COMP 4102A: Assignment 1
Ryan Lo (101117765)

Instructions for submission: Please submit a PDF with your solutions on theory questions. The PDF should explain your work. For coding questions you may use C/C++ and OpenCV or Python with OpenCV Python for this assignment. Comment your code to make it easier to grade. Include your codes for edge detection in a folder. Submit a single zip file that contains: 1 - PDF with your answers to question 1; 2 - folder containing edge detection code, images, and readme file that explains how to run your code; 2 - folder containing your sticks filtering code, images, and readme file that explains how to run your code. Please submit through Brightspace. You are expected to work on the assignment **individually**. Do not leave your submission to the last minute because if you run into technical issues the system will cut you off. You can submit multiple versions of your assignment, and we will grade the latest one.

1 (30 points) Theory questions

1. (5 points) Are three dimensional rotations expressed as R_x, followed by R_y, and then R_z (rotations around the x, y and z axis) commutative? That is, does the order in which they are applied matter. Explain the answer.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & -\mathrm{b} \\ 0 & \mathrm{b} & \mathrm{a} \end{bmatrix}, R_y(\theta) = \begin{bmatrix} \mathrm{a} & 0 & \mathrm{b} \\ 0 & 1 & 0 \\ -\mathrm{b} & 0 & \mathrm{a} \end{bmatrix}, R_z(\theta) = \begin{bmatrix} \mathrm{a} & -\mathrm{b} & 0 \\ \mathrm{b} & \mathrm{a} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using these 3 matrixes, we'll test to see if rotations are commutative.

Let's use the vector $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ to test this.

We'll try $R_x * R_y * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ $and\ R_y * R_x * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$$R_x * R_y * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = R_x * \begin{bmatrix} \mathrm{a} & 0 & \mathrm{b} \\ 0 & 1 & 0 \\ -\mathrm{b} & 0 & \mathrm{a} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= R_x \begin{bmatrix} a+b \\ 1 \\ -b+a \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & -\mathrm{b} \\ 0 & \mathrm{b} & \mathrm{a} \end{bmatrix} \begin{bmatrix} a+b \\ 1 \\ -b+a \end{bmatrix}$$

$$= \begin{bmatrix} a+b \\ a-b(-b+a) \\ b+a(-b+a) \end{bmatrix}$$

Compute the other one and compare

$$R_y * R_x * \begin{bmatrix}1\\1\\1\end{bmatrix} = R_y * \begin{bmatrix}1 & 0 & 0\\0 & a & -\mathrm{b}\\0 & \mathrm{b} & \mathrm{a}\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$= R_y * \begin{bmatrix}1\\a-b\\b+a\end{bmatrix}$$

$$= \begin{bmatrix}\mathrm{a} & 0 & \mathrm{b}\\0 & 1 & 0\\-\mathrm{b} & 0 & \mathrm{a}\end{bmatrix}\begin{bmatrix}1\\a-b\\b+a\end{bmatrix}$$

$$= \begin{bmatrix}a+b(b+a)\\a-b\\-b+a(b+a)\end{bmatrix}$$

As you can see rotating them in different ways produces a different output.

2. (8 points) Find the SVD of A, $U\sum V^T$, where $A = \begin{bmatrix}2 & 2 & 0\\-1 & 1 & 0\end{bmatrix}$

$$AA^T = \begin{bmatrix}2 & 2 & 0\\-1 & 1 & 0\end{bmatrix}\begin{bmatrix}2 & -1\\2 & 1\\0 & 0\end{bmatrix} = \begin{pmatrix}8 & 0\\0 & 2\end{pmatrix}$$

$$AA^T = \begin{bmatrix}2 & -1\\2 & 1\\0 & 0\end{bmatrix}\begin{bmatrix}2 & 2 & 0\\-1 & 1 & 0\end{bmatrix} = \begin{pmatrix}5 & 3 & 0\\3 & 5 & 0\\0 & 0 & 0\end{pmatrix}$$

$$A - \lambda I = \begin{pmatrix}5-\lambda & 3 & 0\\3 & 5-\lambda & 0\\0 & 0 & -\lambda\end{pmatrix}$$

$$= (0)(-1)^{3+1}\begin{vmatrix}3 & 0\\5-\lambda & 0\end{vmatrix} + (0)(-1)^{3+2}\begin{vmatrix}5-\lambda & 0\\3 & 0\end{vmatrix} + (-\lambda)(-1)^{3+3}\begin{vmatrix}5-\lambda & 3\\3 & 5-\lambda\end{vmatrix}$$

$$= (-\lambda)\begin{vmatrix}5-\lambda & 3\\3 & 5-\lambda\end{vmatrix}$$

$$\det(2x2) = ad - bc$$

$$= (5-\lambda)*(5-\lambda) - 3*3 = \lambda^2 - 10\lambda + 16$$

$$det = -\lambda(\lambda-8)(\lambda-2)$$

Eigenvalues are 0, 2, 8

For $\lambda = 0$,

$$\begin{bmatrix}5 & 3 & 0\\3 & 5 & 0\\0 & 0 & 0\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

For $\lambda = 2$,

$$\begin{bmatrix}3 & 3 & 0\\3 & 3 & 0\\0 & 0 & -2\end{bmatrix} = \begin{bmatrix}1 & 1 & 0\\0 & 0 & 1\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}-1\\1\\0\end{bmatrix}$$

For $\lambda = 8$,

$$\begin{bmatrix} -3 & 3 & 0 \\ 3 & -3 & 0 \\ 0 & 0 & -8 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$\sigma_1 = \sqrt{8} = 2\sqrt{2}, \sigma_2 = \sqrt{2}$

$$\Sigma = \begin{bmatrix} 2\sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{bmatrix}$$

$$\bar{u}_1 = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}, \bar{u}_2 = \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}$$

$$u_1 = \frac{1}{\sigma_1} * \begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 0 \end{bmatrix} * v_1 = \frac{1}{2\sqrt{2}} * \begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$u_2 = \frac{1}{\sigma_2} * \begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 0 \end{bmatrix} * v_2 = \frac{1}{\sqrt{2}} * \begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad V^T = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A = U\Sigma V^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Hint: first find $A^T A$, then find λ by solving $det(A^T A - \lambda I) = 0$. Look at this example to find out how to calculate the U and V : https://www.d.umn.edu/~mhampton/m4326svd_example.pdf

3. (4 points) Scale a vector $[x\ y]^T$ in the plane can be achieved by $x' = sx$ and $y' = sy$ where s is a scalar.
(a) Write out the matrix form of this transformation.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

(b) Write out the transformation matrix for homogeneous coordinates.

Transformation matrix is $\begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix}$.

(c) If the transformation also includes a translation
$x' = sx + t_x$ and $y' = sy + t_y$
Write out the transformation matrix for homogeneous coordinates.

Transformation matrix shifted by the translation is $\begin{pmatrix} s+t & 0 \\ 0 & s+t \end{pmatrix}$.

(d) What is the equivalent of the above matrix for three-dimensional vectors?

For a three-dimensional vector that is equivalent to the matrix above, we have
$x' = sx + t_x\,,\ y' = sy + t_y\,,\ z' = sz + t_z$

$$\begin{pmatrix} s+t & 0 & 0 \\ 0 & s+t & 0 \\ 0 & 0 & s+t \end{pmatrix}$$

4. (5 points) Find the least square solution x for $Ax = b$ if

$$A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Verify that the error vector $b - A\bar{x}$ is orthogonal to the columns of A.

$$A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, A^T = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 1 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

$$A^T A x = A^T b$$

$$A^T A = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 5 & -1 \\ -1 & 5 \end{bmatrix}$$

$$A^T b = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

Putting the two together:

$$\begin{bmatrix} 5 & -1 \\ -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

$$5x_1 - x_2 = 2$$
$$-x_1 + 5x_2 = -2$$

We get:

$$x_1 = \frac{1}{3}, x_2 = -\frac{1}{3}$$

Now we verify

$$b - Ax$$

$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ -\frac{1}{3} \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} - \begin{bmatrix} 2/3 \\ -2/3 \\ -2/3 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \\ -\frac{1}{3} \end{bmatrix}$$

Now we check the dot product

$$\begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \\ -\frac{1}{3} \end{bmatrix} \circ \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} = 0$$

$$\begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \\ -\frac{1}{3} \end{bmatrix} \circ \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} = 0$$

Therefore the error vector is orthogonal to A.

5. Matrix K is a discrete, separable 2D filter kernel of size $k \times k$. Assume k is an odd number. After applying filter K on an image I, we get a resulting image I_K.

(a) (3 points) Given an image point (x, y), find its value in the resulting image, $I_K(x, y)$. Express your answer in terms of I, k, K, x and y. You don't need to consider the case when (x, y) is near the image boundary.

$$I_K(x,y) = \sum_{m=1}^{k}\sum_{n=1}^{k}\left(K_{mn} I * \left(x - \left\lfloor\frac{k}{2}\right\rfloor + m, y - \left\lfloor\frac{k}{2}\right\rfloor + n\right)\right)$$

(b) (5 points) One property of this separable kernel matrix ***K*** is that it can be expressed as the product of two vectors $g \in R^{k\times 1}$ and $h \in R^{1\times k}$, which can also be regarded as two 1D filter kernels. In other words, $K = gh$. The resulting image we get by first applying *g* and then applying *h* to the image ***I*** is I_{gh}. Show that $\boldsymbol{I_K} = I_{gh}$.

We can start by using an example kernel matrix K

$$K = \begin{bmatrix} 1 & 3 \\ 3 & 9 \end{bmatrix}$$

We can take g and h to be

$$g = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, h = \begin{bmatrix} 1 & 3 \end{bmatrix}$$

Take the average values for the filter:

$$\frac{1+3+3+9}{4} = 4$$

Take g and apply it with the average values

$$\frac{1+3}{2} = 2$$

Now apply it to h and we get:

$$\frac{1*2+3*2}{2} = 4$$

This shows that $I_K = I_{gh}$

2 (35 points) Edge detection

Write a function that finds edge intensity and orientation in an image. Display the output of your function for one of the given images in the handout.

$$function[img1] = myEdgeFilter(img0, sigma) \quad (1)$$

The function will input a greyscale image (img0) and scalar (sigma). sigma is the standard deviation of the Gaussian smoothing kernel to be used before edge detection. The function will output img1, the edge *magnitude* image.

First, use your convolution function to smooth out the image with the specified Gaussian kernel. This helps reduce noise and spurious fine edges in the image. The size of the Gaussian filter should depend on sigma (e.g., hsize = 2 * ceil(3 * sigma) + 1).

The edge magnitude image img1 can be calculated from image gradients in the x direction and y direction. To find the image gradient imgx in the x direction, convolve the smoothed image with the x-oriented Sobel filter. Similarly, find image gradient imgy in the y direction by convolving the smoothed image with the y-oriented Sobel filter. You can also output imgx and imgy if needed.

In many cases, the high gradient magnitude region along an edge will be quite thick. For finding lines its best to have edges that are a single pixel wide. Towards this end, make your edge filter implement non-maximum suppression, that is for each pixel look at the two neighboring pixels along the gradient direction and if either of those pixels has a larger gradient magnitude then set the edge magnitude at the center pixel to zero. Map the gradient angle to the closest of 4 cases, where the line is sloped at almost0^o, 45^o, 90^o, and 135^o. For example, 30^o would map to 45^o. Please refer to the slides on non-maximum suppression for more details. You do not need to do hysteresis thresholding, but use a simple thresholding.

Your code cannot call on OpenCV edge function, or any other similar functions. You may use the edge detection from library just for comparison and debugging.

Submission

All results should be in a folder called **Edge detection**. You can use your own images and you should include the original image in the folder if you use your own. Submit your code with the **edge detection** result after non-maximum suppression, the **gradient magnitude** image and the **gradient orientation** image. Include your result from sticks filtering to this folder. Your code should have comments for each function and clear variables.

3 (35 points) Sticks filter

To enhance thin and connected line structures, we can use the sticks filter[1], originally designed to reduce speckle noise and preserve linear structures. Sticks filtering uses the maximum result from the m oriented filters as the output for the current pixel. Use the maximum response to decide what to do at the central pixel. Compute the difference among all the directions around the region. The chosen orientation, with maximum difference in intensity from the region, aligns with the locally strongest feature. The intensity should be increased along the direction of maximum response, amplifying local tonal differences.

You can create a sticks map T(x,y) using equation 1:

$$T(x,y) = \max_{1 \le s \le i} \Delta(\mu(s_i) - \hat{I}_n(x,y)) \tag{1}$$

$$\hat{I}_n(x,y) = \frac{\sum_{(r,s)\in nxn} I(r,s)}{n^2} \tag{2}$$

$$\mu(s_i) = \frac{\sum_{(j,k)\in s_i} I(j,k)}{n} \tag{3}$$

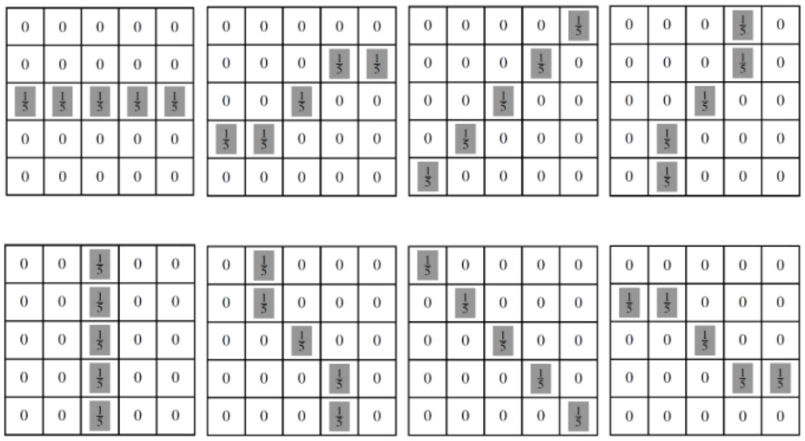

Figure 1: Rasterization of a set of discrete directions, representing 8 line orientations (length n = 5)

The sticks direction T(x,y) determines the maximal contrast response, defined as the maximum difference between the average intensity $\mu(s_i)$ of a stick and the average intensity $\hat{I}n(x,y)$ of the neighboring pixels. Parameter n is the length of each stick; there are i sticks. Apply sticks filtering on **gradient magnitude** image, using n = 5 for i = 8 sticks orientations to enhance the edge segments before applying non-maxima suppression.

Submission

Show the effect of your sticks filtering on your **gradient magnitude** image. Show the results of the edge detection after applying the sticks filtering on the **gradient magnitude**. Note that you need to write a function that creates the kernels of the sticks filter. You can use filter2D function in OpenCV for convolution of the sticks filter kernels.

[1]Line And Boundary Detection In Speckle Images (1997)