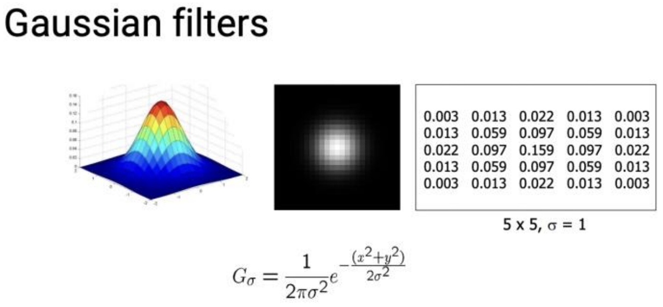
Ruchita Paithankar
CS583
Homework 1

- **split(img)**
  1. figuring this out was the most straightforward thing after the merge(r,g,b) function.
  2. It is basically splitting the array depth-wise into 3 channels – R, B, G and squeezing them into an 1D array.
- create_gaussian_kernel(size,sigma=1.0)
  1. To compute a 2D gaussian kernel I iterated the loops from -size/2 to size/2 so that the center x,y is 0,0.
  2. Then I used the given formula for calculating Gausian kernel.
  3. Tried normalization by summing the values of 'rv'.
  4. I got a little stuck her as I was not clear on the concepts at first. I referred to a few concepts from the week 2 slides.



Gaussian filters

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Like in this image the professor talked about the value of sigma adding up to 1.

  5. Initially I forgot to divide the kernel by the sum of its values.

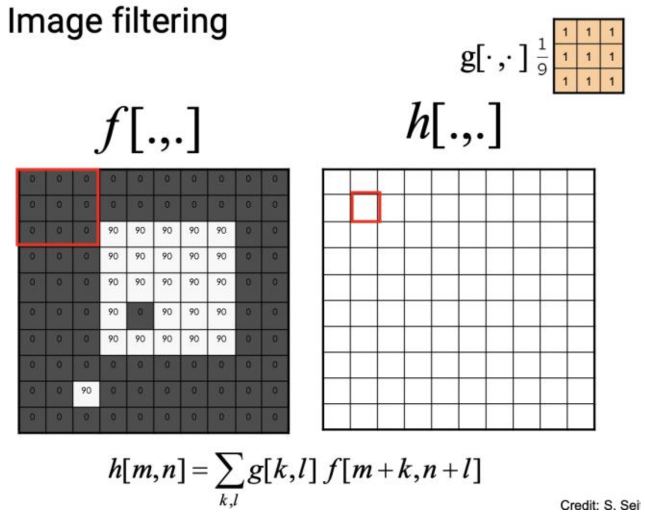


Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

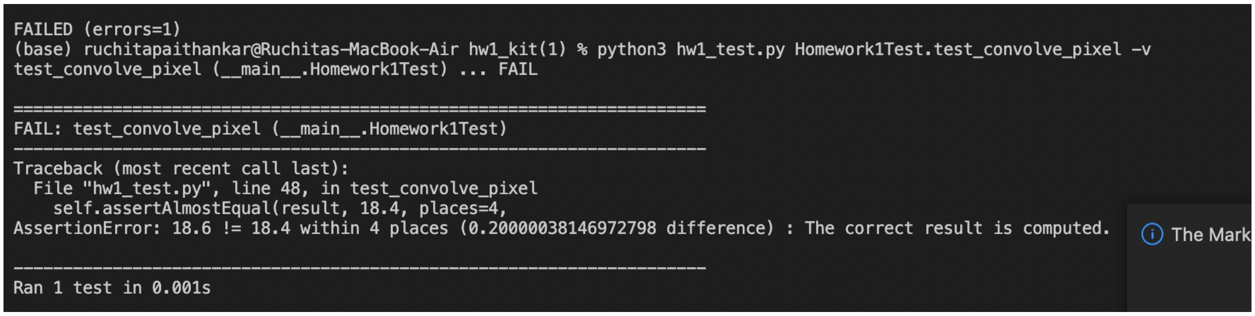
$h[.,.]$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Credit: S. Sei

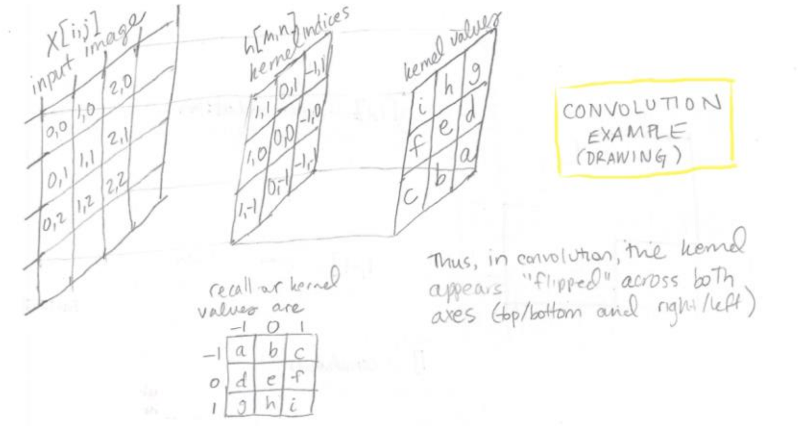
Again, I had to brush up some things conceptually. This box filter slide from week 2 is what I referred to.

- **convolve_pixel(img, kernel, i, j)**
  1. this was where I got stuck the most. Basically, figuring out the condition for when the kernel would land outside the image box, and then the answer I was initially 18.6 instead of 18.4

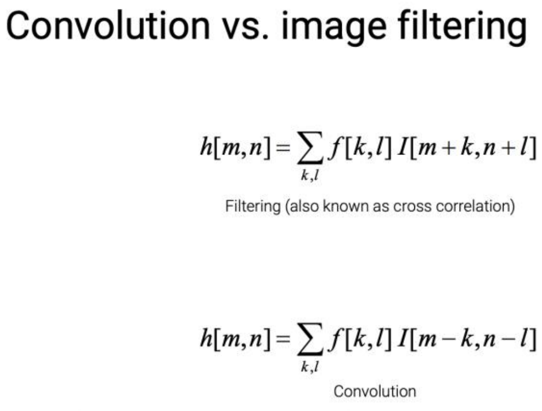


```
FAILED (errors=1)
(base) ruchitapaithankar@Ruchitas-MacBook-Air hw1_kit(1) % python3 hw1_test.py Homework1Test.test_convolve_pixel -v
test_convolve_pixel (__main__.Homework1Test) ... FAIL

======================================================================
FAIL: test_convolve_pixel (__main__.Homework1Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "hw1_test.py", line 48, in test_convolve_pixel
    self.assertAlmostEqual(result, 18.4, places=4,
AssertionError: 18.6 != 18.4 within 4 places (0.20000038146972798 difference) : The correct result is computed.

----------------------------------------------------------------------
Ran 1 test in 0.001s
```

  2. I went through the slack chat and someone had put in a comment like, it correlates to 18.6 but it convolutes to 18.4. That is when I realized where I went wrong.



This is a picture from a website I landed on when I looked for convolution vs correlation. I tried transposing first on paper, but that won't give me the desired matrix. I tried transpose + mirror image, that didn't work too.



Convolution vs. image filtering

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

Filtering (also known as cross correlation)

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

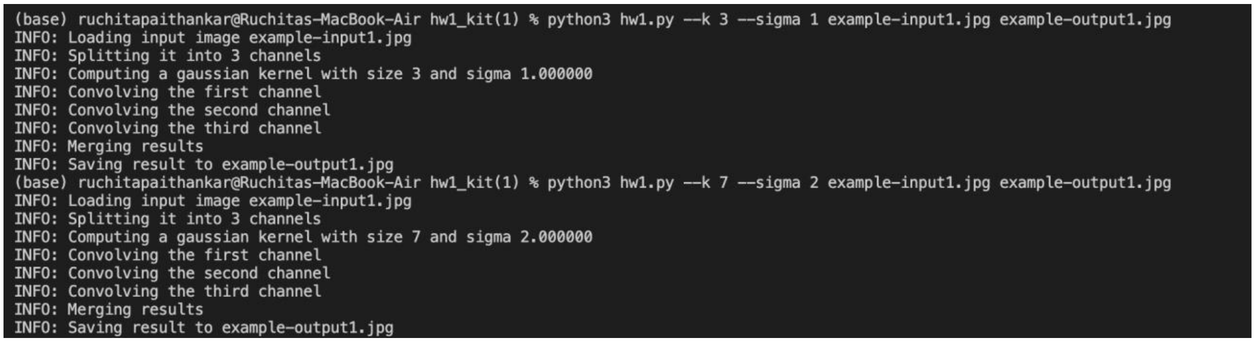
Convolution

Remembered the professor mentioning that x changes to y and y to x. So, I thought of rotating the matrix 90 degree and then rotating it once more to complete a 180 degree rotation, gave me the kernel I needed.

3. Taking this reverse kernel row by row and column by column on the image, multiplying elements by each other and then summing up the result.
4. Again, I used -size/2 to size/2 of the kernel to iterate through the pixels.

- **convolve(img, kernel):**

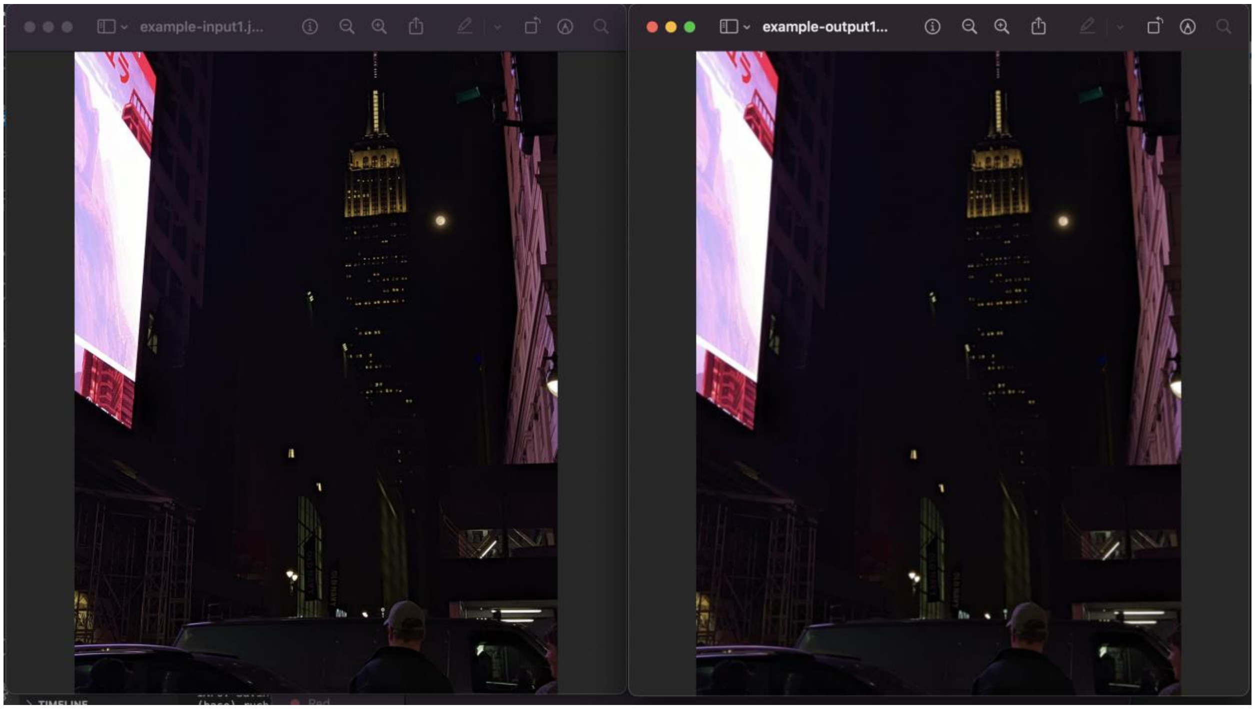
1. convoluting the image pixel by pixel (x,y) with the kernel from the previous function.

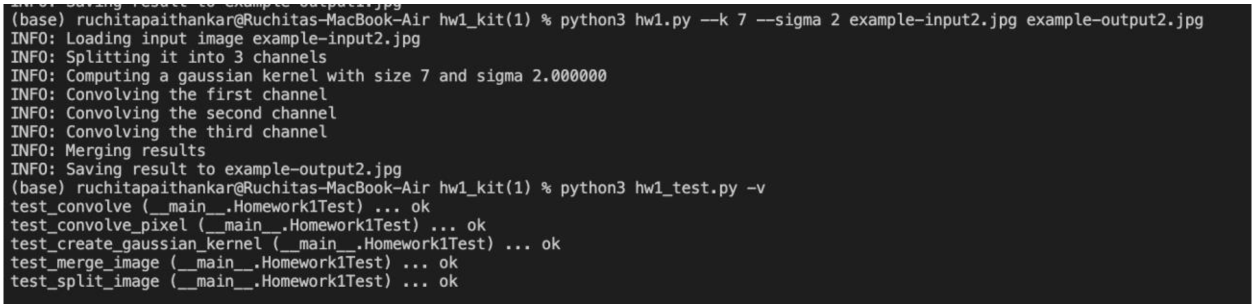


```
(base) ruchitapaithankar@Ruchitas-MacBook-Air hw1_kit(1) % python3 hw1.py --k 3 --sigma 1 example-input1.jpg example-output1.jpg
INFO: Loading input image example-input1.jpg
INFO: Splitting it into 3 channels
INFO: Computing a gaussian kernel with size 3 and sigma 1.000000
INFO: Convolving the first channel
INFO: Convolving the second channel
INFO: Convolving the third channel
INFO: Merging results
INFO: Saving result to example-output1.jpg
(base) ruchitapaithankar@Ruchitas-MacBook-Air hw1_kit(1) % python3 hw1.py --k 7 --sigma 2 example-input1.jpg example-output1.jpg
INFO: Loading input image example-input1.jpg
INFO: Splitting it into 3 channels
INFO: Computing a gaussian kernel with size 7 and sigma 2.000000
INFO: Convolving the first channel
INFO: Convolving the second channel
INFO: Convolving the third channel
INFO: Merging results
INFO: Saving result to example-output1.jpg
```

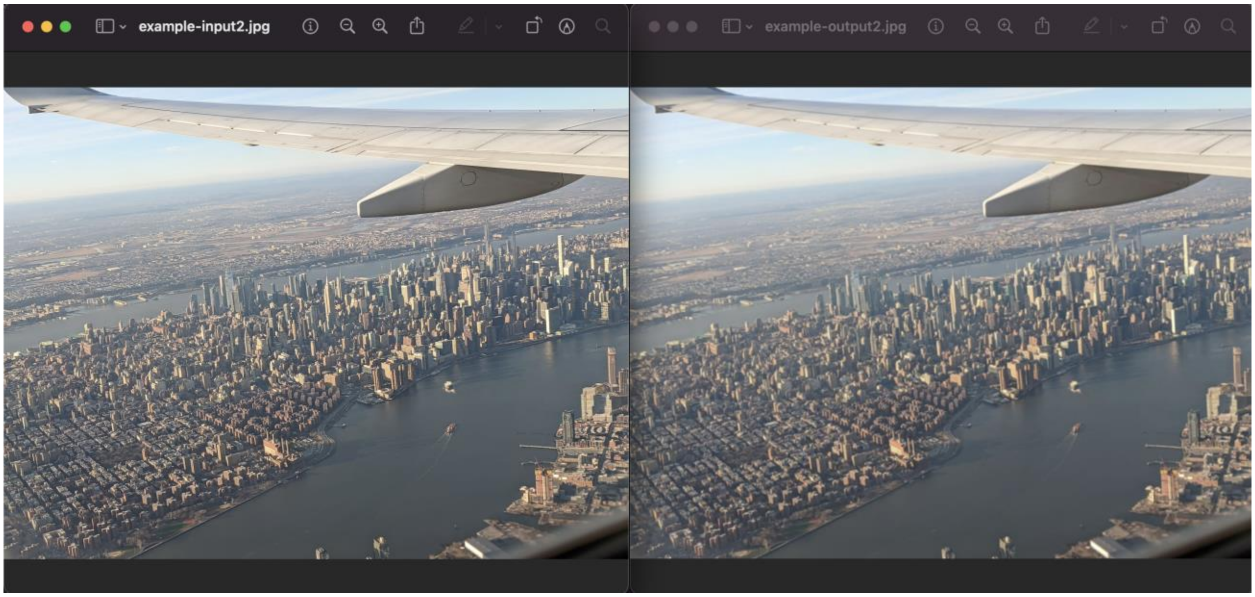
Tried running the file for the example-input1 which k=3 and sigma =1 and didn’t see much change. So tried, k=7 and sigma 2 and noticed slight blur on the edges of some sharp lines.

Below is the side by side comparison of the two example images I tested on. The first example doesn’t show much difference. The second example show a little more noticeable difference.





```
INFO: Saving result to example-output1.jpg
(base) ruchitapaithankar@Ruchitas-MacBook-Air hw1_kit(1) % python3 hw1.py --k 7 --sigma 2 example-input2.jpg example-output2.jpg
INFO: Loading input image example-input2.jpg
INFO: Splitting it into 3 channels
INFO: Computing a gaussian kernel with size 7 and sigma 2.000000
INFO: Convolving the first channel
INFO: Convolving the second channel
INFO: Convolving the third channel
INFO: Merging results
INFO: Saving result to example-output2.jpg
(base) ruchitapaithankar@Ruchitas-MacBook-Air hw1_kit(1) % python3 hw1_test.py -v
test_convolve (__main__.Homework1Test) ... ok
test_convolve_pixel (__main__.Homework1Test) ... ok
test_create_gaussian_kernel (__main__.Homework1Test) ... ok
test_merge_image (__main__.Homework1Test) ... ok
test_split_image (__main__.Homework1Test) ... ok
```


example-input2.jpg
example-output2.jpg