University of British Columbia

CPSC 425: Computer Vision

Assignment 1

Simon Ghyselincks

Self-Studied based off of UBC CPSC 425 2023T1 course material

2021

In this question, you will be practicing filtering by hand on the following image. You will enter your final result for each question in the provided empty tables.

1	0	0	0	1
2	3	0	8	0
2	0	0	0	3
0	0	1	0	0

Note that you do not have to fill all the cells. If a cell contains a number that is not an integer, enter it as a fraction.

Question (1a)

Apply the correlation filter to the image with **no padding**.

0	0	1
0	0	0
-1	0	0

Enter your final result here in integer or fraction:

	-2	0	1	
	0	8	-2	

Question (1b)

Apply the convolution filter to the image with **no padding**.

0	0	1
0	0	0
-1	0	0

Enter your final result here in integer or fraction:

	2	0	-1	
	0	-8	2	

Question (1c)

Apply the filter to the image with **zero padding** (zero padding means to pad your image with zeros; it is not the same as "no padding").

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

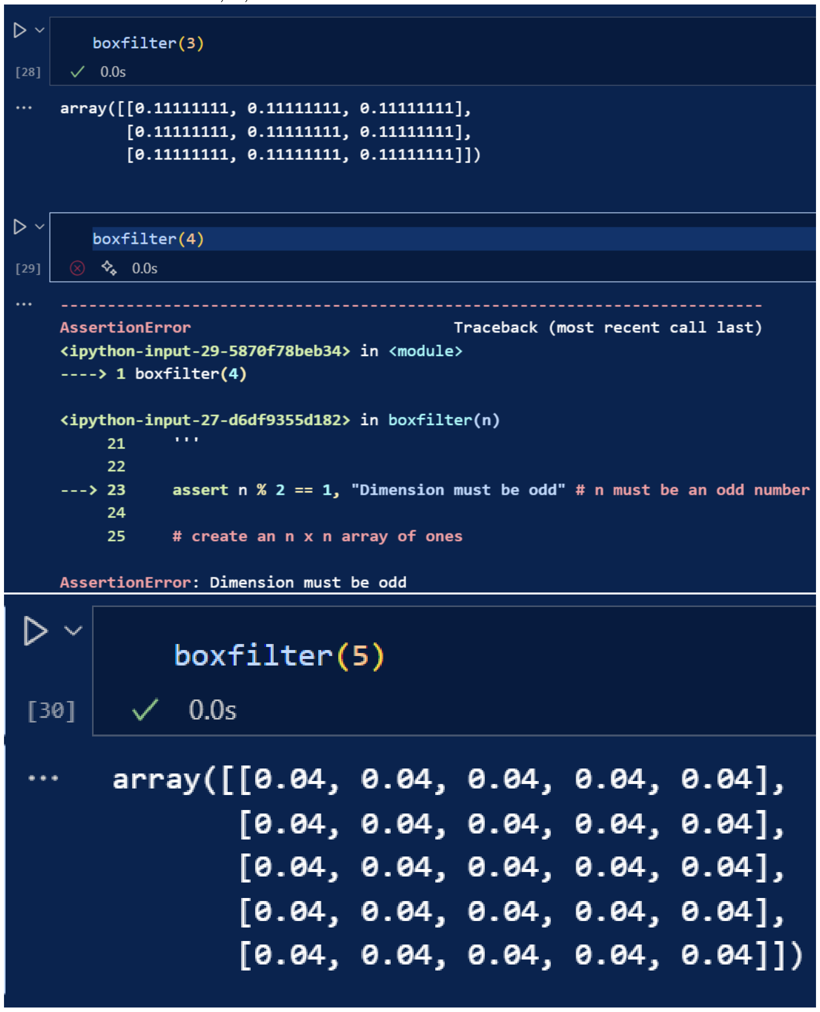
Enter your final result here in integer or fraction:

$\frac{6}{9}$	$\frac{6}{9}$	$\frac{11}{9}$	$\frac{9}{9}$	$\frac{9}{9}$
$\frac{8}{9}$	$\frac{8}{9}$	$\frac{11}{9}$	$\frac{12}{9}$	$\frac{12}{9}$
$\frac{7}{9}$	$\frac{8}{9}$	$\frac{12}{9}$	$\frac{12}{9}$	$\frac{11}{9}$
$\frac{2}{9}$	$\frac{3}{9}$	$\frac{1}{9}$	$\frac{4}{9}$	$\frac{3}{9}$

Part 2

1.

Box filters for 3,4,5

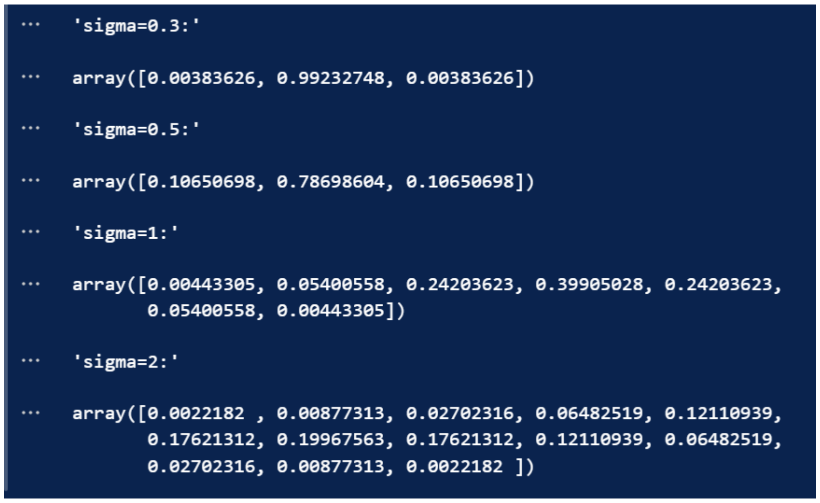


```
boxfilter(3)
[28]  0.0s

array([[0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111]])
```

```
boxfilter(4)
[29]  0.0s

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-29-5870f78beb34> in <module>
----> 1 boxfilter(4)

<ipython-input-27-d6df9355d182> in boxfilter(n)
     21     '''
     22 
---> 23     assert n % 2 == 1, "Dimension must be odd" # n must be an odd number
     24 
     25     # create an n x n array of ones

AssertionError: Dimension must be odd
```

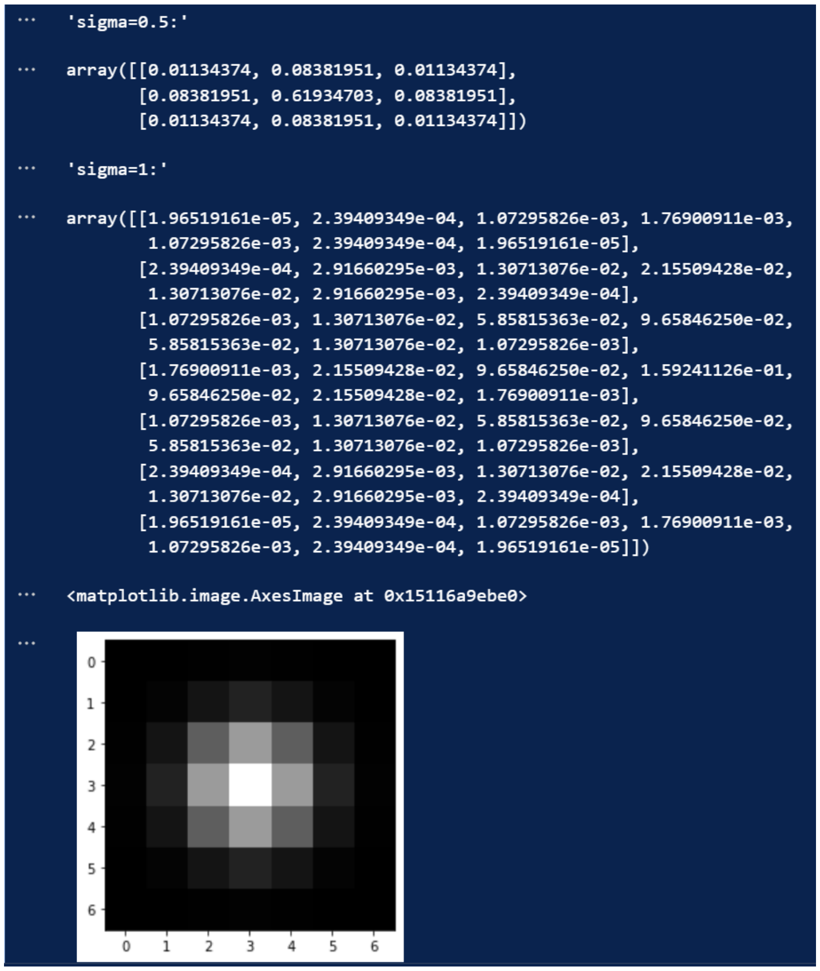
```
boxfilter(5)
[30]  0.0s

array([[0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04]])
```

2.

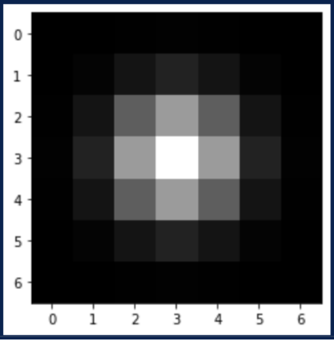


```
'sigma=0.3:'

array([0.00383626, 0.99232748, 0.00383626])

'sigma=0.5:'

array([0.10650698, 0.78698604, 0.10650698])

'sigma=1:'

array([0.00443305, 0.05400558, 0.24203623, 0.39905028, 0.24203623,
       0.05400558, 0.00443305])

'sigma=2:'

array([0.0022182 , 0.00877313, 0.02702316, 0.06482519, 0.12110939,
       0.17621312, 0.19967563, 0.17621312, 0.12110939, 0.06482519,
       0.02702316, 0.00877313, 0.0022182 ])
```

3.

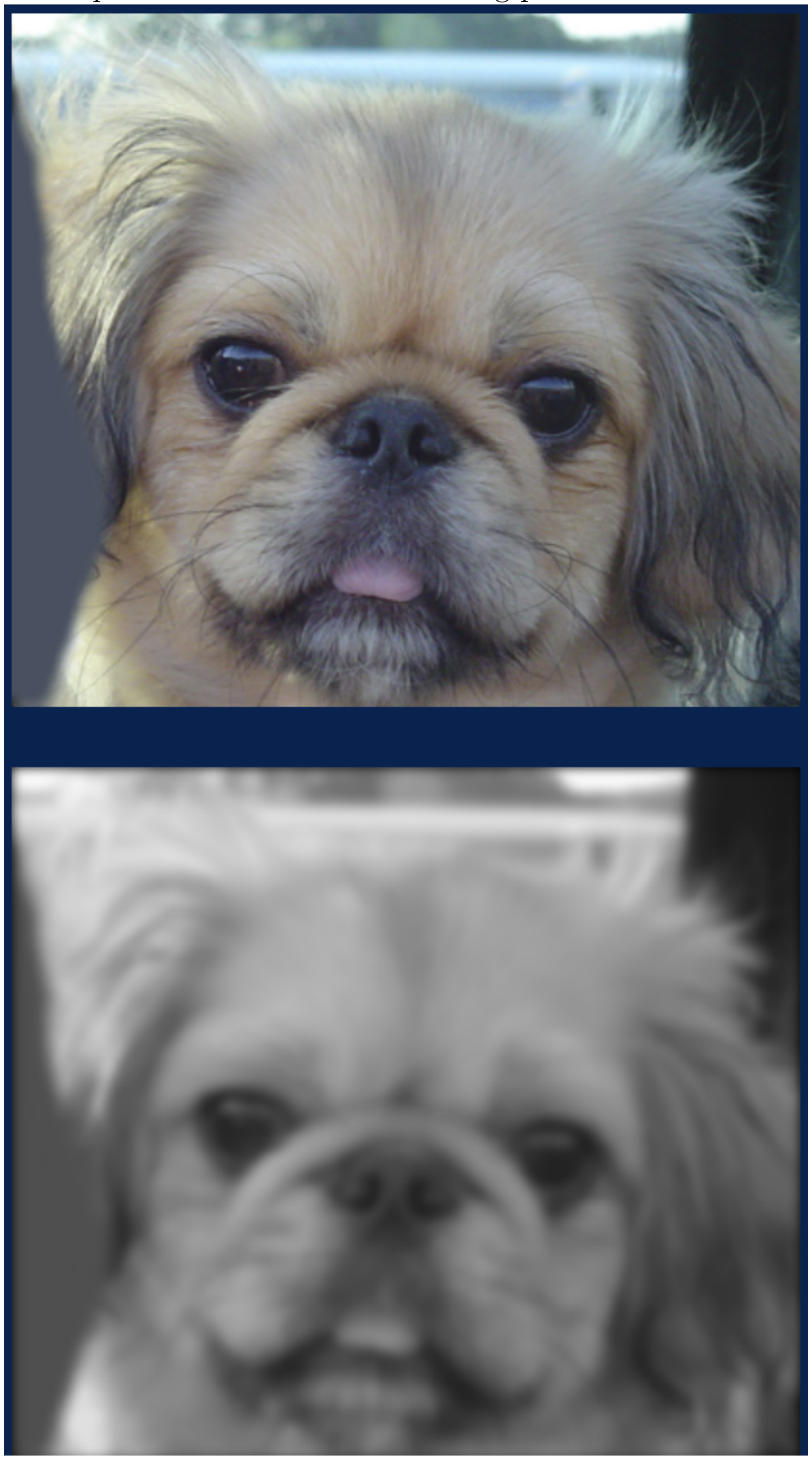


```
'sigma=0.5:'

array([[0.01134374, 0.08381951, 0.01134374],
       [0.08381951, 0.61934703, 0.08381951],
       [0.01134374, 0.08381951, 0.01134374]])

'sigma=1:'

array([[1.96519161e-05, 2.39409349e-04, 1.07295826e-03, 1.76900911e-03,
        1.07295826e-03, 2.39409349e-04, 1.96519161e-05],
       [2.39409349e-04, 2.91660295e-03, 1.30713076e-02, 2.15509428e-02,
        1.30713076e-02, 2.91660295e-03, 2.39409349e-04],
       [1.07295826e-03, 1.30713076e-02, 5.85815363e-02, 9.65846250e-02,
        5.85815363e-02, 1.30713076e-02, 1.07295826e-03],
       [1.76900911e-03, 2.15509428e-02, 9.65846250e-02, 1.59241126e-01,
        9.65846250e-02, 2.15509428e-02, 1.76900911e-03],
       [1.07295826e-03, 1.30713076e-02, 5.85815363e-02, 9.65846250e-02,
        5.85815363e-02, 1.30713076e-02, 1.07295826e-03],
       [2.39409349e-04, 2.91660295e-03, 1.30713076e-02, 2.15509428e-02,
        1.30713076e-02, 2.91660295e-03, 2.39409349e-04],
       [1.96519161e-05, 2.39409349e-04, 1.07295826e-03, 1.76900911e-03,
        1.07295826e-03, 2.39409349e-04, 1.96519161e-05]])

<matplotlib.image.AxesImage at 0x15116a9ebe0>
```



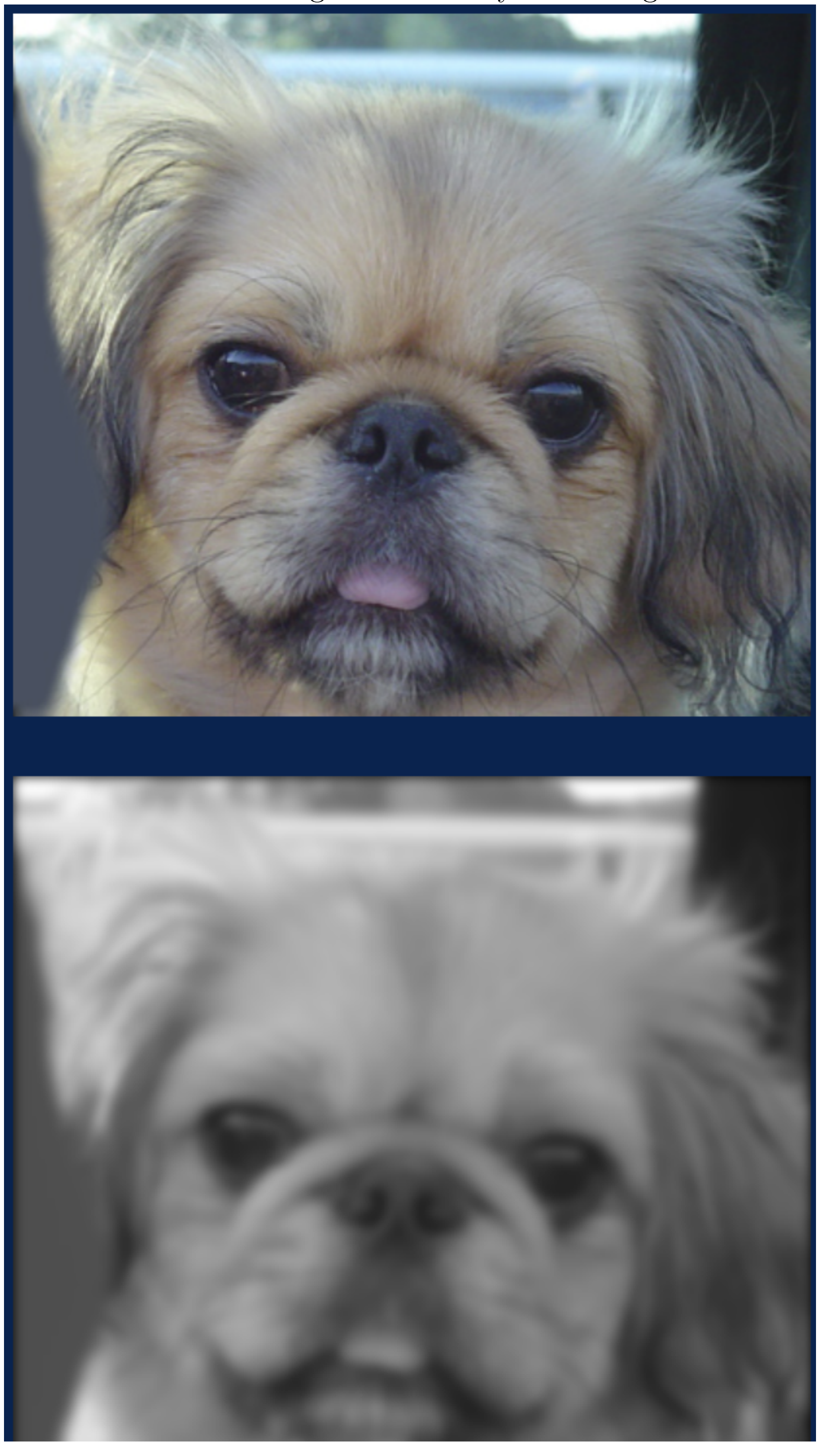
4.

A comparison of before and after dog picture



5.

There are two different functions because they are not equivalent. The correlation and convolution are only the same for a 180 degree rotationally symmetric kernel, which is true for the Gaussian. Rotating the kernel by 180 still gives the same result.



6.

Manual method takes 0.8032005310058594 seconds
Scipy method takes 0.7100874423980713 seconds

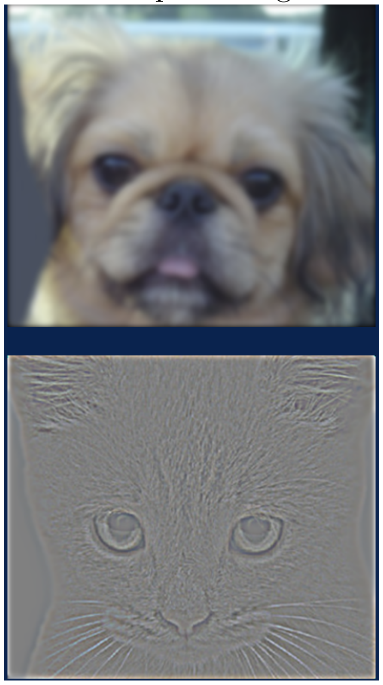
The scipy method is faster, likely because it is more optimized. One thing that could be improved is vectorizing the nested for loops. The code for scipy has the capability to be written in C or something lower level that will overall have better speed.

7.

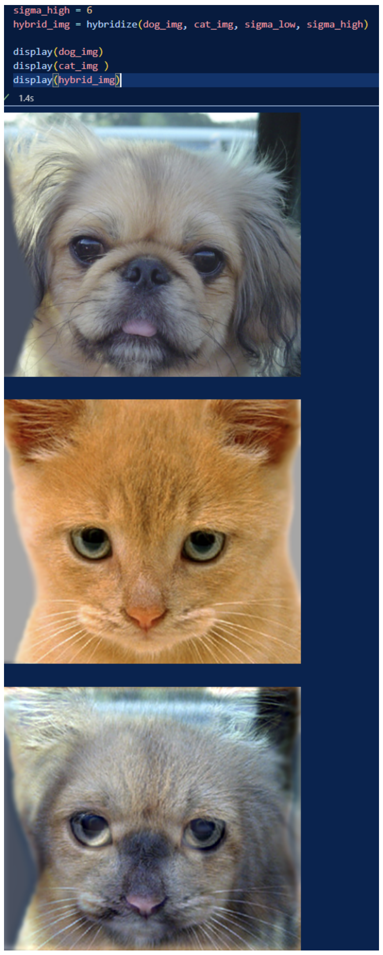
The gaussian kernel is seperable as we have shown using the construction via an outer-product. We are making a lot of repetition in the calculations because we first convolved two m dimension 1D matrices to form a m^2 kernel convolved with an n^2 image array. There are a total of $O(n^2m^2)$ operations to make for the convolution done in this order. Instead we could leverage the seperability and do two m dimension 1D convolutions on the n^2 image array. This would result in $O(2mn^2)$ operations, which is much less than the previous method for when m is much larger than two.

Part 3

The decomposed dog and cat images are shown:

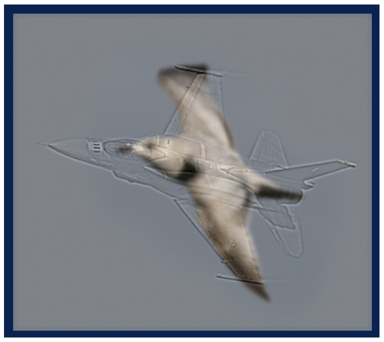


The result using different values of σ are shown, this is for $sigma_{dog} = 3$ and $\sigma_{cat} = 6$:



```
sigma_high = 6
hybrid_img = hybridize(dog_img, cat_img, sigma_low, sigma_high)

display(dog_img)
display(cat_img )
display(hybrid_img)
```

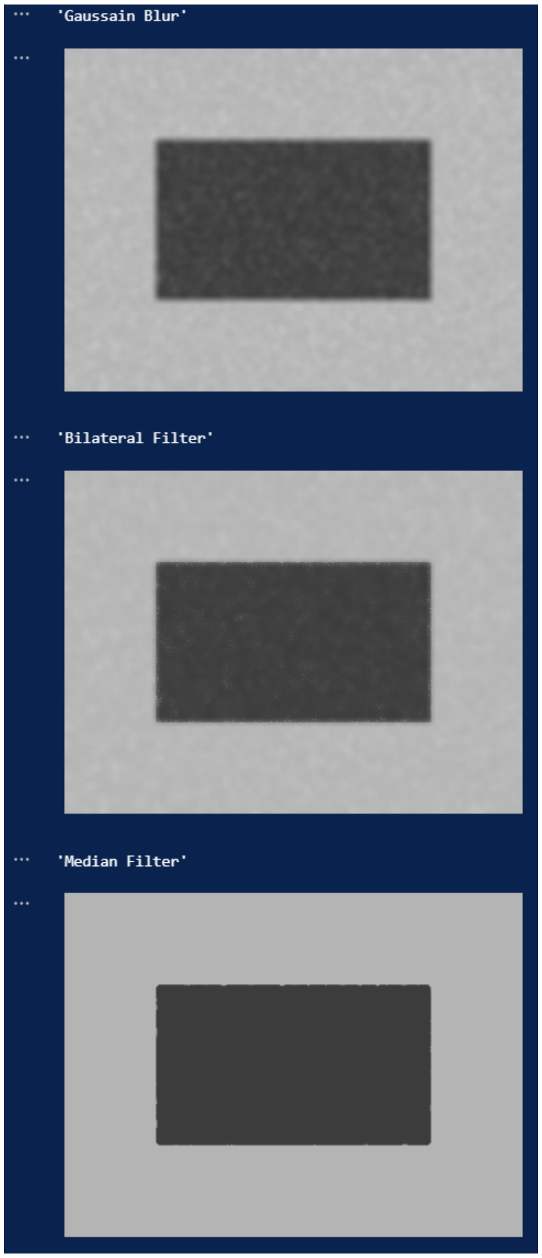
```
✓ 1.4s
```



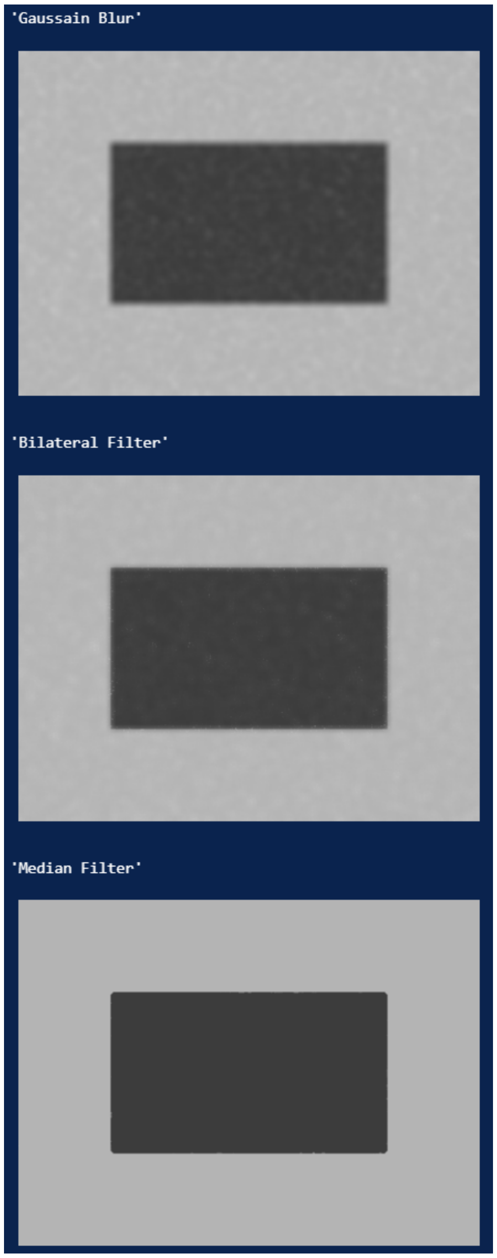
Other combinations have been tried but in the interest of time are not included in this report

Part 4

The different blur techniques are shown. The speckeled is on the left, with Gaussian noise on the right:



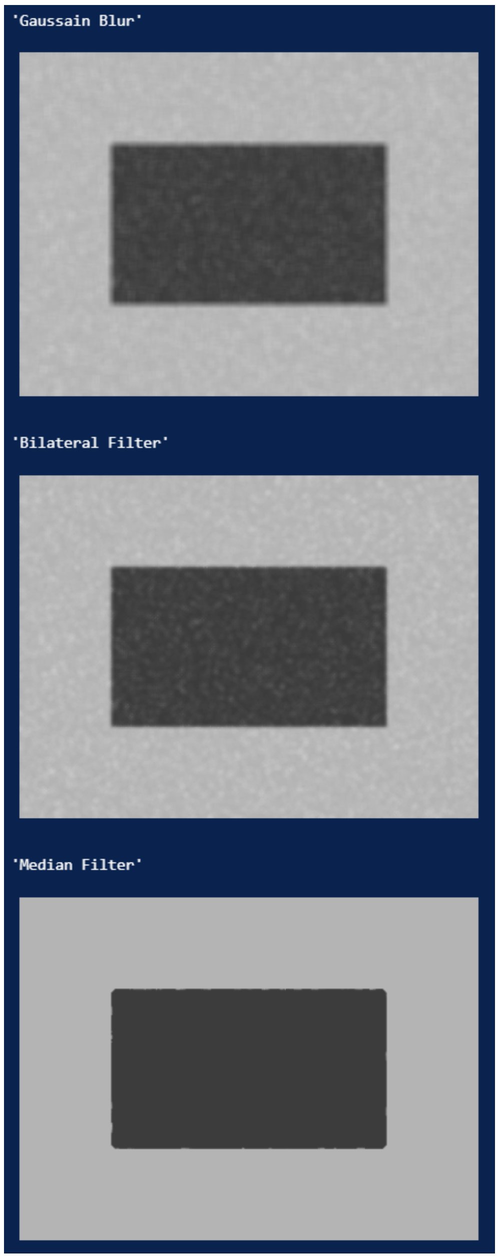
```
'Gaussain Blur'
'Bilateral Filter'
'Median Filter'
```

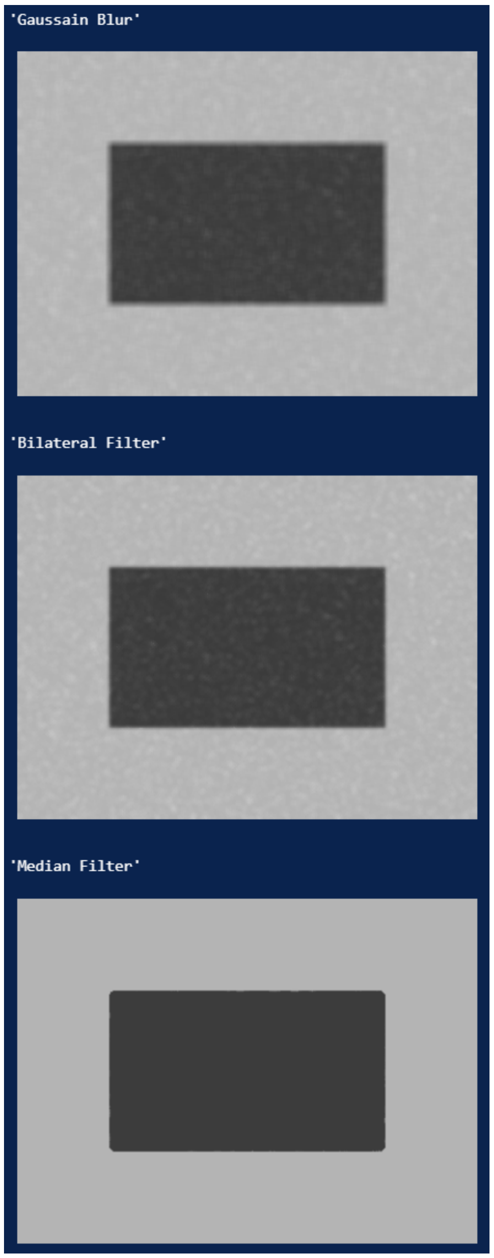


```
'Gaussain Blur'
'Bilateral Filter'
'Median Filter'
```

2.

The speckeled is on the left, with Gaussian noise on the right. Applying the specific filters yields:


'Gaussain Blur'
'Bilateral Filter'
'Median Filter'


'Gaussain Blur'
'Bilateral Filter'
'Median Filter'