National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

EE 433: Digital Image Processing

Name: Amal Saqib

CMS: 282496

Class: BSCS 9C

Lab 8: Spatial Filtering Basics-1

Date: 8th November 2021

Time: 2.00Pm to 5.00Pm

Instructor: Dr. Imran Malik

Lab 8

Spatial Filtering Basics-1

Task 1a

Increasing the size of the filter, increases the amount of blurriness as the effect of the current pixel on the output pixel decreases. The output pixel is dependent on more neighbouring pixels as the size of the filter increases.

In weighted filters, such as the one given in the lab, the weightage of the current pixel is more as compared to the neighbouring pixels, hence, the output pixel will be impacted more by the current pixel.

CODE

```
from PIL import Image
import numpy as np


# returns the filter of size nxn
def generateFilter(filterSize):
    filter_array = [[1 for j in range(filterSize)] for i in range(filterSize)]
    return filter_array


# calculates the pixel value by averaging
def averaging(inputArray, filterArray):
    pixel_value = np.sum(np.multiply(inputArray, filterArray) / np.sum(filterArray))
    return pixel_value


# applies the filter on the image
def applyingFilter(inputArray, filterArray, filterSize):
    # zero padding
    padding = int((filterSize - 1) / 2)
```

```
    inputArray = np.pad(inputArray, padding)

    height, width = inputArray.shape

    outputArray = inputArray.copy()

    # iterate the original image
    for x in range(padding, height - padding):
        for y in range(padding, width - padding):
            # gets the part of the image the size of the filter matrix
            neighbourhood_array = inputArray[x - padding:x + padding + 1, y - padding: y + padding + 1]
            outputArray[x][y] = averaging(neighbourhood_array, filterArray)

    # removes padding from output
    outputArray = outputArray[padding:height - padding, padding: width - padding]

    return outputArray



# open image
input_image = Image.open("../Lab08/smoothing.tif").convert('L')
input_array = np.asarray(input_image)

# apply averaging
filter_size = 3
filter_array = generateFilter(filter_size)
output_array = applyingFilter(input_array, filter_array, filter_size)

# make and save output image
```

```
output_image = Image.fromarray(output_array)

output_image.save("Output "+ str(filter_size) + "x" + str(filter_size) + ".tif")
```

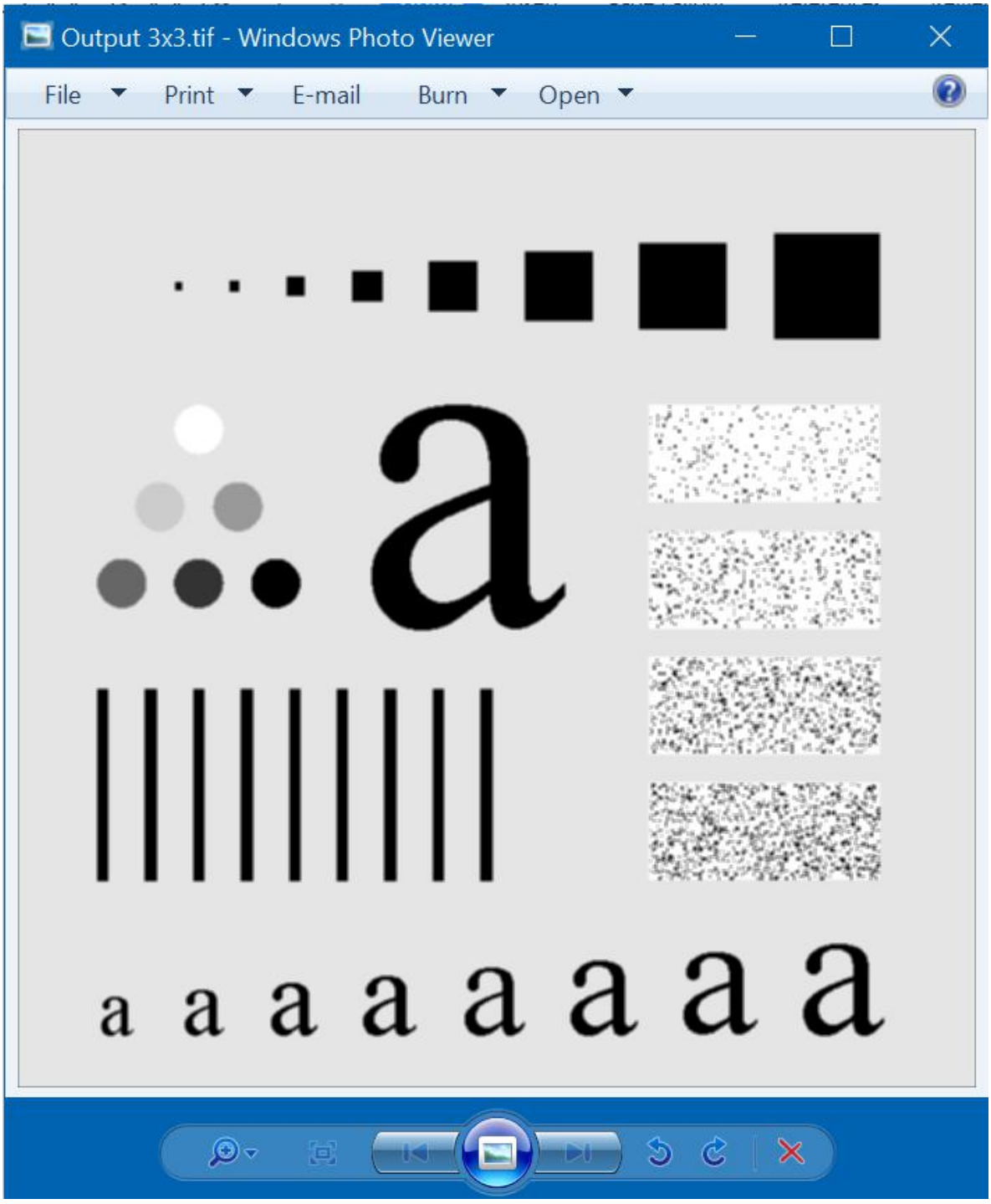
```python
from PIL import Image
import numpy as np

# returns the filter of size nxn
def generateFilter(filterSize):
    filter_array = [[1 for j in range(filterSize)] for i in range(filterSize)]
    return filter_array

# calculates the pixel value by averaging
def averaging(inputArray, filterArray):
    pixel_value = np.sum(np.multiply(inputArray, filterArray) / np.sum(filterArray))
    return pixel_value

```

```python
# applies the filter on the image
def applyingFilter(inputArray, filterArray, filterSize):
    # zero padding
    padding = int((filterSize - 1) / 2)
    inputArray = np.pad(inputArray, padding)

    height, width = inputArray.shape

    outputArray = inputArray.copy()

    # iterate the original image
    for x in range(padding, height - padding):
        for y in range(padding, width - padding):
            # gets the part of the image the size of the filter matrix
            neighbourhood_array = inputArray[x - padding:x + padding + 1, y - padding: y + padding + 1]
            outputArray[x][y] = averaging(neighbourhood_array, filterArray)

    # removes padding from output
    outputArray = outputArray[padding:height - padding, padding: width - padding]

    return outputArray
```

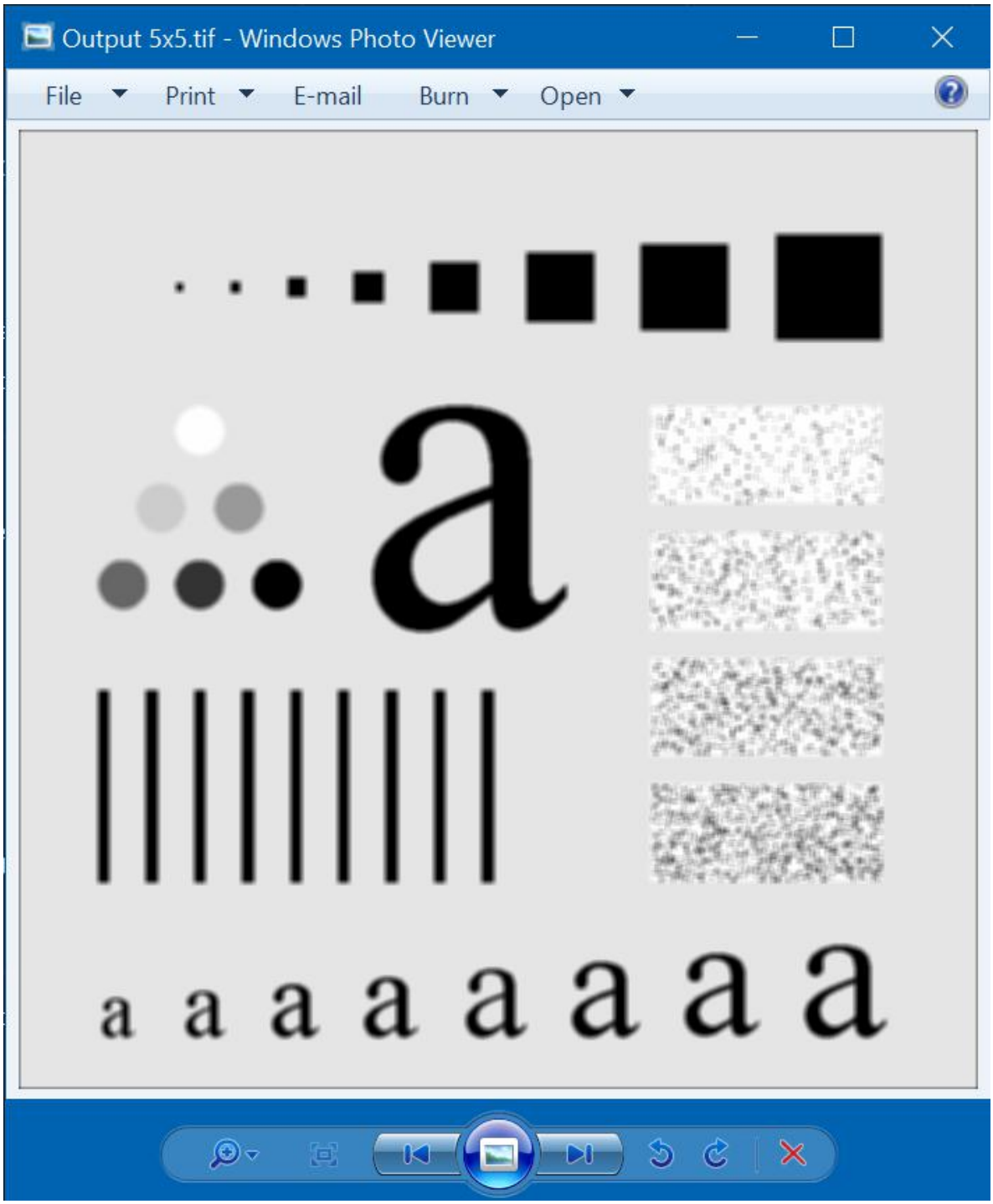
```python

# open image
input_image = Image.open("../Lab08/smoothing.tif").convert('L')
input_array = np.asarray(input_image)

# apply averaging
filter_size = 3
filter_array = generateFilter(filter_size)
output_array = applyingFilter(input_array, filter_array, filter_size)

# make and save output image
output_image = Image.fromarray(output_array)
output_image.save("Output "+ str(filter_size) + "x" + str(filter_size) + ".tif")
```

OUTPUT

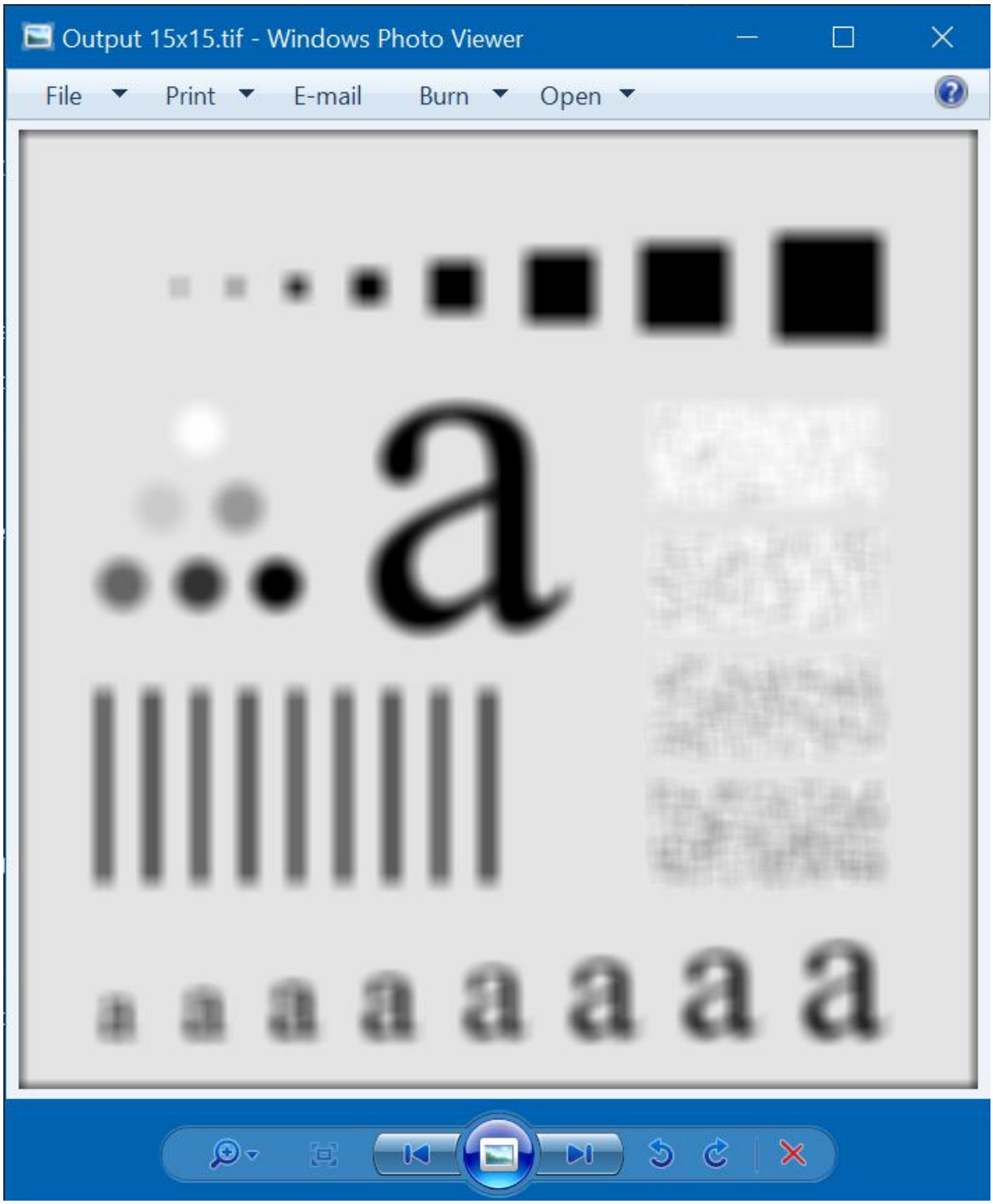
3x3


Output 3x3.tif - Windows Photo Viewer
File
Print
E-mail
Burn
Open

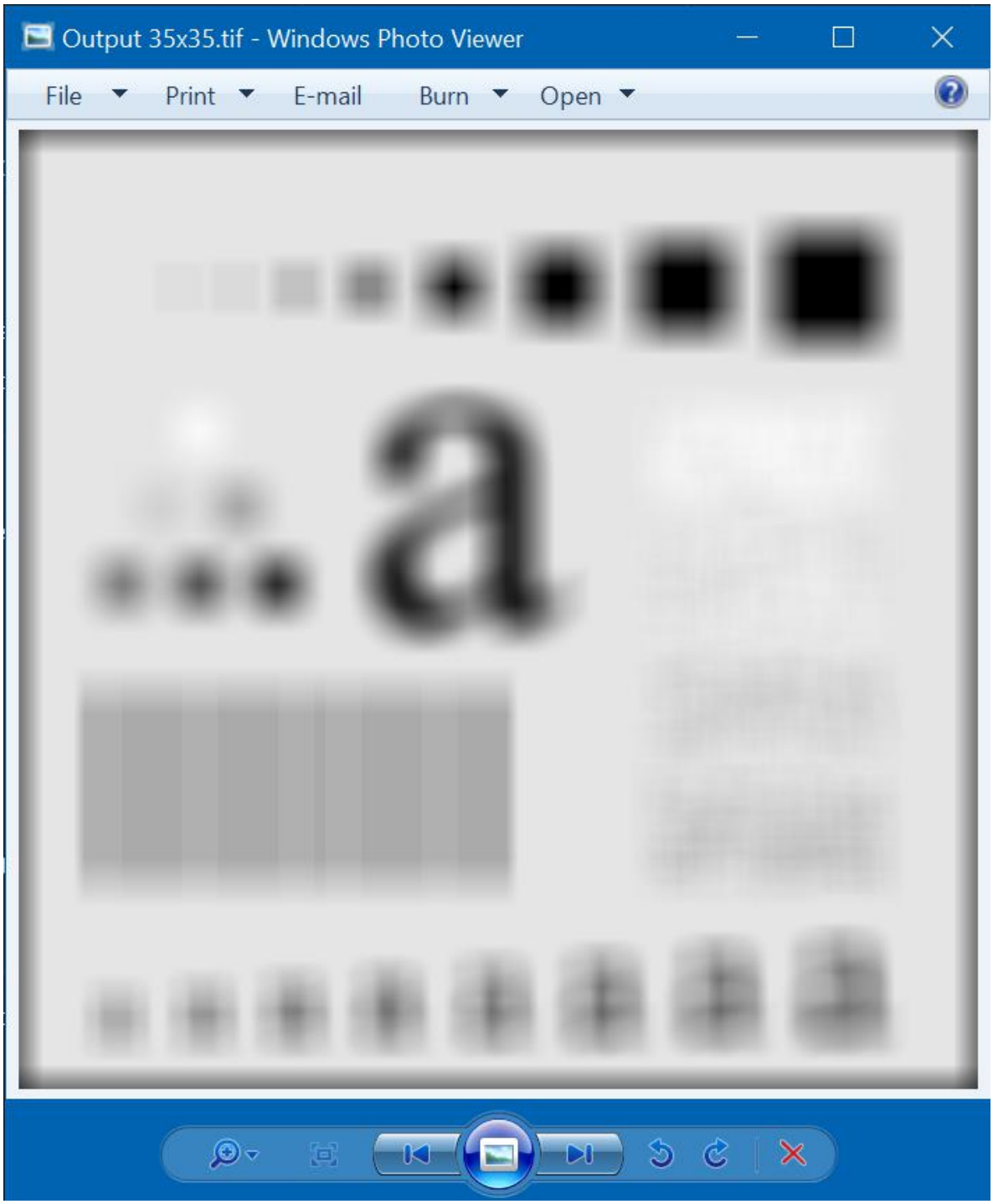
5x5


Output 5x5.tif - Windows Photo Viewer
File
Print
E-mail
Burn
Open

15x15


Output 15x15.tif - Windows Photo Viewer
File
Print
E-mail
Burn
Open

35x35


Output 35x35.tif - Windows Photo Viewer
File
Print
E-mail
Burn
Open

Task 1b

CODE

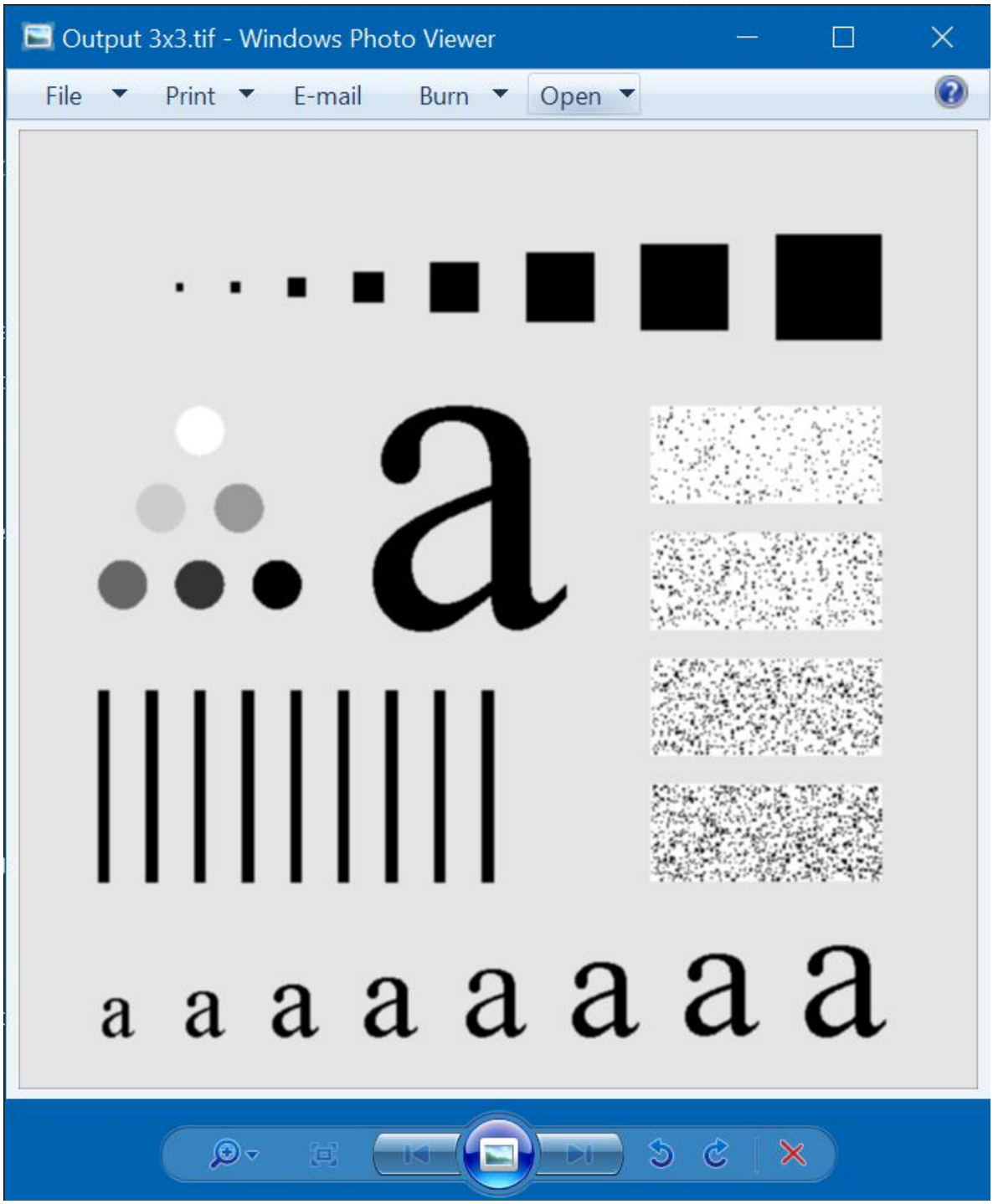
```
from PIL import Image
import numpy as np


# returns the filter of size nxn
def generateFilter(filterSize):
    filter_array = [[1,2,1], [2,4,2], [1,2,1]]
    return filter_array


# calculates the pixel value by averaging
def averaging(inputArray, filterArray):
    pixel_value = np.sum(np.multiply(inputArray, filterArray) / np.sum(filterArray))
    return pixel_value


# applies the filter on the image
def applyingFilter(inputArray, filterArray, filterSize):
    # zero padding
    padding = int((filterSize - 1) / 2)
    inputArray = np.pad(inputArray, padding)


    height, width = inputArray.shape


    outputArray = inputArray.copy()


    # iterate the original image
    for x in range(padding, height - padding):
        for y in range(padding, width - padding):
            # gets the part of the image the size of the filter matrix
```

```
        neighbourhood_array = inputArray[x - padding:x + padding + 1, y - padding: y + padding + 1]
        outputArray[x][y] = averaging(neighbourhood_array, filterArray)

    # removes padding from output
    outputArray = outputArray[padding:height - padding, padding: width - padding]

    return outputArray



# open image
input_image = Image.open("../Lab08/smoothing.tif").convert('L')
input_array = np.asarray(input_image)

# apply averaging
filter_size = 3
filter_array = generateFilter(filter_size)
output_array = applyingFilter(input_array, filter_array, filter_size)

# make and save output image
output_image = Image.fromarray(output_array)
output_image.save("Output "+ str(filter_size) + "x" + str(filter_size) + ".tif")
```

All the code is the same as task 1a, except the filter is hard coded (line 6)

```
1   from PIL import Image
2   import numpy as np
3
4   # returns the filter of size nxn
5   def generateFilter(filterSize):
6       filter_array = [[1,2,1], [2,4,2], [1,2,1]]
7       return filter_array
8
9   # calculates the pixel value by averaging
10  def averaging(inputArray, filterArray):
11      pixel_value = np.sum(np.multiply(inputArray, filterArray) / np.sum(filterArray))
12      return pixel_value
13
```

OUTPUT


Output 3x3.tif - Windows Photo Viewer
File
Print
E-mail
Burn
Open

Task 2

Gaussian Filter

Increasing sigma will result in the filter being such that the middle and the nearest neighbouring pixels' impact will decrease, making the image blurrier.

CODE

```
from PIL import Image
import numpy as np


# returns the filter of size nxn
def generateFilter(filterSize):
    filter_array = [[1,1,2,2,2,1,1],
                    [1,2,2,4,2,2,1],
                    [2,2,4,8,4,2,2],
                    [2,4,8,16,8,4,2],
                    [2,2,4,8,4,2,2],
                    [1,2,2,4,2,2,1],
                    [1,1,2,2,2,1,1]]

    return filter_array


# calculates the pixel value by averaging
def averaging(inputArray, filterArray):
    pixel_value = np.sum(np.multiply(inputArray, filterArray) / np.sum(filterArray))
    return pixel_value


# applies the filter on the image
def applyingFilter(inputArray, filterArray, filterSize):
    # zero padding
```

```
    padding = int((filterSize - 1) / 2)
    inputArray = np.pad(inputArray, padding)

    height, width = inputArray.shape

    outputArray = inputArray.copy()

    # iterate the original image
    for x in range(padding, height - padding):
        for y in range(padding, width - padding):
            # gets the part of the image the size of the filter matrix
            neighbourhood_array = inputArray[x - padding:x + padding + 1, y - padding: y + padding + 1]
            outputArray[x][y] = averaging(neighbourhood_array, filterArray)

    # removes padding from output
    outputArray = outputArray[padding:height - padding, padding: width - padding]

    return outputArray



# open image
input_image = Image.open("../Lab08/smoothing.tif").convert('L')
input_array = np.asarray(input_image)

# apply averaging
filter_size = 7
filter_array = generateFilter(filter_size)
output_array = applyingFilter(input_array, filter_array, filter_size)
```

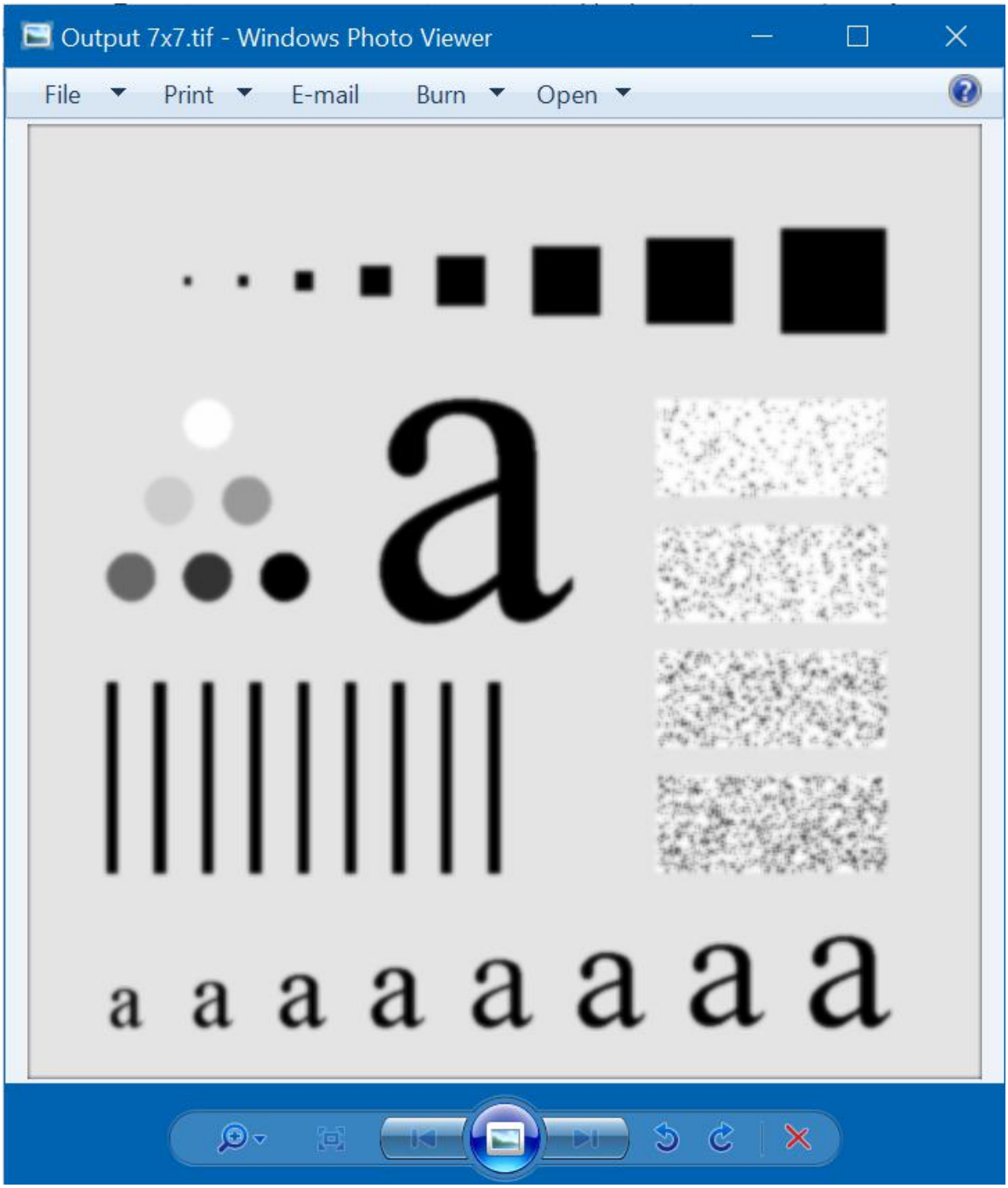
```
# make and save output image

output_image = Image.fromarray(output_array)

output_image.save("Output "+ str(filter_size) + "x" + str(filter_size) + ".tif")
```

All the code is the same as task 1a, except the filter is hard coded (line 6 - 12) and filter size (line 49)

```
1   from PIL import Image
2   import numpy as np
3
4   # returns the filter of size nxn
5   def generateFilter(filterSize):
6       filter_array = [[1,1,2,2,2,1,1],
7                       [1,2,2,4,2,2,1],
8                       [2,2,4,8,4,2,2],
9                       [2,4,8,16,8,4,2],
10                      [2,2,4,8,4,2,2],
11                      [1,2,2,4,2,2,1],
12                      [1,1,2,2,2,1,1]]
13
14      return filter_array
```

```
48  # apply averaging
49  filter_size = 7
50  filter_array = generateFilter(filter_size)
51  output_array = applyingFilter(input_array, filter_array, filter_size)
```

OUTPUT


Output 7x7.tif - Windows Photo Viewer
File
Print
E-mail
Burn
Open