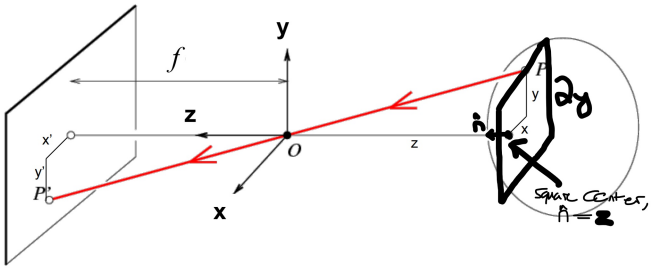
COS 429 Assignment 1

Preeti Iyer and Matthew Hetrick

December 14, 2019

Question 1: Pinhole Camera Model

Question: Consider a pinhole camera with focal length f. Let this pinhole camera face a whiteboard, in parallel, at a distance L between the whiteboard and the pinhole. Imagine a square of area S drawn on the whiteboard. What is the area of the square in the image? Justify your answer.
Solution: From lecture, we used the equation based on Figure 1 below, where we imagine P as the center of the top side of the square on the whiteboard.



- Using similar triangles: $\frac{x}{z} = \frac{x'}{f}$ $\frac{y}{z} = \frac{y'}{f}$

Figure 1: Edited Image from Cos 429 Lecture 2, slide 22

Assuming that the square is drawn on the whiteboard with side length 2y and the normal aligned with the axis of the pinhole, then we can use

$$\frac{y}{z} = \frac{y'}{f} \tag{1}$$

to determine the side length s. In this case, L is the length z from the screen, $s' = (2y)^2$ (aka $y = \frac{\sqrt{s}}{2}$), and $s' = (2y')^2$. Plugging $y = \frac{\sqrt{s}}{2}$ into equation (1) and multiplying by f, we get:

$$y' = \frac{f\sqrt{s}}{2L} \tag{2}$$

Plugging equation (2) into $s' = (2y')^2$, we get s', the area of the square in the image:

$$\begin{aligned} s' &= (2f\frac{\sqrt{s}}{2L})^2 = (f\frac{\sqrt{s}}{L})^2 \\ s' &= \frac{sf^2}{L^2} \end{aligned} \tag{3}$$

Question 2: Linear Filters

$$I = \begin{bmatrix} -1 & 0 & 2 \\ 1 & -2 & 1 \end{bmatrix}, F = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

2.a

Question: Convolve the 2x3 matrix $I = [1, 0, 2; 1, 2, 1]$ with the 3x3 matrix $F = [1, 1, 1; 1, 1, 1; 0, 0, 0]$. Use zero-padding when necessary. The output shape should be 'same' (same as the 2x3 matrix I).
Solution: We need to flip the matrix F to perform the convolution:

$$F_{inverted} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

If we convolve using same-padding so that we output a 2x3 matrix, our resulting matrix has the following equations:

$$\begin{aligned}(I * F)[1,1] &= I[1,1] * F[2,2] + I[1,2] * F[2,3] + I[2,1] * F[3,2] + I[2,2] * F[3,3] \\ &= -1 + 0 + -1 + 2 = \mathbf{0}\end{aligned} \tag{4}$$

$$\begin{aligned}(I * F)[1,2] &= I[1,1] * F[2,1] + I[1,2] * F[2,2] + I[1,3] * F[2,3] + I[2,1] * F[3,1] \\ &+ I[2,2] * F[3,2] + I[2,3] * F[3,3] \\ &= -1 + 0 + 2 - 1 + 2 - 1 = \mathbf{1}\end{aligned} \tag{5}$$

$$\begin{aligned}(I * F)[1,3] &= I[1,2] * F[2,1] + I[1,3] * F[2,2] + I[2,2] * F[3,1] + I[2,3] * F[3,2] \\ &= 0 + 2 + 2 - 1 = \mathbf{3}\end{aligned} \tag{6}$$

$$\begin{aligned}(I * F)[2,1] &= I[1,1] * F[1,2] + I[1,2] * F[1,3] + I[2,1] * F[2,2] + I[2,1] * F[2,3] \\ &= 0 + 0 + 1 - 2 = \textbf{-1}\end{aligned} \tag{7}$$

$$\begin{aligned}(I * F)[2,2] &= I[1,1] * F[1,1] + I[1,2] * F[1,2] + I[1,3] * F[1,3] + I[2,1] * F[2,1] \\ &+ I[2,2] * F[2,2] + I[2,3] * F[2,3]7 = 0 + 0 + 0 + 1 - 2 + 1 = \mathbf{0}\end{aligned} \tag{8}$$

$$\begin{aligned}(I * F)[2,3] &= I[1,2] * F[1,1] + I[1,3] * F[1,2] + I[2,2] * F[2,1] + I[2,3] * F[2,2] \\ &= 0 + 0 - 2 + 1 = \textbf{-1}\end{aligned} \tag{9}$$

This gives us the convoluted matrix:

$$I * F = \begin{bmatrix} 0 & 1 & 3 \\ -1 & 0 & -1 \end{bmatrix}$$

Note that we did not write the zero padding in the equations for clarity and space sake, as they can be considered zero in the operation. We effectively used the zero padding to eliminate non-overlapping matrix entries from the equations. We will do this for part (b) as well.

2.b

Question: Note that F is separable, i.e., it can be written as a product of two 1D filters: $F_1 = [1; 1; 0]$ and $F_2 = [1, 1, 1]$. Compute $(I * F_1)$ and $(I * F_1) * F_2$, i.e., first perform 1D convolution on each column, followed by another 1D convolution on each row.

Solution: Invert F_1 and F_2:

$$F_1 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, F_1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Now convolving F_1 and I with $[i, j] = [row, column]$:

$$(I * F_1)[1, 1] = I[1, 1] * F_1[2] + I[2, 1] * F_1[3] = -1 - 1 = -2 \tag{10}$$

$$(I * F_1)[1, 2] = I[1, 2] * F_1[2] + I[2, 2] * F_1[3] = 0 - (-2) = 2 \tag{11}$$

$$(I * F_1)[1, 3] = I[1, 3] * F_1[2] + I[2, 3] * F_1[3] = 2 - 1 = 1 \tag{12}$$

$$(I * F_1)[2, 1] = I[1, 1] * F_1[1] + I[2, 1] * F_1[2] = 0 + 1 = 1 \tag{13}$$

$$(I * F_1)[2, 2] = I[1, 2] * F_1[1] + I[2, 2] * F_1[2] = 0 - 2 = -2 \tag{14}$$

$$(I * F_1)[2, 3] = I[1, 3] * F_1[1] + I[2, 3] * F_1[2] = 0 + 1 = 1 \tag{15}$$

This gives the matrix

$$I * F_1 = \begin{bmatrix} -2 & 2 & 1 \\ 1 & -2 & 1 \end{bmatrix}$$

Now convolving this matrix with F_2:

$$((I * F_1) * F_2)[1, 1] = (I * F_1)[1, 1] * F_1[2] + (I * F_1)[1, 2] * F_1[3] = -2 + 2 = 0 \tag{16}$$

$$((I*F_1)*F_2)[1, 2] = (I*F_1)[1, 1]*F_1[1]+(I*F_1)[1, 2]*F_1[2]+(I*F_1)[1, 3]*F_1[3] = -2+2+1 = 1 \tag{17}$$

$$((I * F_1) * F_2)[1, 3] = (I * F_1)[1, 2] * F_1[1] + (I * F_1)[1, 3] * F_1[2] = 2 + 1 = 3 \tag{18}$$

$$((I * F_1) * F_2)[2, 1] = (I * F_1)[2, 1] * F_1[2] + (I * F_1)[2, 2] * F_1[3] = 1 - 2 = -1 \tag{19}$$

$$((I*F_1)*F_2)[2,2] = (I*F_1)[2,1]*F_1[1]+(I*F_1)[2,2]*F_1[2]+(I*F_1)[2,3]*F_1[3] = 1-2+1 = 0 \tag{20}$$

$$((I*F_1)*F_2)[2,3] = (I*F_1)[2,2]*F_1[1]+(I*F_1)[2,3]*F_1[2] = -2+1 = -1 \tag{21}$$

This outputs the matrix $(I*F_1)*F_2 = \begin{bmatrix} 0 & 1 & 3 \\ -1 & 0 & -1 \end{bmatrix}$ which we see is equivalent to the original matrix $(I*F_1)$.

2.c

Question: Prove that for any separable filter $F = F_1F_2$:

$$I*F = (I*F1)*F2 \tag{22}$$

Hint: expand the 2D convolution equation directly.

Solution: By definition, $(I*F)[i,j] = \sum_{k,l} I[i-k,j-l]F[k,l] = \sum_{k,l} F[i-k,j-l]I[k,l]$ by commutation of convolution. Given that $F[i,j] = F_2[i]F_1[j](*)$ and expanding this for $k_1,k_2,...,k_n$ and $l_1,l_2,...,l_n$:

$$\begin{aligned}(I*F)[i,j] = F[i-k_1,j-l_1]I[k_1,l_1] + F[i-k_1,j-l_2]I[k_1,l_2] + ... + F[i-k_1,j-l_n]I[k_1,l_n] \\ +F[i-k_2,j-l_1]I[k_2,l_1] + F[i-k_2,j-l_2]I[k_2,l_2] + ... + F[i-k_2,j-l_n]I[k_2,l_n] + ... \\ +F[i-k_n,j-l_n]I[k_n,l_n]\end{aligned} \tag{23}$$

Now grouping this by k_i's,

$$(I*F)[i,j] = \sum_l F[i-k_1,j-l]I[k_1,l]+\sum_l F[i-k_2,j-l]I[k_2,l]+...+\sum_l F[i-k_n,j-l]I[k_n,l] \tag{24}$$

Using the definition of separability equation $(*)$ and factoring out the constants from the sums:

$$\begin{aligned}(I*F)[i,j] = F_2[i-k_1]\sum_l F_1[j-l]I[k_1,l] + F_2[i-k_2]\sum_l F_1[j-l]I[k_2,l] + ... \\ +F_2[i-k_n]\sum_l F_1[j-l]I[k_n,l]\end{aligned} \tag{25}$$

This yields:

$$(I*F)[i,j] = \sum_k F_2[i-k]\sum_l F_1[j-l]I[k,l] = \sum_k F_2[i-k](F_1*I)[k,l] \tag{26}$$

which in matrix notation is $(I*F) = F_2*(F_1*I)$.
Using the commutation property again, we get

$$(I*F) = F_2*(F_1*I) = F_2*(I*F_1) = (I*F_1)*F_2 \tag{27}$$

which satisfies our proof.

Question 3: Difference of Gaussian Detector

3.a

Question: Recall that a 1D Gaussian is:

$$g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{x^2}{2\sigma^2}) \tag{28}$$

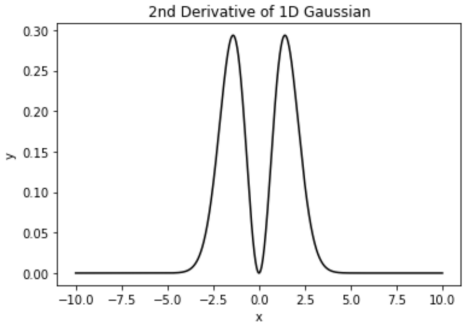
Calculate the 2^{nd} derivative of the 1D Gaussian with respect to x and plot it in Python (use =1). Submit all steps of your derivation and the generated plot in the PDF file you turn in. Hint: Create a large number of x using np.linspace, and get function outputs from those.
Solution: Let $u = -\frac{x^2}{2\sigma^2}$. We then get

$$g'_\sigma(x) = -\frac{x}{\sqrt{2\pi}\sigma^3} exp(-\frac{x^2}{2\sigma^2}) \tag{29}$$

Now to take the derivative of this, use the product rule and simplify:

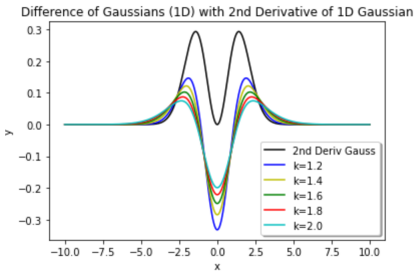
$$\begin{aligned}
g'_\sigma(x) =& (-\frac{x}{\sigma^2})(\frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{x^2}{2\sigma^2})) \\
=& -\frac{x}{\sigma^2}\frac{-x}{\sqrt{2\pi}\sigma^3} exp(-\frac{x^2}{2\sigma^2}) + \frac{-1}{\sqrt{2\pi}\sigma^3} exp(-\frac{x^2}{2\sigma^2}) \\
=& \frac{exp(-\frac{x^2}{2\sigma^2})}{\sqrt{2\pi}\sigma^3}(\frac{x^2}{\sigma^2} + 1) \\
=& \frac{exp(-\frac{x^2}{2\sigma^2})}{\sqrt{2\pi}\sigma^5}(x^2 - \sigma^2) = \frac{exp(-\frac{x^2}{2})}{\sqrt{2\pi}}(x^2) \qquad for \sigma = 1
\end{aligned}$$

Plotting this in python:



3.b

Question: Use Python to plot the difference of Gaussians in 1D given by $(x, \sigma, k) = \frac{g_{k\sigma}(x) - g_{\sigma}(x)}{k\sigma - \sigma}$ using k = 1.2, 1.4, 1.6, 1.8, 2.0. State which value of k gives the best approximation to the 2^{nd} derivative with respect to x. Assume =1.
Solution: See code attached at end of pdf and output image directly below.



Visually, it appears as if k = 1.2 gives the best approximation outside of the interval (-1.5,1.5). Inside this interval, k = 2.0 appears to give the best approximation.

3.c

Question: The 2D equivalents of the plot above are rotationally symmetric. To what type of image structure will the difference of Gaussian respond maximally?
Solution: The difference of Gaussian acts as a low-pass band filter of sorts with respect to intensity as a function of space. Thus it removes high-frequency components from the image, essentially blurring parts of the image that are more similar. It therefore highlights the edges in the resulting image, but as a result also decreases the overall contrast of the image because of the blurring of non-distinct features. So, images with high amounts of noise and softly-defined edges with minimal contrast to begin with will be optimally filtered by the difference of Gaussian.

Question 4

Once we developed our algorithm, we iterated through possible variations of sigma, T_h, and T_l to find the optimal edge detection and tested on gray scale and float images. We based our intuition for experimenting based on the theory and mathematics behind the algorithm: here sigma dictates the scale of the objects being simplified, a smaller Gaussian filter would cause less blurring (allowing the ability to detect smaller lines) whereas a larger filter which causes more blurring would allow for detecting smoother and larger edges (so ultimately used a sigma of 1 and kept sigma constant while tuning our lower and upper thresholds). Additionally when setting lower and upper thresholds we kept in mind that the edge pixels below the threshold are discarded and those above the upper limit are considered in an edge map. We tested several varieties in large ranges between our upper and lower thresholds, shifting both thresholds up, or shifting both thresholds down.

We looked at the normalized thin image matrix, I, and in order to have a successful edge capture that balanced thin lines and larger edges, we wanted to create a threshold where we were selective about what would be considered an edge but also give flexibility to consider enough potential pixels; when looking at I, we tried many iterations of what to consider, but found that around 25 percent above and below the mean value of the normalized thin image matrix gave us a good lower and upper threshold : where it was high enough to filter our noise and non-applicable pixels and low

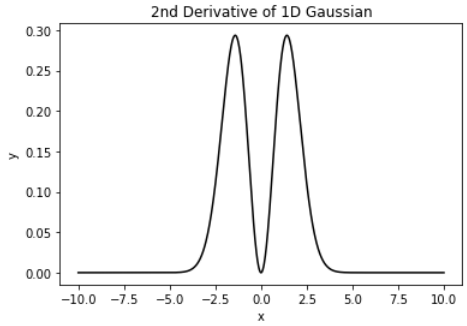
enough to consider many options. Ultimately the values that worked well for us were sigma = 1, lower threshold = .4, upper threshold = .6

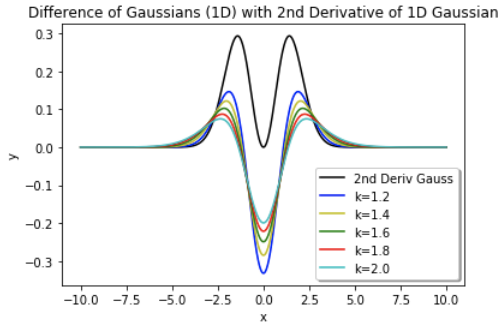
We tried a variety of images with varying edge detection capabilities; in addition to the test images provided to us on the assignment page, we also started with simple circular and square based images where the detection was simpler and tuning the parameters gave us a better understanding for the effect on edge detection. We then looked at images with a distinct foreground and background, ones with complex patterns and varying colors, and testing our initial intuition to see if our tuning still applied.

In [1]:
```
import numpy as np
import cv2
import matplotlib
import matplotlib.pyplot as plt
```

In [6]:
```
ks = [1.2, 1.4, 1.6, 1.8, 2.0]
x = np.linspace(-10, 10, 1000)
pi = np.pi
def gaussian(x, sig):
    return (1/(sig*np.sqrt(2*pi)))*np.exp(-(x**2)/(2*(sig**2)))
yprimeprime = (1/np.sqrt(2*pi))*(x**2)*np.exp(-(x**2)/2)
plt.figure(1)
plt.plot(x, yprimeprime, 'k', label='2nd Deriv Gauss')
plt.xlabel('x')
plt.ylabel('y')
plt.title('2nd Derivative of 1D Gaussian')
plt.figure(2)
plt.plot(x, yprimeprime, 'k', label='2nd Deriv Gauss')
colors = ['b', 'y', 'g', 'r', 'c']
labels = ['k=1.2', 'k=1.4', 'k=1.6', 'k=1.8', 'k=2.0']
for i in range(len(ks)):
    y = gaussian(x, 1)
    yk = gaussian(x, ks[i])
    D = (yk - y) / (ks[i]-1)
    plt.plot(x,D,colors[i], label=labels[i])
plt.legend(loc='lower right', shadow=True)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Difference of Gaussians (1D) with 2nd Derivative of 1D Gaussian')
```

Out[6]:
```
Text(0.5, 1.0, 'Difference of Gaussians (1D) with 2nd Derivative of 1D Gaussian')
```





In [ ]: