Algorithm 6: Filter

November 19, 2021

Introduction to Computer Graphics, Fall 2021

Solution Key

Instructions: You may discuss this assignment with other students, but you must abide by the rules stated in the collaboration policy (no notes from any discussions, your handin must be completely written up by you and contain only your own work, 15 minute delay between discussions and writing down any notes of your own) and write the logins of the student(s) you collaborated with next to each problem. Hand in the assignment on Gradescope as a PDF (please make sure the document is anonymous) no later than the due date indicated above.

Be sure to run the signal processing applets (everything from Sampling to Filtering is relevant) before diving into this assignment: `https://cs.brown.edu/courses/cs123/demos.shtml`.

1 Duality of Domains

1.1 [1 point] The *dual* of the box-like function in Figure 1(a) in the frequency domain is the sinc-like function shown in Figure 1(b) in the spatial domain (note that the box unction shown need not have width 1, because there are no labels on the axes)

Sketch the dual of box-like function in Figure 2. Your drawing doesn't need to be perfectly accurate; we just need to see that you get the idea.

Solution:

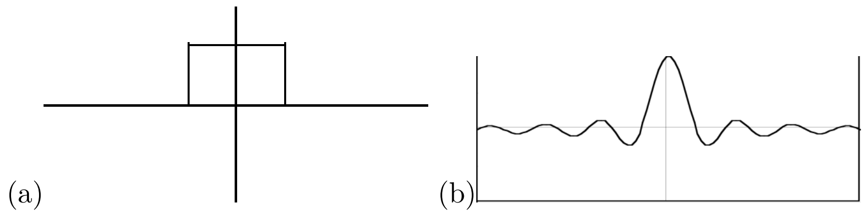


Figure 1: A box-like function and its sinc-like dual

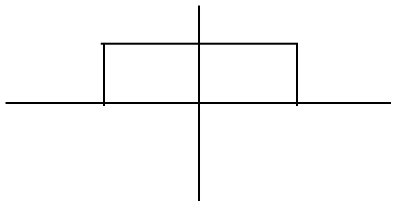
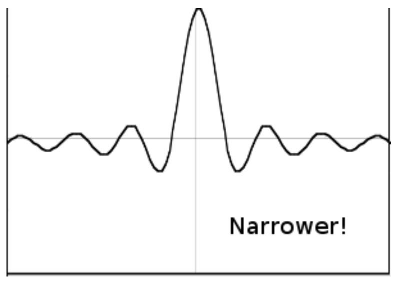


Figure 2: A box-like function, about twice as wide as the one in Figure 1(a).



(Hint: Think about the graphs of sin(x) and sin(2x))

1.2 [2 points] What do we usually use to approximate the sinc function, and why do we make this approximation when translating these theoretical concepts into code?

Solution: We use a triangle or gaussian filter. The sinc function has an infinite support width which makes it impractical in the real world (O(n4)). Triangle and gaussian are a pretty good approximation.

A *pixel*, as we defined it in the Brush assignment, was a colored square at a specific point on the image canvas.

1.3 [1 point] How has our notion of a pixel changed since Brush?

Solution: In Filter, we consider a pixel to be a discrete sample of a continuous function.

2 Convolution

For the following problems, consider the following function and filter. Remember that convolution is integration of the product of a function and a filter, with the filter reflected about the vertical axis; in formulas,

$$(f \star g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)\ dt \tag{1}$$

If the interval where f is nonzero (its *support*) is finite, and similarly for g, there may be values of x for which the two factors of the integrand in Equation 1 are never simultaneously nonzero, i.e., the supports of the two factors don't overlap at all. For other values of x, the supports might overlap completely. Or they might overlap only in a single point. This problem is all about exploring those possibilities. We'll do that using a function and a filter (playing the roles of f and g in Equation 1) shown in Figure 3.

2.1 [1/2 point each] What is the value of the function convolved with the filter at...

a. $x = 2.5$ **0**

b. $x = 2.25$ **0.25**

c. $x = 1.75$ **0.75**

d. $x = 1.25$ **1**

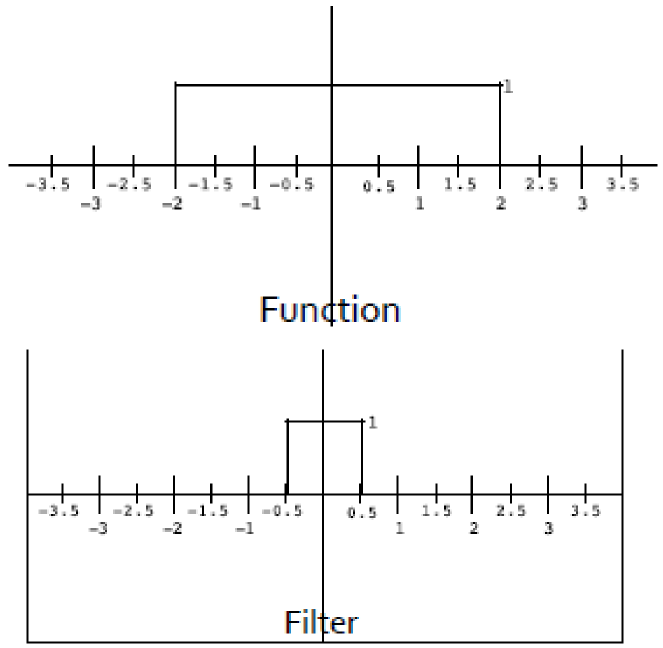
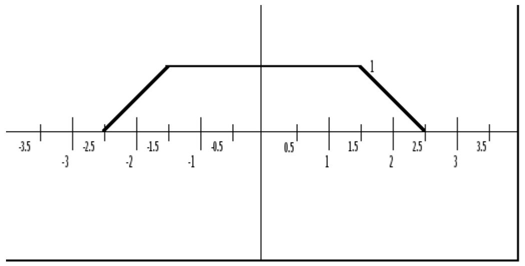


Figure 3: The functions for Problem 2.1

2.2 [1 point] Draw the function convolved with the filter on a blank graph (with labelled axes).



Let *F(x)* and *G(x)* be the frequency domain duals of spatial domain functions *f(x)* and *g(x)*, respectively (unrelated to those of the previous problem). Note that $\star$ is the convolution operator.

2.3 [1 point] What is the dual of $f \star g$?
Your answer should involve F and G

Solution: $F(x) \cdot G(x)$.

3 Prefiltering

3.1 [1 point] If we're sampling a continuous function at a frequency of n samples per unit, what is the largest frequency we can represent, according to the **Nyquist limit**?

Solution: just less than n/2

Now, examine the frequency plot of the infamous Mandrill Figure 4.

If you were going to sample this signal at a rate of 8 samples per unit, to avoid aliasing you would use what we know about the **Nyquist limit** to prefilter it.

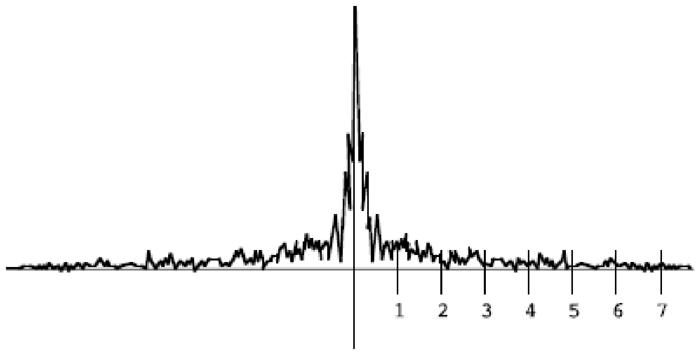
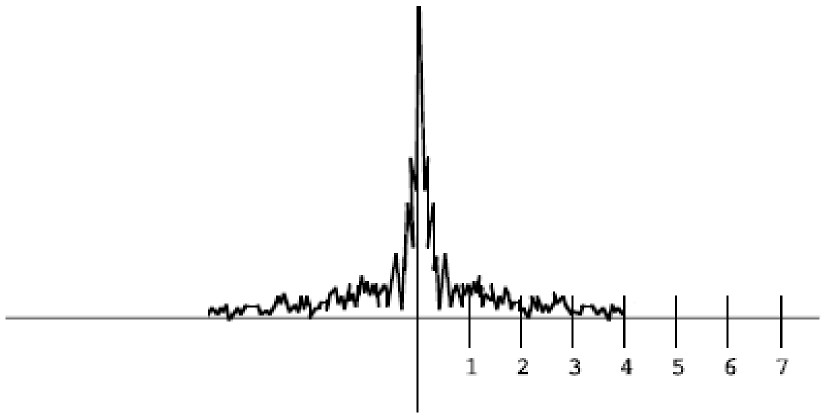


Figure 4: The frequency spectrum of the mandrill image

3.2 [1 point] Sketch the new frequency plot after someone has performed this prefiltering step optimally.
Solution:



4 Blur

4.1 [1 point] What is ***the effect*** of an ideal blur (anti-aliasing) filter in the ***frequency*** domain?

Solution: In the frequency domain, for some frequency n, frequencies greater than n samples per unit are removed (zeroed).

4.2 [1 point] What ***is*** the ideal blur filter in the ***spatial*** domain?

Solution: Sinc is the ideal blur filter in the spatial domain. To zero out frequencies greater than n samples per unit, in the frequency domain we multiply (not convolve) with a box function of width n/2. Since the dual of the box function is sinc, we convolve with a sinc filter in the spatial domain.

Now we're going to perform a blur, manually, in the spatial domain. This is how you're probably going to want to implement blur in the Filter assignment, because it's the easiest.

Let's perform a 1-dimensional blur of radius 2 on a row of an image. Assume zero-based indexing into the kernel and image arrays.

4.3 [2 points] in the 1-dimensional image, which is represented by an array `f` of floats, consider a point at index x in the image array. Write the summation equation for the color at point x in the new, blurred image. You can ignore edge cases for this question. Keep in mind that the kernel width is actually 5, so the center point of the filter array `g` is at `g[2]` (why?).

Solution: Two equivalent solutions are provided (other equivalent solutions accepted).

$$blurredImage(x) = \sum_{i=0}^{4} image[x + (i-2)] \cdot kernel[i]$$

$$blurredImage(x) = \sum_{i=x-2}^{x+2} image[i] \cdot kernel[i + 2 - x]$$

4.4 [2 points] How will you handle cases in your summation where the blur radius extends past the edge of the image (see previous question)? There are multiple correct solutions to this. Be sure to explain how you will keep the image's average brightness constant.

Solution: One way to handle this is to use pixel values at the edge of the image for pixel values beyond (image[0] for image[< 0] and image[max] for image[> max]). Another way to handle this is to bound the summation inside the image. This has the effect of "cutting" off the part of the filter outside the image. In this case, since the area under the filter no longer sums to one, you must normalize with the area under the cut filter lest the image darken.

5 Scaling [OPTIONAL]

5.1 [1 point] What is the filter support width when you scale (up) by a factor of $n > 1$? **2**

5.2 [1 point] What is the filter support width when you scale (down) by a factor of $0 < n < 1$? $\mathbf{\frac{2}{n}}$

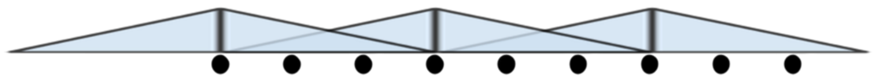
Recall that back-mapping refers to finding the correct filter placement given a pixel in the output image.

The naive back-map function is$f(x)_{naive} = \frac{x}{a}$ but the correct back-map function is $f(x) = \frac{x}{a} + \frac{1-a}{2a}$.

Suppose we wanted to scale **down** a 9 pixel (1D) image by a factor of 3 ($a = \frac{1}{3}$).

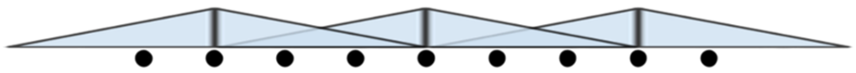
5.3 [1 point] Draw a picture to show where we would sample the original image using the naive back-map. Above each sample point include an illustration of the filter.

Solution:

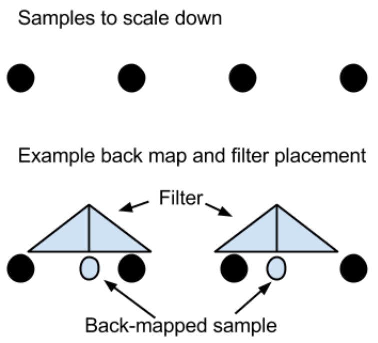


5.4 [1 point] Draw another picture to show where we would sample the original image using the correct back-map. Above each sample point include an illustration of the filter.

Solution:

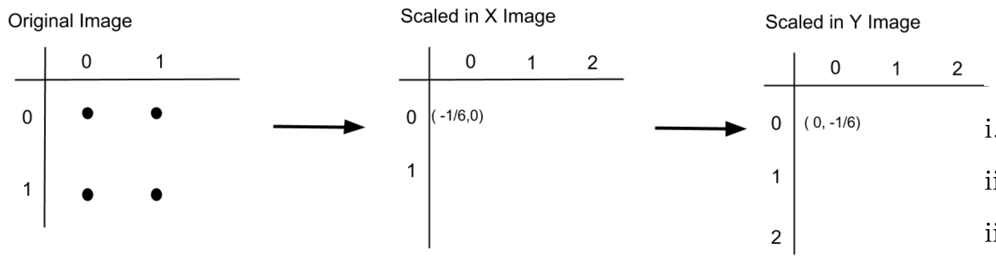


If you are confused about how and where to draw a filter, look at the example below.



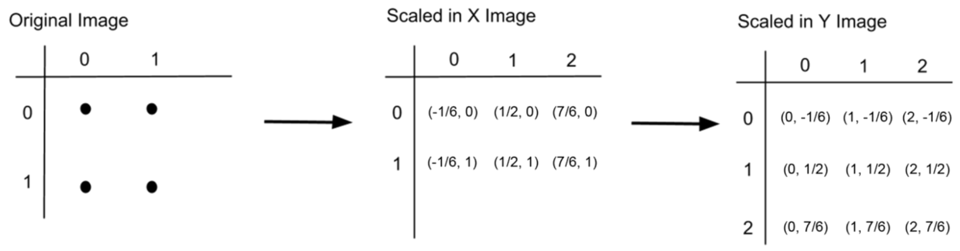
The first graphic below represents the upper left corner of an image. The numbers on the border are the pixel row and column indices. We want to scale this image up by a factor of 1.5.

The second graphic represents the image scaled in the x-direction, and the third image represents the second one scaled in the y-direction. The (x, y) pairs show where the filter must be centered on the previous image in order to calculate the pixel values in the scaled image.

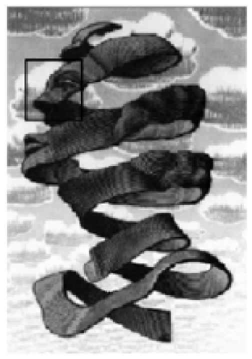


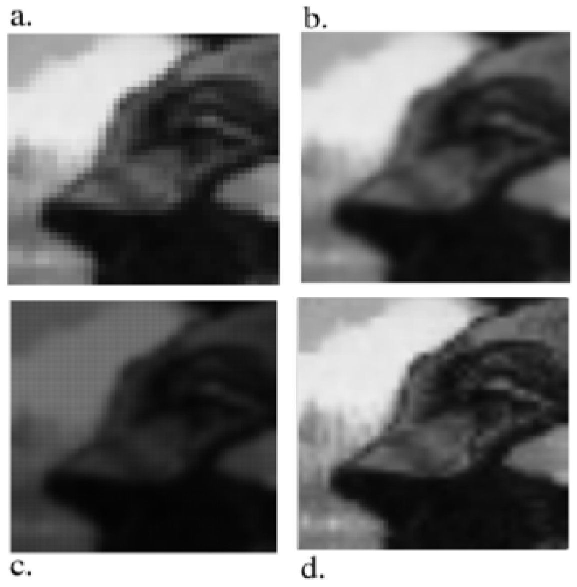
5.5 [2 points] Calculate the rest of the (x, y) pairs above. We've given you the upper left one.

Solution:



5.6 [$\frac{1}{2}$ point each] The following are four different scalings of an image. One was scaled using a Gaussian filter, another using a linear (box) filter, another using a triangle filter, and another using a very badly designed filter. Which is which?





i. Gaussian **B**

ii. Linear **A**

iii. Triangle **D**

iv. Bad **C**