Filtering and Frequencies

Discussion #3

Written by:
Chung Min*, Justin*

Topics

This section covers filters, convolution, and frequency decomposition.

Logistics

Remember Project 2 is due Friday, 9/26 at 11:59pm!

1 Convolution

Problem 1.1: ***Basic convolution.*** Consider the following 5×5 image and 3×3 filter:

Image:

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

Filter:

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

Compute the convolution of the image with the filter by using 0-padding. What is the resulting 5x5 output image?

0	1	2	3	2
0	1	2	3	2
0	1	2	3	2
0	1	2	3	2
0	1	2	3	2

Problem 1.2: ***Feel the filter.*** Give any 3x3 example of each of the following types of image convolution filters:

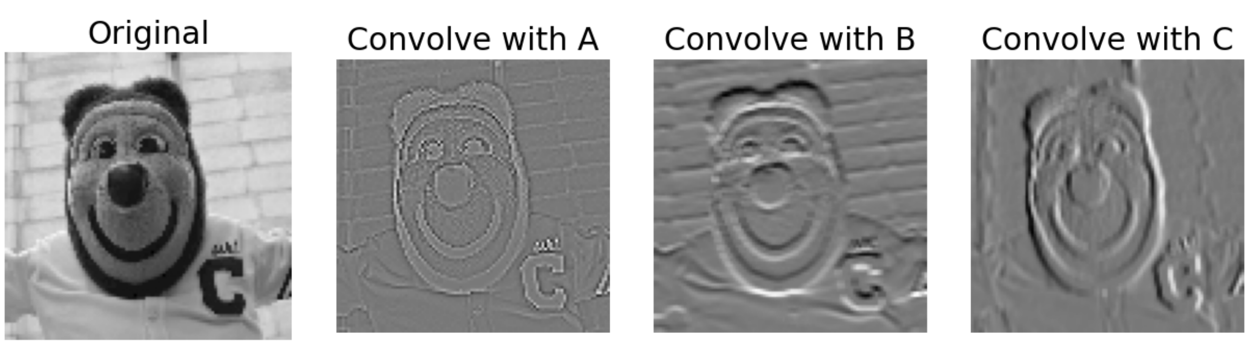
(a) Image darkening

(b) 1-pixel left shift

(c) Smoothing

(a) anything that doesn't sum to 1 would work since it would reduce the average pixel value. (b) A pattern with a 1 on the center left of the image and 0s everywhere else (remember that filters are flipped during convolution). (c) Examples can be a box filter (all 1/9), or any sort of centered spreading function which sums to 1.

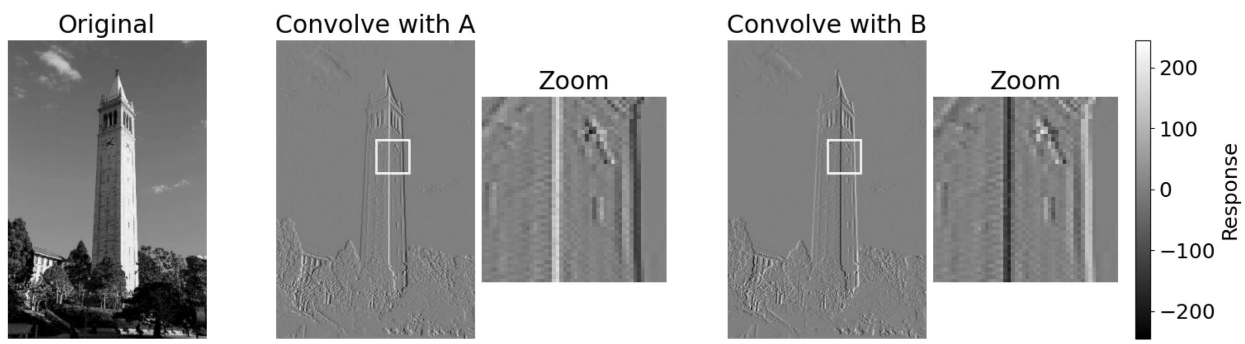
Problem 1.3: ***Up down, left right?*** Match the filters to their names.



$$\_\_\_\_\_\_\_\_ : \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad \_\_\_\_\_\_\_\_ : \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad \_\_\_\_\_\_\_\_ : \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

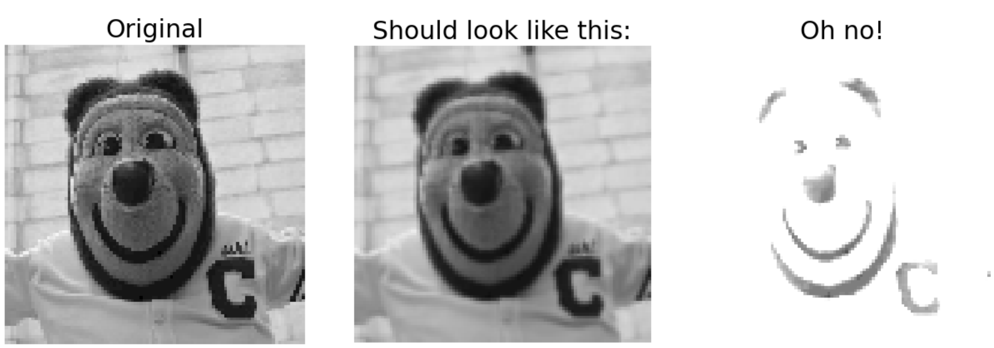
The first kernel results in output B, the second in C, and the third in A. For the first two, pay attention to which edges produce a signal in the output activation (vertical edges or horizontal edges), and notice that in the third kernel it should uniformly respond to all directions of change in the image. The first two are examples of directional image derivatives and the third is a laplacian filter.

Problem 1.4: ***Finding direction (on campus).*** Write the 1x3 convolution filters for A and B. *Hint: one has values* $[1, 0, -1]$*, and the other has* $[-1, 0, 1]$*. Pay close attention to the colors!.*



A is convolved with [1,0,-1] and B is convolved with [-1,0,1]. You can think about this by working backwards: pick an edge in the image and note whether it is light-to-dark or dark-to-light. For example in image A, the main central shadow line goes from dark to light, and the result is highly activated (white output). I like to think about cross-correlation first and then flip the result to avoid confusion. This means that the filter responds strongly to increasing signals, which would be *correlation* of [-1,0,1], hence flipping results in the *convolution* kernel of [1,0,-1].

Problem 1.5: ***Did you forget a step?*** A student made a mistake constructing an image blur, and got the resulting image. What did they forget to do?



They forgot to design a filter whose values sum to 1, meaning that the pixels aren't averaged but multiplied by a positive scalar, increasing brightness.

Problem 1.6: ***Invariance***

1. Is the Sobel filter invariant to constant image brightness shifts? (i.e., $+/-$ pixel values)

   Yes! derivatives measure local intensity changes, which are unaffected by uniform shifts.

2. Is the Sobel filter invariant to image contrast? (recall histogram equalization from Lecture)

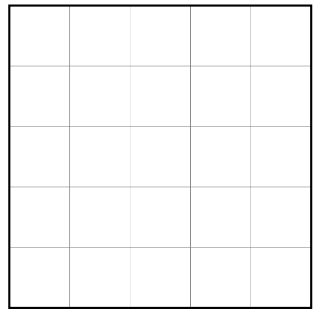
   No, changing image contrast will change the difference in pixels between images (for example equalizing the histogram stretches values such that light becomes lighter and dark becomes darker.)

2 Frequency

Problem 2.1: Intuition
(a) Which has on average higher frequency? `I` or `blur(I)`?

The original image I has higher average frequency (except at the edge case where blurring doesn't change the image). Blurring is a low-pass filter operation that removes high-frequency components (sharp edges, fine details), leaving mainly low-frequency components (smooth variations).

(b) Suppose I want to fill in this 5x5 image in a way that produces high frequencies in the FFT. How could I do that? (There are lots of correct answers)



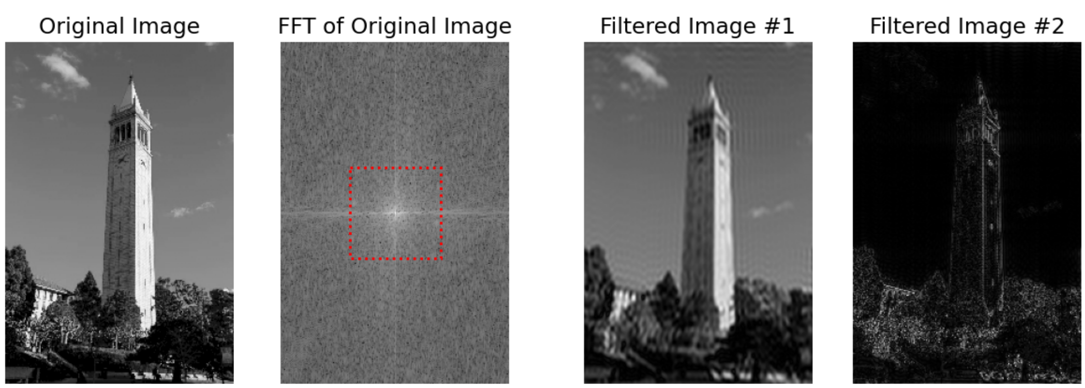
In words, what sort of image features produce high frequencies?

> Any pattern which has sharp discontinuities from 0 to 1 would work, for example a checkboard, step function left-right or up-down, a single "speck" pixel etc. High frequency corresponds to highly varying signals that are not smooth.

(c) Suppose I downsample an image by a factor of 2. What is the highest frequency that this new image can represent in terms of the previous highest frequency h_f?

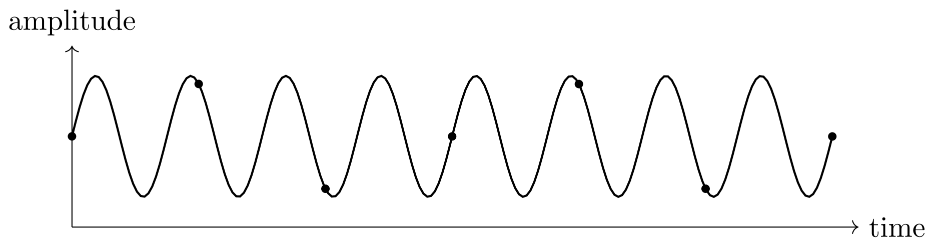
> The highest frequency (Nyquist frequency) is halved. According to the Nyquist theorem, the maximum frequency that can be represented is half the sampling rate. When you downsample by a factor of 2, you're reducing the sampling rate by half, so the maximum representable frequency also decreases by half. This is why anti-aliasing filters like blur are applied before downsampling to prevent high frequencies from being aliased into lower frequencies.

(d) Below is an image and its FFT transform. Find the corresponding images if we reconstructed it using inverse FFT after (a) masking out the inside of the red box and (b) masking the outside of the red box.



(a) masking the inside is a high-pass filter and would result in image #2, (b) masking the outside is a low-pass filter and would result in #1. "low-pass" means we take only the low frequencies which correspond to a blurry version of the image, while "high-pass" means we take only high frequencies, which emphasizes edges and sharp features.

Problem 2.2: ***Aliasing*** Consider this 1D signal sampled at regular intervals (black dots show sample points):



What lower-frequency signal would you reconstruct if you connected these sample points? Sketch it below:

Connect the dots, wow they look super different from the original signal!

Why might this be a problem when constructing image pyramids?

When sampling (resizing) images, aggressively resizing can result in these aliasing artifacts that destroy image information. You can also see this in real life when taking pictures of high-frequency details from afar like bricks or window screens. Inside image pyramids, downsizing too aggressively without careful antialiasing can result in wonky images (which you might have experienced in your Proj 1 if using too many pyramid levels).