Algorithm 3 Filter
Introduction to Computer Graphics, Fall 2015

Solution Key

1 Instructions

Write or type all of your answers on a separate sheet of paper and hand in to the cs123 hand-in bin. Late hand-ins will receive no credit. Do not ask a friend to hand in the assignment for you.

This assignment is worth 25% of your Filter grade.

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

2 Duality of Domains

[1 point] If the dual of the box function (a) in the frequency domain is the sinc function in the spatial domain (b),

(a)

(b)

sketch the dual of box function (c). Your drawing doesn’t need to be perfectly accurate; we just need to see that you get the idea.

(c)

(d)

[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(n^4)). Triangle and gaussian are a pretty good approximation.

[1 point] What is a pixel? How has the meaning of a pixel changed since the Brush assignment?

Solution: A bitmap pixel, as we defined it in the Brush assignment, is a discrete color (little square) at a specific point on the image canvas. In filter, we consider a pixel to be a discrete sample of a continuous function.

3 Convolution

For the following problems, consider the following function and filter.

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

a. \( x = -1.5 \)

b. \( x = -2 \)

c. \( x = -2.25 \)

d. \( x = -2.5 \)

[1 point] Draw the function convolved with the filter on a blank graph.

Let \( F(x) \) and \( G(x) \) be the frequency domain duals of spatial domain functions \( f(x) \) and \( g(x) \), respectively. Fill in the right hand side of the following equation with...
an equivalent operation involving \( F(x) \) and \( G(x) \). Note that * is the convolution operator.

**[1 point]** What is the dual of \( f(x) \ast g(x) \)?

**Solution:** \( F(x) \cdot G(x) \)

### 4 Prefiltering

**[1 point]** If we’re sampling at a frequency of \( n \) samples per unit, what is the largest frequency we can represent, according to the **Nyquist limit**?

**Solution:** \( n/2 \)

Now, examine the frequency plot of the infamous Mandrill.

![Frequency plot of Mandrill](image)

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.

**[1 point]** Sketch the new frequency plot after this prefiltering step.

**Solution:**

![New frequency plot](image)

### 5 Blur

**[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).

**[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.

**[2 points]** At a particular point \( x \) along the image, 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.

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

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

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

**[2 points]** How will you handle edge cases in your inner loop (previous question)? There are multiple correct solutions to this. Be sure to explain how you will keep the image’s lightness 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.

### 6 Scaling

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

**Solution:** \( 2 \)

**[1 point]** What is the filter support width when you scale (down) by a factor of \( 0 < n < 1 \)?

**Solution:** \( 2/n \)

**[1 point]** When scaling up and down, when do you and when don’t you have to normalize the filtered pixel values?
Solution: Regardless of our scale factor, the area of our filter function is 1. However, when the support width is not an integer, the sum of the weights of the filter varies as a function of the sampling position. As a result, you don’t need to normalize when scaling up (width=2), but you often do need to normalize when scaling down (width=2/n).

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

The naive back-map is \( f(x)_{naive} = \frac{x}{a} \) and our 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=1/3).

[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:

[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:

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

Solution:

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

Samples to scale down

![Filter Placement Example](image)

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.

![Example Back Map and Filter Placement](image)

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.

![Calculated (x,y) Pairs](image)

[\frac{1}{2} \text{ point each}] The following are four different scalings of an image. One was scaled using a Gaussian filter, another using a linear (constant) 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