Problem 1B: Have some fun with 2D convolutions and image processing !

Pertinent readings for Problem 1b:

(same as the 4 excerpts from page 1 of Problem 1a)

Part 1: Gain some mathematical intuition for 6 different filters

Suppose you were given a toolbox with 6 filters, a collection of equivalent calculus operations, and 2 main filter jargons of interest:

Filters

$$H_1 = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (\ Laplacian\ )$$

$$H_3 = \frac{1}{4}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{pmatrix} 3 - point\ weighted \\ moving\ avg \end{pmatrix}$$

$$H_4 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{pmatrix} Sobel \\ x - direction \end{pmatrix}$$

$$H_5 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{pmatrix} Gaussian\ blur \\ 3 \times 3 \end{pmatrix}$$

$$H_6 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Jargons

Smoothing filters

Edge-detection filters

Equivalent calculus operations

$$\frac{\partial u}{\partial x} \quad , \quad \frac{\partial u}{\partial y}$$

$$\frac{\partial^2 u}{\partial x^2} \quad , \quad \frac{\partial^2 u}{\partial y^2}$$

$$\frac{\partial^2 u}{\partial x \ \partial x} \quad , \quad \frac{\partial^2 u}{\partial y \ \partial x}$$

$$\int u \ dx \quad , \quad \int u \ dy$$

★ For this problem: You want to think about *finite differences approximations* of $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial x}$, $\frac{\partial^2 u}{\partial x^2}$, etc !! ★

Your tasks for this section:

1) For each of the 6 above filters, match the filter to all pertinent calculus operations associated to that filter, and also, tag that filter with the appropriate "filter jargon."

For example, if our filter was H_0 (shown below), I would associate it with:

Filter	*Calculs operations*	*Jargon*
$H_0 = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{du}{dx}\begin{pmatrix} centerpoint \\ slope \end{pmatrix}$	*Edge detection*

** You can echo your answers in matlab using the *disp( )* function if you'd like !!

Part 2: Treasure hunt !

Suppose you were doing research in the radiology department at Beth Israel, and you were presented with the x-ray image in Figure 1, where a patient "accidentally" swallowed a toothbrush……… =/

(This was a ***real case study***, btw !!!!! I pulled it off of *radiopaedia.org*):

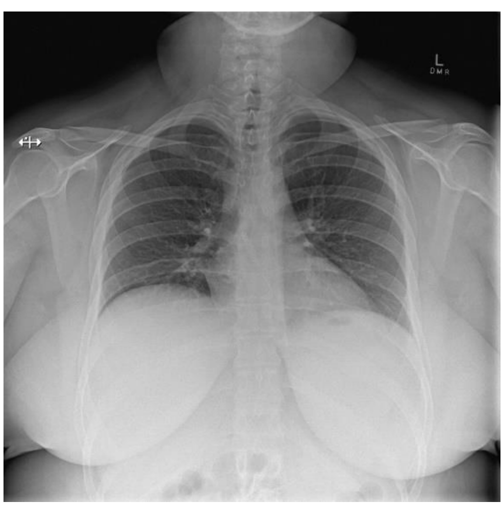

Figure 1a: X-ray photo of a real patient, where the patient swallowed a toothbrush ! This is the frontal view

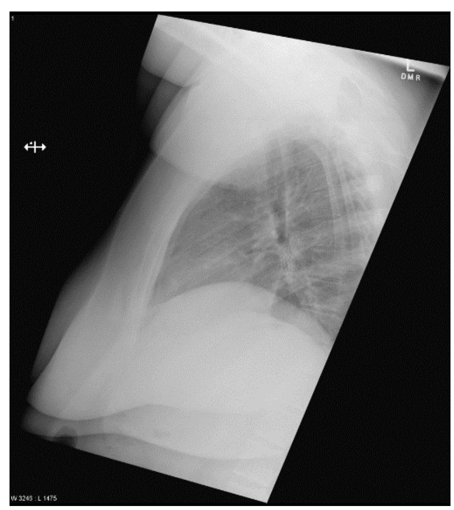

Figure 1b: Side view of the same patient

Your tasks for this section:

1) Using *imread*, load the image "swallowed_toothbrush_verB_frontal.tif" into matlab as matrix X.

2) Using matlab's *imshow* command, immediately plot your original image to make sure that you've loaded it correctly.

3) Using matlab's *double* command, convert your matrix X into "math-able" double-precision floating point numbers.

4) Now, let's check out the maximum / minimum pixel intensity values for your image by plotting a *pcolor* version of matrix X. Add a colorbar to your figure, and label your axes as:

i) Horizontal axis label: "x-axis (pixels)"
ii) Vertical axis label: "y-axis (pixels)

iii) For this example, it's easier to see what going on by using the inverted version of the "pink" colormap:

$$colormap(\,flipud(copper)\,) \rightarrow \begin{array}{c} Notice\ the\ black\ and\ white \\ tones\ have\ been\ reversed: \end{array} \begin{cases} Black\ = 255 & (dark\ brown - black) \\ White\ = 0 & (bright\ orange) \end{cases}$$

Ok ! Here's the fun part =) Suppose you were given 6 new convolution filters:

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad H_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad H_3 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$H_4 = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad H_5 = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad H_6 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

5) There is one filter that will perform way better than the others in revealing the location of our toothbrush ! Try them all, and when you find the right filter, do the following:

a) Plot a *pcolor* version of your post-filtered image $Y = H * X$

b) Add a colorbar to your pcolor figure.
Make the title of your pcolor plot as: "Filtered pcolor image"

c) For ease in visualizations, limit your colorbar axis color shadings to see the finer details:

caxis ([-250 250]) $\rightarrow$ $\textit{You can vary these values to enhance the contrast of your image (if you want to)}$!

d) Then, apply the inverted version of the pink colormap to your pcolor image.

$colormap(\ flipud\ (copper)\ )$ $\rightarrow$ $\textit{Again, I think this one works the best for this image, but you can try other colormaps (inverted or not) and see if it works better for you}$

e) Using matlab's *uint8* function, convert your filtered image Y into unsigned 8-bit integers.

f) Using matlab's *imshow* , plot your filtered image. Indicate that this photo is the post-filtered image by adding a title to this figure.

g) Finally: Indicate the location of the toothbrush on your plot by either:

i) Using Microsoft paint (or whatever drawing tool you have on your laptop), paint a circle around the toothbrush.... Or

ii) Tell me the approximate (x, y) – pixel location of the toothbrush

★ *Hint: The toothbrush is easier to see if you actually try to analyze the lateral X-ray image ! Go ahead and open up "swallowed_toothbrush_verB_lateral.tif" to get an approximate location =)* ★