image so that the background is [0.0] and keys appear as [1.0]. You should have a binary image with only [0.0] (background) and [1.0] (keys).
- Choose your favorite key and crop with a narrow boundary. Use this image as your template.
- Modify your template by setting background pixels to [-1.0], so that you have [+1.0] for the key and [-1.0] for background. The reason for creating a signed template is improved matching performance.
- Implement cross-correlation with the binary input image and your new signed template image.
- 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 similar. Discuss if this location matches your expectation. Discuss the appearance and cause of other local peaks, and how they compare to the global peak.

In your report, show the original image, peak image (output of cross-correlation), and your template which was used for correlation.

3) Image Panoramas

An image panorama is created by stitching together several images. The images must be taken under certain criteria: the camera cannot undergo translation between pictures; only rotational transformations are allowed.

If you have ever tried to create a panorama by hand by cutting out pictures and taping them together you know that it doesn't work very well. Translation and rotation are not sufficient to align the two images together. In order for the panorama to look right, one of the images must also be transformed/warped.

A key observation is that images taken under the aforementioned criteria are equivalent under a perspective projection. We can take corresponding points between two images and create a linear system. In general, we choose many correspondences so that the system is overconstrained, and the transformation is found by linear least squares.

The following equations detail how to set up the matrix system:

Project 2

Linear transformation with matrix P

$$\bar{x}^* = P\bar{x} \qquad P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{pmatrix}$$

$$\begin{aligned} x^* &= p_{11}x + p_{12}y + p_{13} \\ y^* &= p_{21}x + p_{22}y + p_{23} \\ z^* &= p_{31}x + p_{32}y + 1 \end{aligned}$$

Perspective equivalence

$$\begin{aligned} x' &= \frac{p_{11}x+p_{12}y+p_{13}}{p_{31}x+p_{32}y+1} \\ y' &= \frac{p_{21}x+p_{22}y+p_{23}}{p_{31}x+p_{32}y+1} \end{aligned}$$

Multiply by denominator and reorganize terms

$$\begin{aligned} p_{31}xx' + p_{32}yx' - p_{11}x - p_{12}y - p_{13} &= -x' \\ p_{31}xy' + p_{32}yy' - p_{21}x - p_{22}y - p_{23} &= -y' \end{aligned}$$

Linear system, solve for P

$$\begin{pmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 \\ & & & \vdots & & & & \\ -x_N & -y_N & -1 & 0 & 0 & 0 & x_Nx'_N & y_Nx'_2 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y'_1 & y_1y'_1 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y'_2 & y_2y'_2 \\ & & & \vdots & & & & \\ 0 & 0 & 0 & -x_N & -y_N & -1 & x_Ny'_N & y_Ny'_N \end{pmatrix} \begin{pmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{23} \\ p_{23} \\ p_{31} \\ p_{32} \end{pmatrix} = \begin{pmatrix} -x'_1 \\ -x'_2 \\ \vdots \\ -x'_N \\ -y'_1 \\ -y'_2 \\ \vdots \\ -y'_N \end{pmatrix}$$

- Take two photos which overlap and differ only in rotation, e.g. similar to the example images above.
- Implement an image panorama for the case of 2 images (a source and a target). The algorithm is:
    - Define corresponding points between the two images (these are pixel coordinates which you can find by hovering your mouse in a paint program, or found in any manner of your choice).
    - Using the corresponding points, construct the known matrix and vector shown above. Note that this is similar to our affine example but with a different matrix structure.
    - Solve for parameters P with linear least squares (e.g. use np.linalg.lstsq or similar).
    - Using the found parameters P, transform the target image. You must implement the transformation yourself, but you may use any code from the demos. You can use any interpolation of your choice.
    - Combine the source and transformed target together on the same canvas. You can handle the area of overlap in anyway you choose (simplest is to always overwrite the grey values in the canvas image). Don't worry if your panorama has an obvious seam.
- Compare results using 4, 5, 6, and 7 corresponding points.

Include all results and discussion in your report. Remember your report needs to be a standalone document that someone outside of class can follow.
