A0 - Introduction: Playing with Pictures

Andrew Parmar
CS6475 Fall 2022
aparmar32@gatech.edu

I. My Image

Fig. 1. Original image. Stationary Art.

Exposure Time	Aperture	ISO
1/120	f/1.6	80

II. Filter Information

I chose to use a box filter with k=3 giving a kernel of shape (HxW) 7x7. The filter will blur the image by summing the values under the kernel and dividing by the number of cells in the kernel i.e 49. As convolution is associative, instead of dividing the final sum by 49, I am including the division by 49 in my filter itself. The final filter then is 7x7 matrix with 1/49 as each cell's value as shown below.

1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49
1/49	1/49	1/49	1/49	1/49	1/49	1/49

III. Convolution Results

Visually the results of my manual and cv2 convolution functions appear identical. Both output images show the a blurry version of the original.

For both manual and cv2 assisted convolution functions the numerical outputs are the same. I compared the resulting output by testing for equality and the matrix values are identical.

The following helper method was useful in determining that values at each cell location in the two matrices are identical.

Fig. 2. convolveManual. Blur using Box filter of size k=3.

Fig. 3. convolveCV2. Blur using Box filter of size k=3.

```
image = cv2.imread("images/source/image.png")
my_filter = myFilter()
img_manual = convolutionManual(image, my_filter)
img_cv2 = convolutionCV2(image, my_filter)
assert np.all(img_manual[:,:,:] == img_cv2[:,:,:])
```

Slicing a small arbitrary region of the two images also shows identical values. This method was more helpful while developing my functions as it showed my functions differed in most cases by a one-off value. This suggested that there was some difference in the way the functions were rounding out the final output.

```
r, c = 115, 336
img_manual[r:r+5,c:c+5,0]
Out[4]:
array([[26, 25, 25, 27, 29],
       [28, 25, 25, 26, 28],
       [29, 26, 25, 25, 27],
       [30, 26, 24, 23, 25],
       [30, 26, 23, 22, 23]], dtype=uint8)
img_cv2[r:r+5,c:c+5,0]
Out[5]:
array([[26, 25, 25, 27, 29],
       [28, 25, 25, 26, 28],
       [29, 26, 25, 25, 27],
       [30, 26, 24, 23, 25],
       [30, 26, 23, 22, 23]], dtype=uint8)
```

IV. Reflection

Converting the images passed into the various functions to $dtype = np.float64$ will be the first thing I will do going forward. Not doing it this way introduced a lot of issues with rounding.

My original filter was a 3x3 box filter. While this worked numerically, it wasn't clear enough when I put my images in the report. I had to increase the size of the filter to blur my source image enough that the convolution results actually showed a difference. In hindsight choosing a gradient filter would have been much more visually discernible. Furthermore not using a symmetric filter would have revealed issues related to correlation vs convolution quicker. With a symmetric filter both correlation and convolution produce the same result so it wasn't directly clear if my implementation was indeed convolution and not correlation.

Lastly I would have started with the convolutionCV2 method first as this was easier to implement and would have provided a ground truth reference for the development of the manual function.

References

[1] Richard Szeliski, Computer Vision - Algorithms and Applications, Springer, 2022, Image Filtering, 95.

[2] Kimberly Piyawan Sirichoke, Assignment 0, ED post and comments #6.

[3] OpenCV, filter2D, filter2D, Accessed: Aug 28, 2022