Class: Computer Vision
Assignment: HW01
Developer: Nathaniel Rose
Due Date: 02/20/20

Problem 01 Report:

We needed to implement a Gaussian smoothing (blurring) algorithm and apply it to an input image. Part a and c required 1d Gaussian Masks, and part b required a 2d Gaussian Mask. The overall process included creating a Gaussian mask with desired sigma, convolving the mask over the input image, and normalizing the pixels to get the final output image. We used sigma values of 1, 5, and 11 for the masks. The final output images were smoothed differently depending on the given sigma values of the Gaussian Mask and also ended up slightly darker around the edges because I padded the edges of the matrix with zeros to avoid an image disturbance.

1D Gaussian)

We needed to apply a 1d Gaussian Mask to a 1d image and then print out our results to a txt file. This smoothed out the values near the edges of the txt file which gives it a good Gaussian distribution. The 1d Gaussian Mask G(x) can be described as:

$$G(x) = \left(\frac{1}{\sqrt{(2\pi)}\sigma}\right)\left(\exp^{-\left(\frac{x^2}{2\sigma^2}\right)}\right)$$

2D Gaussian)

We needed to apply a 2d Gaussian Mask to a 2d image and then write our result to a .pgm output file. We needed to provide a 2d convolution between the mask and the input image and pad it with zeros around the edges. A 2d Gaussian Mask G(x,y) can be described as:

$$G(x,y) = \left(\frac{1}{(2\pi)\sigma}\right)\left(\exp^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}\right)$$

Output Images:

Sigma = 1

Sigma = 5

Sigma = 11

Separable Gaussian)

We needed to do the same thing as part b, but instead on 1 2d convolution we need to do 2 1d convolutions with 1d Gaussian Masks. So we follow the formula for part a, but we create a mask for the x pixels and the y pixels and convolve each of them with the input matrix separately. This reduces the time complexity of the convolution for a pixel and allows for parallel programming. The output images are not the exact same as part 2, but very similiar, and I am guessing this is due to rounding errors when normalizing the pixel values to integers.

$$G(x,y) = G(x) * G(y) = \left(\frac{1}{\sqrt{(2\pi)}\sigma}\right)\left(\exp^{-\left(\frac{x^2}{2\sigma^2}\right)}\right) * \left(\frac{1}{\sqrt{(2\pi)}\sigma}\right)\left(\exp^{-\left(\frac{y^2}{2\sigma^2}\right)}\right)$$

Output Images:

Sigma = 1

Sigma = 5

Sigma = 11