Homework 2

Due 11:55pm, Tuesday Feb 4th

If we want to perform the large shrinking operations that we talked about in the last homework we need to first smooth our image. We'll start out by filtering the image with a box filter. There are very fast ways of performing this operation but instead, we'll do the naive thing and implement it as a convolution because it will generalize to other filters as well!

2.1 Image filtering

Image filtering (or convolution) is a fundamental image processing tool. See chapter 3.2 of Szeliski and the lecture materials to learn about image filtering (specifically linear filtering). Numpy has numerous built in and efficient functions to perform image filtering, but you will be writing your own such function from scratch for this assignment. More specifically, you will implement **cross_correlation_2d**, followed by **convolve_2d** which would use **cross_correlation_2d**.

2.2 Implement a Gaussian kernel

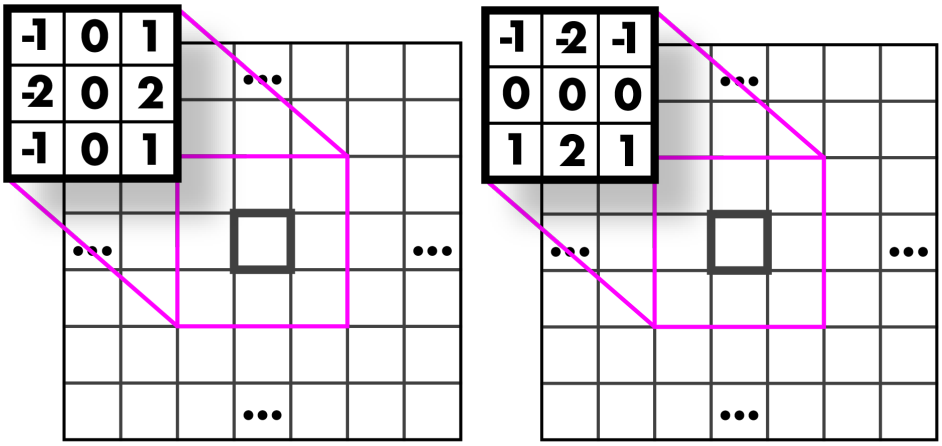
As you have seen in the lectures, there are a few different ways to blur an image, for example taking a weighted average of the neighboring pixels. Gaussian blur is a special kind of *weighted* averaging of neighboring pixels, and is described in the lecture slides. To implement Gaussian blur, you will implement a function **gaussian_blur_kernel_2d** that produces a kernel of a given *height* and *width* which can then be passed to **convolve_2d** from above, along with an image, to produce a blurred version of the image.

2.3 Image shrinking

Now using the Gaussian Kernel and convolution implemented in 2.1 and 2.2 to smooth the image first and then perform the image shrinking. Compare the direct image shrinking with blur + image shrinking.

2.4 Sobel filters

We can use sobel filters to estimate the gradients and direction of those gradients in an image. They should be straightforward now that you all are such pros at image filtering. First implement the functions to make our sobel filters. They are for estimating the gradient in the x and y direction:



2.5 Calculate gradient magnitude and direction

Fill in the function `sobel_image(im)`. It should return three images, the gradient magnitude and the derivative image in x and y direction. The strategy can be found here. Normalize the images before visualization using feature normalization.