Report

March 6, 2021

Assignment 1

Mohamed A. AbdelHamed - 900163202

oscar@aucegypt.edu

1 Part 1:

1.1 Transformation Formats:

[1]:
```python
from parse_utils import trans_format
print(trans_format)
```

```
Please insert a list of your transformations in the following format:
<trans_key1 …args1> <trans_key2 …args2> … <trans_key_n …args_n>

Available transormations:
<TRANS offset>
<SCALE Sx Sy>
<ROT angle(degrees) Px Py>
<NTHP n>
<HE>
```

1.2 (a) Matrix transformations:

1.2.1 i. Translations:

[2]:
```python
import part1
trans = '<TRANS 50 100>'
part1.run(interactive=False, img_path='samples/rome.jpg', trans_str=trans,␣
 ↪effect='Translated')
```

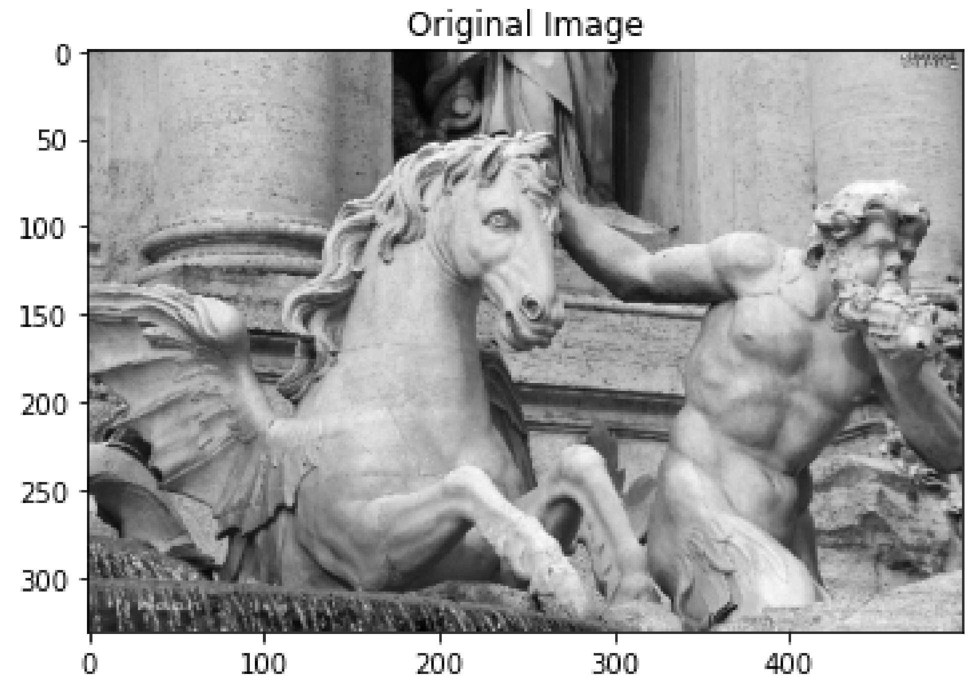
1.2.2 ii. Rotation:

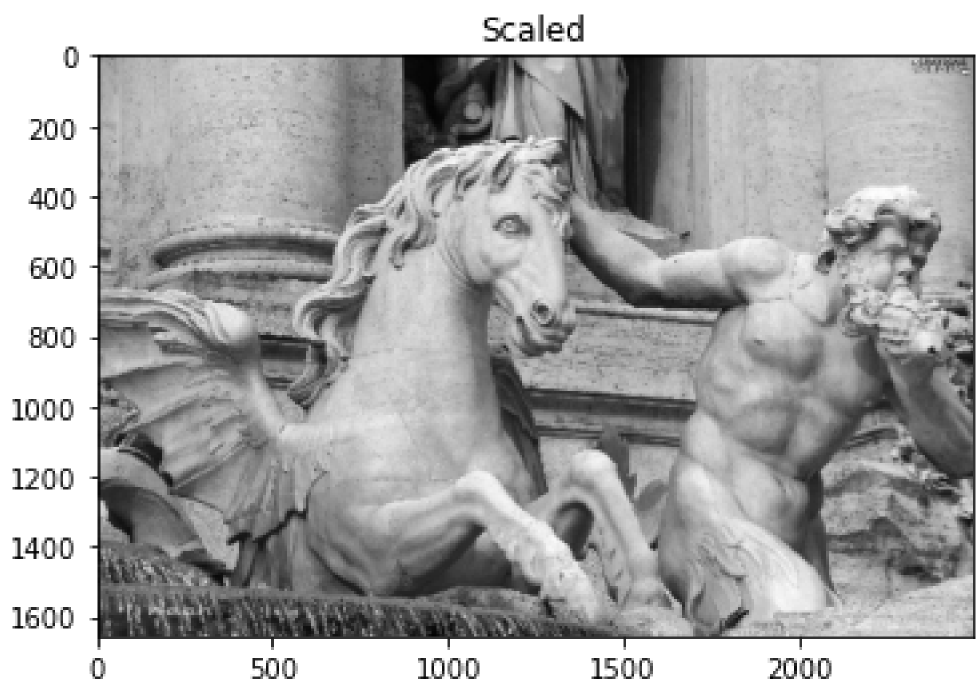
[3]:
```python
import part1
trans = '<ROT 45 250 250>'
part1.run(interactive=False, img_path='samples/rome.jpg', trans_str=trans, effect='Rotated')
```

1.2.3 iii. Scaling:

Note the axes values are used to observe new scaling in pixels.

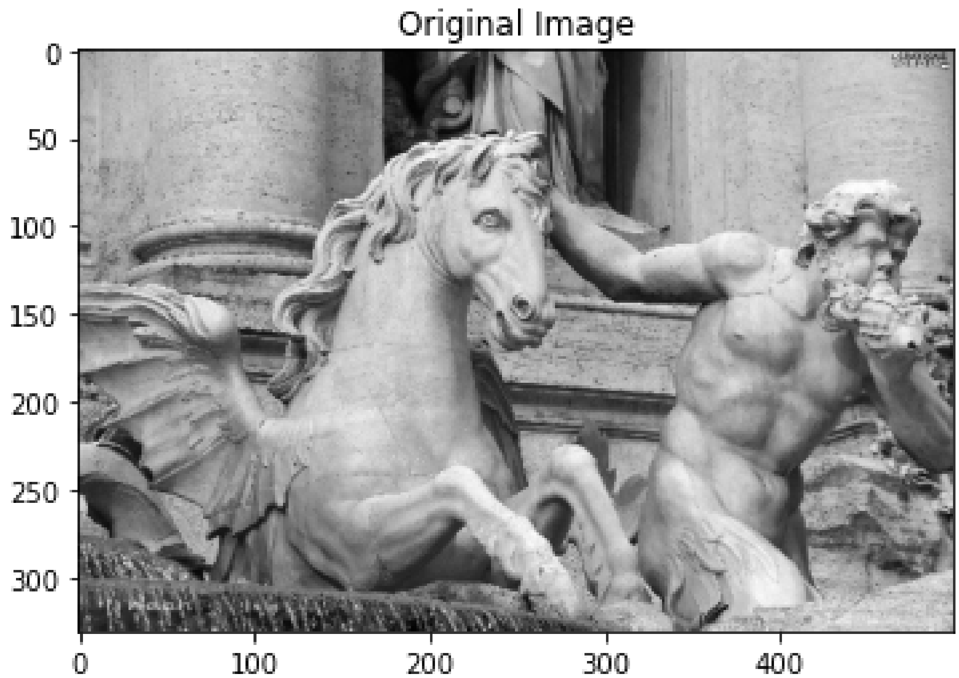
[4]:
```python
import part1
trans = '<SCALE 5 5>'
part1.run(interactive=False, img_path='samples/rome.jpg', trans_str=trans, effect='Scaled', side_by_side=False)
```

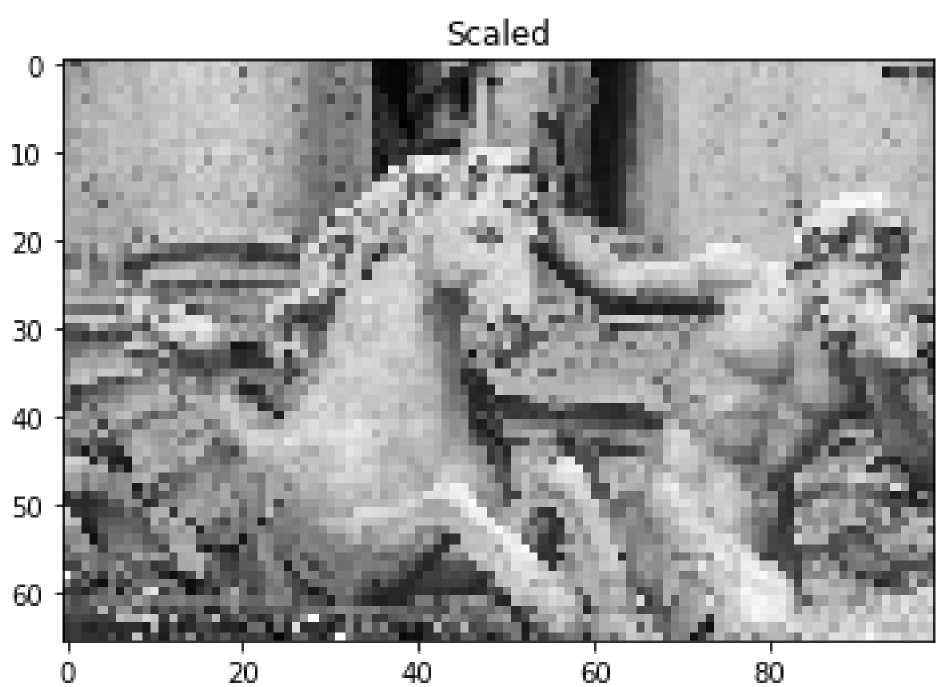

Original Image
0
50
100
150
200
250
300
0
100
200
300
400


Scaled
0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
2000

[5]:

```
import part1
trans = '<SCALE 0.2 0.2>'
part1.run(interactive=False, img_path='samples/rome.jpg', trans_str=trans,␣
 ↪effect='Scaled', side_by_side=False)
```



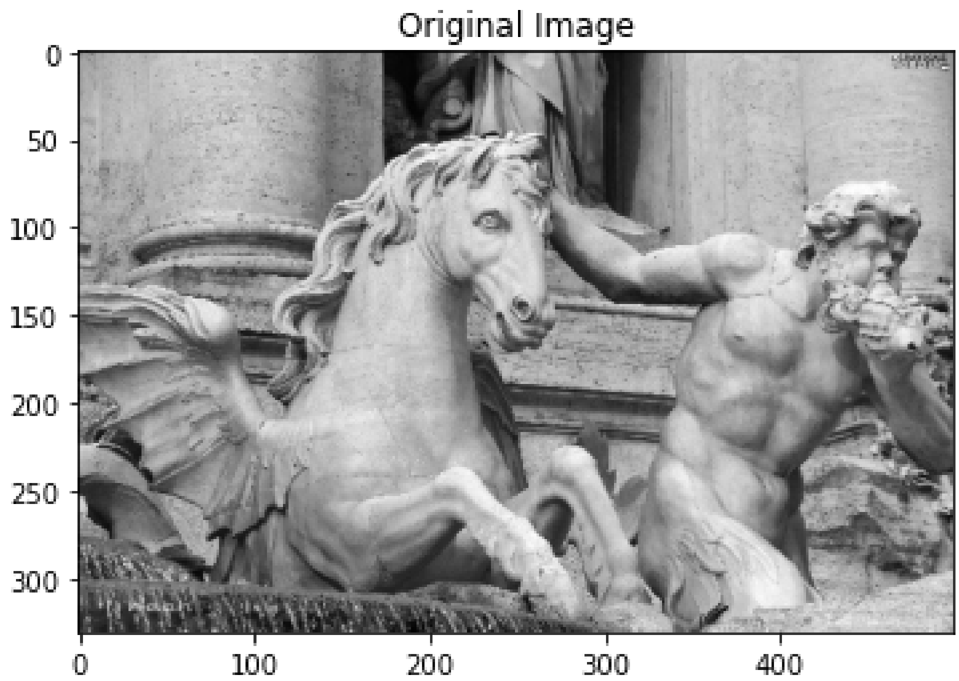


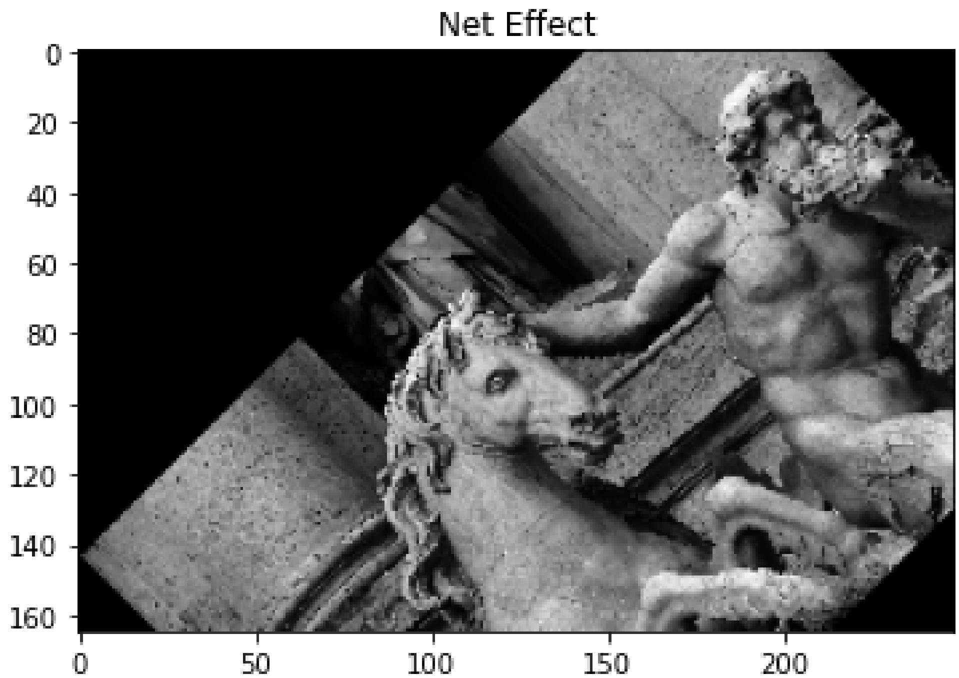
Comments:
1. Scaling up does not show notable gaps because the transformation is executed in reverse (i.e. looping over destination pixel instead of source pixels).
2. Scaling down clearly reduces image quality due to loss of pixels.

1.2.4 iv. Combined Transformations:

Note that the program also supports mixing transformations with the ones in part b (HE, and n^{th} power).

[6]:
```
import part1
trans = '''
<TRANS 50 100>
<ROT 45 250 250>
<SCALE 0.5 0.5>
<NTHP 2>
'''
part1.run(interactive=False, img_path='samples/rome.jpg', trans_str=trans,␣
↪effect='Net Effect', side_by_side=False)
```

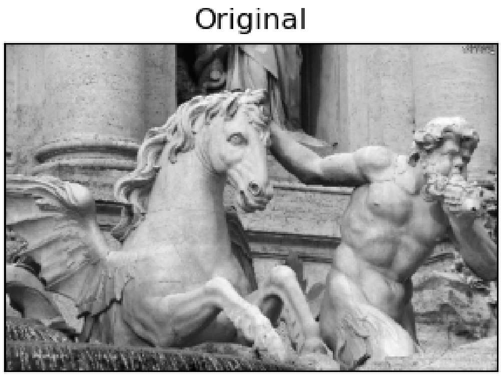




1.3 (b):

1.3.1 i. n^{th} power:

[7]:
```
import part1
trans1 = '<NTHP 5>'
part1.run(interactive=False, img_path='samples/rome.jpg', trans_str=trans1)
```



[8]:
```
import part1
trans2 = '<NTHP 0.2>'
```

```
part1.run(interactive=False, img_path='samples/rome.jpg', trans_str=trans2)
```

1.3.2 ii. Histogram Equalization:

[9]:
```
import part1
trans = '<HE>'
part1.run(interactive=False, img_path='samples/he.jpg', trans_str=trans)
```

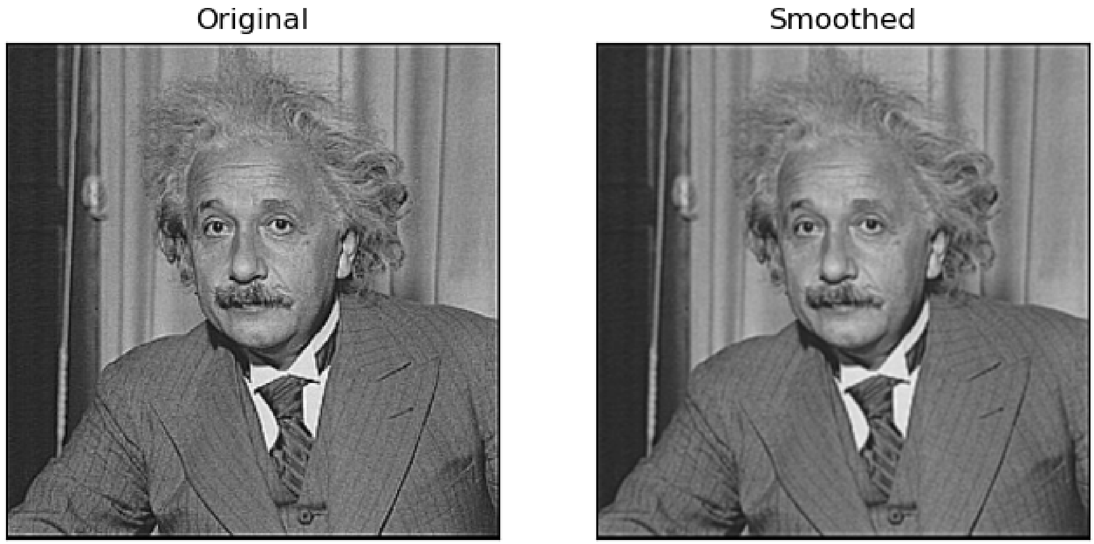
2 Part 2:

2.1 (a) Smoothing Filter (using averaging):

$$F_{avg} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sample run on a sharpened image:

```python
[10]: from part2 import part_a
      part_a()
```

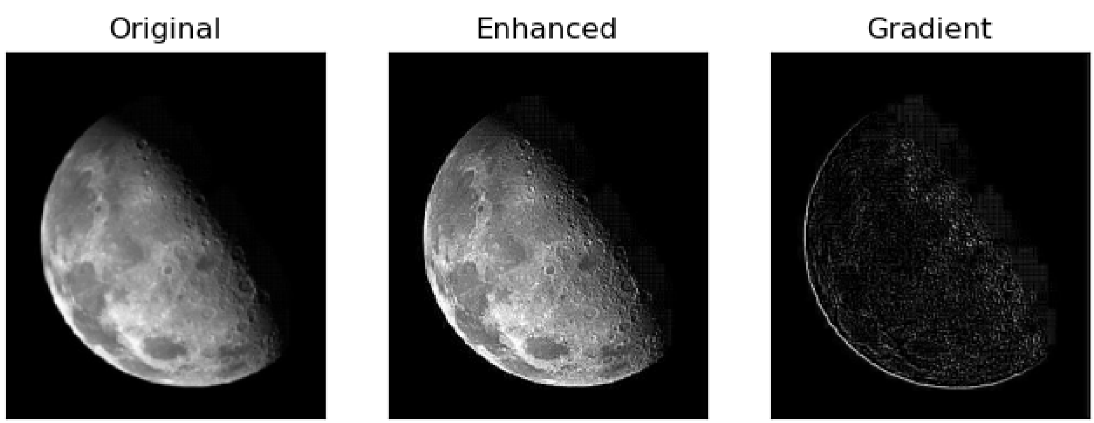


2.2 (b) Gradient Filter (Laplacian):

$$F_{lap} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sample run:

```python
[11]: from part2 import part_b
      part_b()
```

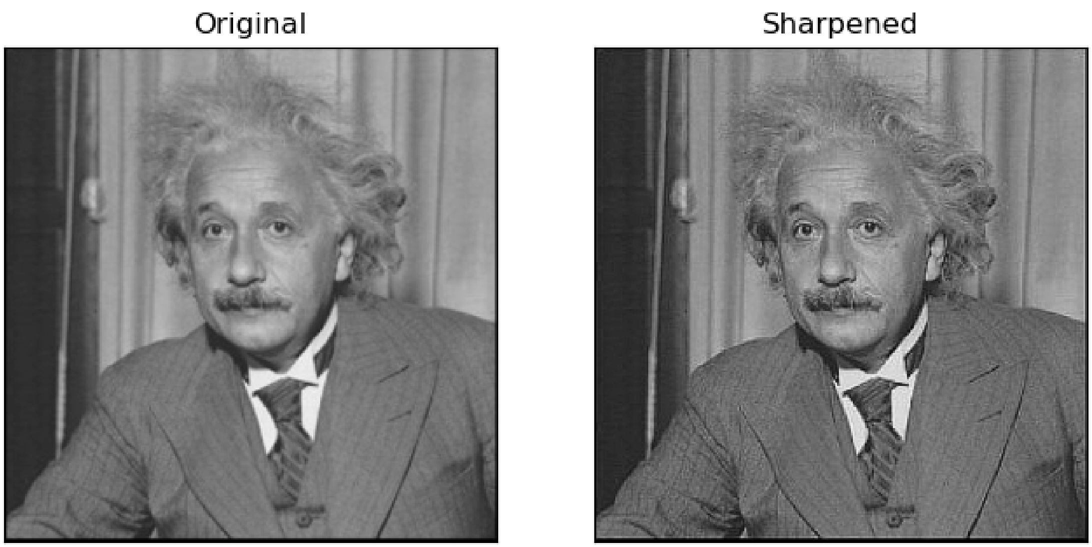


2.3 (c) Sharpening Filter:

$$F_{sharp} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sample run on smoothed image:

[12]:
```
from part2 import part_c
part_c()
```



3 Part 3:

3.1 (a) One other separable filter would be the vertical (or horizontal) Sobel filter for edge detection:

For our example, we use the vertical edge filter:

$$\text{2D Sobel} = F = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

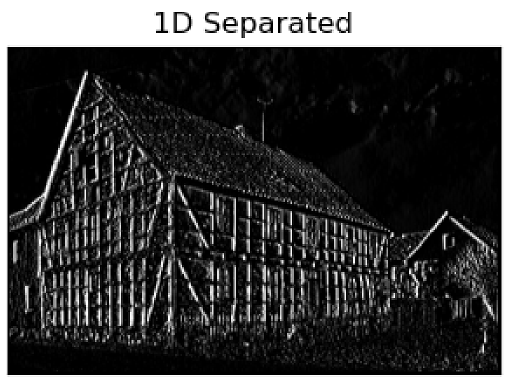
1D Decomposition:

$$F_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, F_2 = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \text{ such that } F = F_1.F_2$$

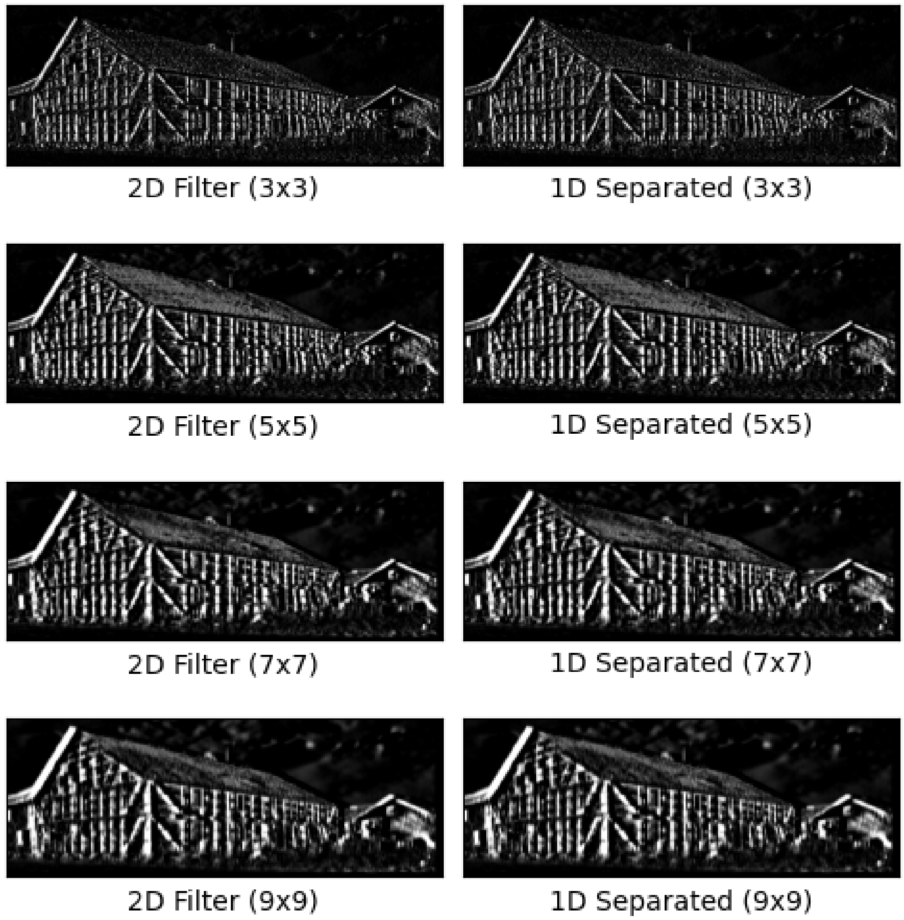
3.2 (b)

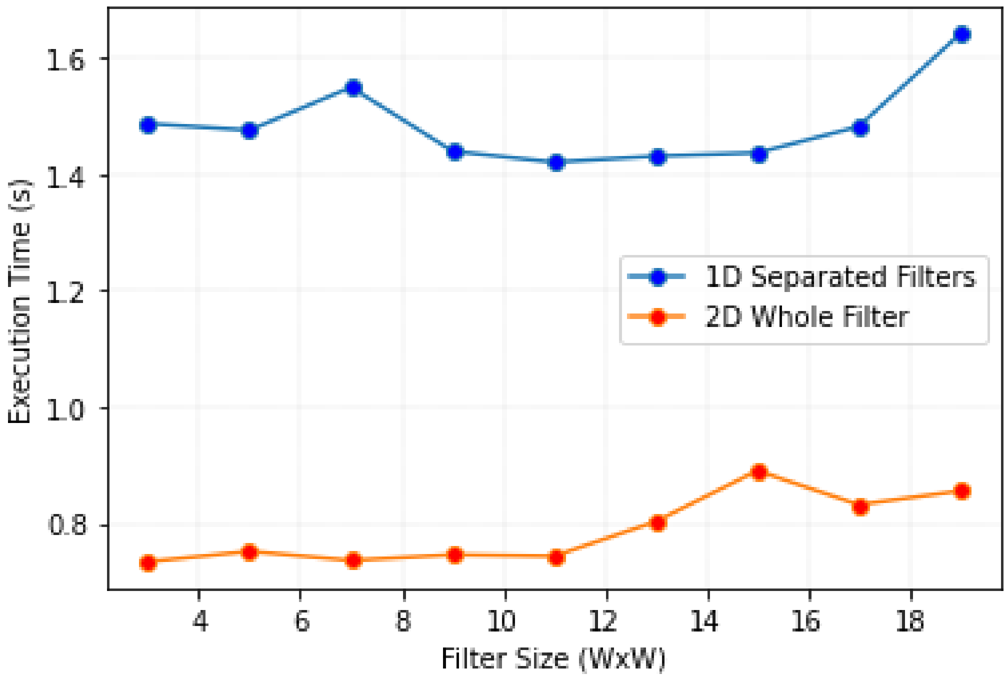
To extend the Sobel filter to larger sizes, the method described in this paper was used.
Below are the results comparison for different filter sizes (2D vs. Separated 1D):

[1]:
```
import part3
part3.run()
```



2D Filters vs 1D Decomposition





Comments:

1. Filtered images are emphasizing the vertical lines as expected.
2. For the same filter size, the 1D and 2D filter versions look identical as expected.
3. For relatively small filter sizes (like the one used in this experiment), program overheads (e.g. loop packing, memory allocation, copying) dominate the performance gain of filter separation until filter size is large enough for the gain to appear (when the gap starts to close).
4. The consistency of the trend shown is fragile due to high sensitivity in execution time with relatively small filters.