PHYS 481 Assignment 2: Images and Filters

Due: Friday Sept 20 (by 23:00)

AI policy for this assignment: no use of generative AI tools is allowed.

Note the slightly different submission requirement for this assignment (see below).

This assignment will introduce some basic image processing techniques. Other than questions 2c and 2d, which require the pillow and numba packages respectively, please use only numpy and matplotlib. The file 'a2_template.ipynb' has some code that will help get you around some Python/Jupyter quirks with displaying images.

Question 1 [3 pts for code, 3 pts for images] [see rubric at end]

Load the file 'mandrill.png' using matplotlib and perform some basic manipulations on it. For each of these questions, display the original and the altered image side-by-side with no rescaling of the pixel size. You can use the helper function in the template.

a. Create a 10-pixel wide vertical black line through the middle of the image by replacing data values in the image with zeroes. Display the original and the altered image side-by-side.

b. Transpose the original image. Display the original and the altered image side-by-side.

c. "Posterize" the original image by reducing the number of different color levels in each channel from 256 to 2. In other words, reduce the precision in the data values in each color channel (R,G,B) so they assume only the value 0 or 1, and no values in between. Display the original and the altered image side-by-side.

Question 2 [3 pts for code, 4 pts for images, 1 pt for table and discussion]

To rotate an n-row, m-column image by an angle θ in the anticlockwise direction, pixel location [i,j] in the rotated (new) image corresponds to pixel location [x,y] in the original image, where

$$x = \left(i - \frac{n}{2}\right)\cos\theta + \left(j - \frac{m}{2}\right)\sin\theta + \frac{n}{2}$$

$$y = -\left(i - \frac{n}{2}\right)\sin\theta + \left(j - \frac{m}{2}\right)\cos\theta + \frac{m}{2}$$

With a few notes:

1. If [x,y] falls outside the range of the original image (which happens on the edges of the rotated image), a fill color should used. Please use black for this assignment.

2. The values of x and y will generally be non-integer, so this rotation requires evaluation of the image data at locations that are not exactly on pixel centers in the original image. In other words, some sort of interpolation scheme is required. For this assignment, please simply round to the nearest integer. This is known as "nearest neighbor" interpolation. Bi-linear interpolation or other interpolation schemes are sometimes used (and have advantages in some circumstances), but please just use nearest neighbor for this assignment.

a. Load the mandrill image and rotate it 20 degrees anticlockwise using the equations above implemented in a nested loop. Include all the calculations (even the trig functions) in the loop. Time the execution using %timeit. Plot the original and rotated images.

b. Repeat question 2a but pull the trig functions and anything else you can pre-calculate outside the loops and rotate the original image 20 degrees clockwise.

c. Repeat question 2b but use Just-In-Time compilation and rotate 30 degrees anticlockwise.

d. Repeat question 2a but use PIL (Python Image Library) to perform the rotation and rotate 30 degrees clockwise.

e. Prepare a table of how long each method took. In a few sentences, comment on the speed of the various methods and why you think there was such a difference.

Question 3 [3 pts for code, 3 pts for images]

Image arrays can be usefully filtered by 2D centered FIR filters:

$$y'_{i,j} = \sum_{m=-m_1}^{m_1} \sum_{n=-n\_1}^{n_1} a_{m,n} y_{i+m,j+n}$$

These filters can perform many different functions including blurring, sharpening and edge detection. The coefficient array $a_{m,n}$ is known as the filter "kernel" and has size $2m_1 + 1$ by $2n_1 + 1$. Filter kernels are usually square, with $m_1 = n_1$ This filtering operation is known as 2D convolution.

Edge detection is usually accomplished by smoothing (to remove image noise) and then applying a derivative operator. Further nonlinear operations can be used to refine the edges by suppressing weak edges, joining strong edges, etc. A commonly used derivative kernel is the Sobel operator. For the horizontal (x) derivative, the Sobel operator is:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

and the vertical derivative uses the transpose $S_y = S_x^T$. These are simple extensions of the 1D 3-point stencil derivative operator from last week. The gradient magnitude of array A is then

$$G = \sqrt{(S_x * A)^2 + \left(S_y * A\right)^2}$$

where the $*$ symbol denotes 2D convolution and the square root is taken on each element of the array individually.

Another commonly used filter kernel is the Laplacian (∇^2) kernel

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

which detects second derivatives. This is sometimes used in image sharpening filters, which take the form

$$K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \alpha L$$

where the degree of sharpening can be controlled by α.

Start by writing a general function to apply an arbitrary filter (2D convolution) of the type described above, and then answer the following questions. Your function may discard the edge points where the filter cannot be applied (m_1 points from each of the 2 edges in the first dimension and n_1 from the 2 edges in the second dimension).

a. Load the mandrill image again and perform a "boxcar smoothing" with a 15x15 kernel of the form $a_{m,n} = 1/15^2$. Display the original and smoothed images.

b. Load the "peppers.png" image and apply a 5x5 boxcar smoothing. Find the gradient magnitude G using the Sobel operator on each channel (R,G,B) and sum them to get an overall gradient. Apply a threshold of 30% of the maximum gradient to discard weak edges (that is, if the gradient is less than 30% of the max, set the RGB values to 0,0,0 and otherwise set R,G,B=1,1,1). Plot the original image and the processed edges.

c. Load the "eye.png" image and apply a Laplacian sharpening filter with kernel K as given above and $\alpha = 0.3$ (moderate sharpening) and then with $\alpha = 3.0$ (extreme sharpening). This filter may cause some of the RGB values to go outside their nominal [0,1] range; they should be clipped to constrain the range. Plot the original image and the sharpened image for both values of α.

Submission

Gradescope has difficulties with Jupyter notebooks larger than 1 MB. For this assignment, your notebook will almost certainly be larger than 1 MB, so please submit your work as a PDF file instead. To create a PDF from your Jupyter notebook:

1. If you're using a JupyterLab environment (like syzygy), try selecting File -> "Save and Export Notebook As…"->PDF.
2. If you're using Visual Studio Code, click the "…" (More Actions) at the top of the file editing window (beside "Outline") and select "Export" and then "PDF".
3. If you encounter trouble with exporting your PDF (which is not uncommon on many platforms), log into ucalgary.syzygy.ca, upload your .ipynb file and follow step 1.

Rubric

Each question receives a grade for coding and images. For 2e, there is 1 point for a table and discussion.

Code	**Commenting**: Clear and concise comments explaining the code.	1 pt	0: Missing or major error. 0.5: Minor error. 1: Correct.
	Logical Structure: Code is logically organized into functions and modules.	1 pt	
	Readability: Code is well-formatted with consistent and easily understood naming conventions.	1 pt	
Images	**Correctness**: Image shows the expected outcome of the question.	1pt	
Table	**Readability**: Table and discussion are clear and logical.	1pt	

Image sources:

mandrill.png and peppers.png are standard test images from the USC Signal and Image Processing Institute available at https://sipi.usc.edu/database/database.php?volume=misc and are public domain.

eyes.png is from the Wikipedia page for unsharp masking and is available at https://commons.wikimedia.org/wiki/File:Unsharped_eye.jpg . Image author is Ru_dragon, and the image is distributed under the GNU Free Documentation License.