EECS 531 A1 Exercise 1

Alexander Telich

February 19, 2018

1 Image Blur

1.1 The Gaussian Kernel

To get the desired effect of blurring an image a kernel matrix must be convoluted with an equal size matrix of pixel color values based on patches taken from the original image. The 2D Gaussian function is defined in equation (1). Equation (2) shows the process of convolution. The matrix on the left being the kernel matrix and the matrix on the right, the matrix you are operating on.

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

where $G\left(x,\ y\right)$ is the gaussian transformation at point $\left(x,\ y\right)$

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right) [center] = (i \cdot 1)+(h \cdot 2)+(g \cdot 3)+(f \cdot 4)+(e \cdot 5)+(d \cdot 6)+(c \cdot 7)+(b \cdot 8)+(a \cdot 9) \tag{2}$$

where the left matrix is the kernel matrix, the right matrix is the matrix you are operating on, and [center] is the center coordinates of each matrix

2 Code

```
1 import cv2 as cv
2 import numpy as np
3
4 # Reads the image into a variable
5 img = cv.imread('Exercise1_Image.jpg', 0)
6
7
8 # Function to calculate the Gaussian transformation of a given point
9 def gaussian(x, y, sigma):
10     g = np.multiply(np.divide(1, 2 * np.pi * np.power(sigma, 2)), np.exp(
11         np.divide(-(np.power(x, 2) + np.power(y, 2)),
12                   np.multiply(2, np.power(sigma, 2)))))
13     return g
14
```

```
15
16 # Function to calculate the kernel matrix of an inputted size
17 # x: width, y: height
18 def kernel(x1, y1, sigma):
19     w, h = x1, y1 - 1
20     w2 = w / 2
21     h2 = h / 2
22     matrix = []
23     for y in range(int(h2), int(-h2 - 1), -1):
24         matrix.append([gaussian(x, y, sigma) for x in range(int(w2), int(-w2 -
       1), -1)])
25     return np.array(matrix)
26
27
28 # Function to convolude kernel matrix and image pixel matrix
29 def convolution(kernelMatrix, image):
30     (imageHeight, imageWidth) = image.shape[:2]
31     (kernelHeight, kernelWidth) = kernelMatrix.shape[:2]
32
33     # Generates a padding based on kernel matrix width minus 1 and divided by 2
34     padding = int(np.divide(kernelWidth - 1, 2))
35
36     # Makes a border around the image with the size of the padding
37     image = cv.copyMakeBorder(image, padding, padding, padding, padding,
38                               cv.BORDER_REPLICATE)
39     imageArray = np.array(image)
40
41     # Convolution of the kernel matrix and the image pixel matrix
42     for i in np.arange(padding, imageHeight + padding):
43         for j in np.arange(padding, imageWidth + padding):
44             patch = image[i - padding:i + padding + 1, j - padding:j + padding
       + 1]
45
46             convolve = (patch * kernelMatrix).sum()
47
48             imageArray[i - padding, j - padding] = convolve
49
50     return imageArray
51
52
53 imageBlurred = convolution(kernel(3, 3, 0.84089642), img)
54
55 cv.imshow('image', imageBlurred)
56 cv.imwrite('Exercise1_OutputImage.png', imageBlurred)
57 cv.waitKey(0)
```

Figure 1: Original Image

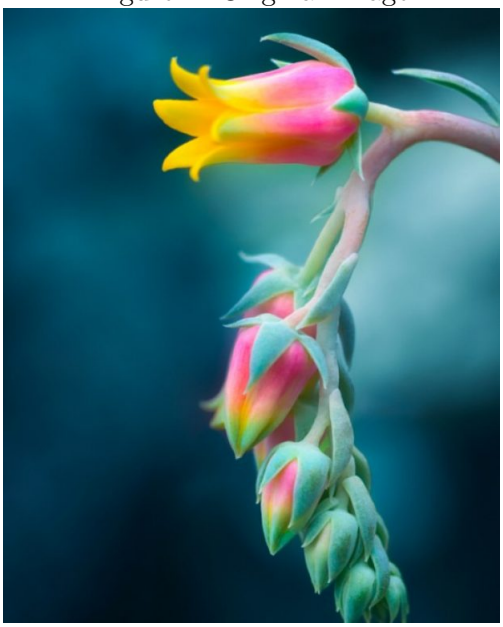

Figure 2: Image After Gaussian Blur Applied