```
In [1]: %reset -f
```

Honglu Xu
EECS531
HW 1
Due 02/21/2018

Filter implementation

1. Blurring filter:

Blurring filter is a filter to make the image looks more blurring, sometimes it is called smooth. For now, we have many ways to blur the image. We can blur the image either horizontally or vertically. We can use a matrix which only contains ones as the filter, or we can use the matrix that contains some gradients, such as the 'Gaussian filter', as the filter. In this experiment, we will try saveral ways to achieve the image blurring method.

Let's start with the easist method, 'box'. To achieve that, we need a filter that contains only ones. We can adjust the size starting from 3.

```
In [2]: import numpy as np
        from PIL import Image
```

```
In [3]: box_filter = np.ones((3,3),dtype=np.int)
        print(box_filter)
```

```
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

Now we have the filter, we can implement a function to modifiy the image with this filter.

```
In [4]: # read the image
        def read_image(image_name,method):
            image = Image.open(image_name)
            image = image.convert(method)
            image_data = np.array(image)
            image_data_row = len(image_data)
            image_data_col = len(image_data[0])
            return (image_data,image_data_row,image_data_col)
```

In [5]:
```
# give the image some extra space so it can fill the filter
def expend_image(image_information,myfilter):
    image_data = image_information[0]
    image_row = image_information[1]
    iamge_col = image_information[2]
    k = len(myfilter)
    increase = k // 2
    image_ex_data = np.zeros((image_row+increase*2,iamge_col+increase*2),dtype=np.uin
t8)
#     print(len(image_ex_data),image_row,iamge_col)
    image_ex_data[increase:(-increase),increase:(-increase)] = image_data
    return (image_ex_data,image_row+increase*2,iamge_col+increase*2)
```

In [6]:
```
def filtering(image_name,myfilter):
    image_information = read_image(image_name,'L')
    image_row = image_information[1]
    image_col = image_information[2]
    ex_iamge_information = expend_image(image_information,myfilter)
    ex_iamge_data = ex_iamge_information[0]
    ex_image_row = ex_iamge_information[1]
    ex_image_col = ex_iamge_information[2]
    k = len(myfilter)
    increase = k // 2
    temp_matrix = np.zeros((k,k),dtype=np.uint8)
    data_matrix = np.zeros((k,k),dtype=np.uint8)
    result = 0
    result_image_data = np.zeros((image_row,image_col),dtype=np.uint8)
    kernal_sum = myfilter.sum()
    for i in range(increase,ex_image_row-increase):
        for j in range(increase,ex_image_col-increase):
            data_matrix = ex_iamge_data[i-increase:i+increase+1,j-increase:j+increase
+1]
            temp_matrix = data_matrix*myfilter
            result = temp_matrix.sum()
            result = result*(1/kernal_sum)
            result_image_data[i-increase,j-increase] = result
    return (result_image_data,image_row,image_col)
```

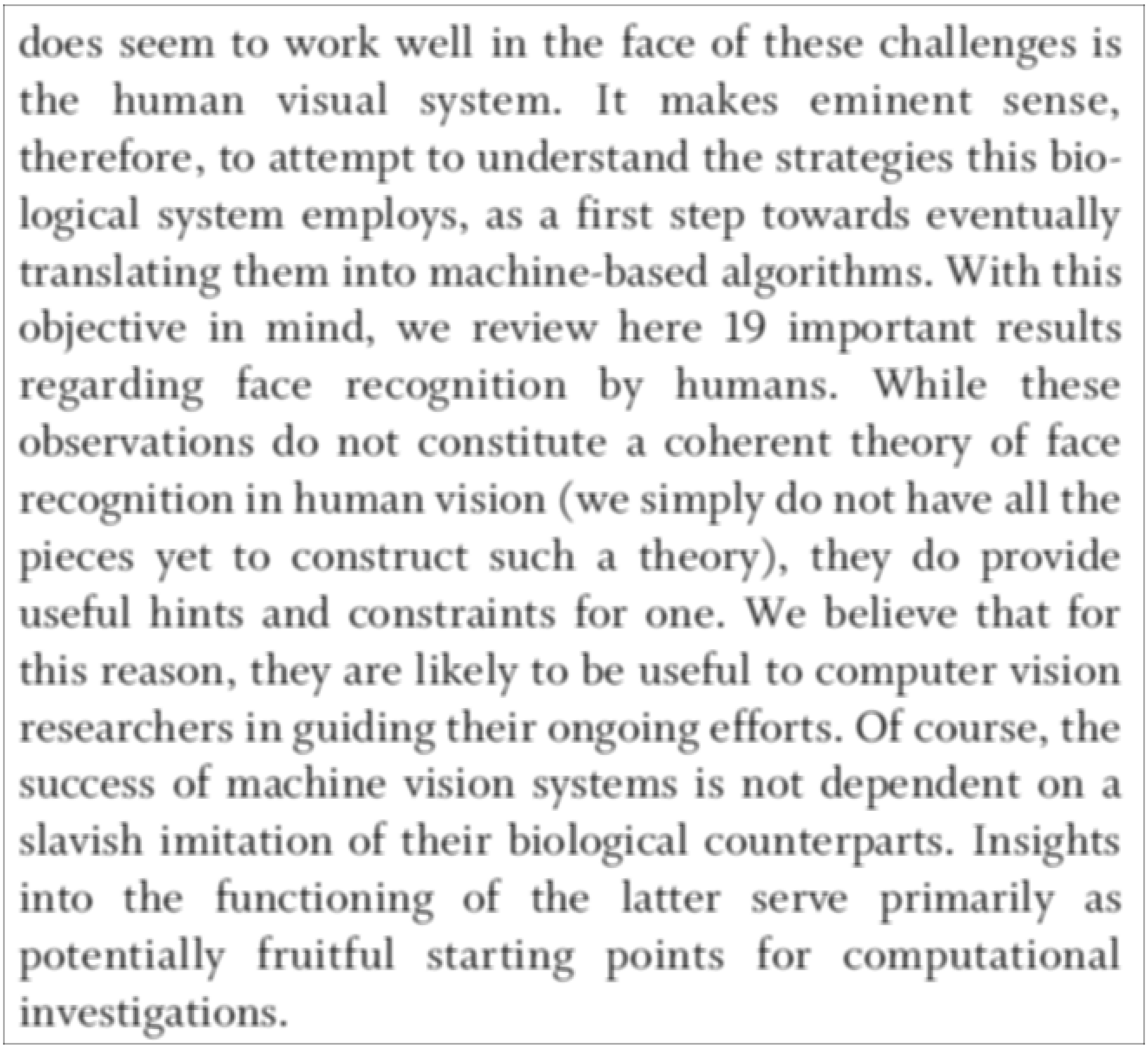
In [7]:
```
result_image = filtering('characters.png',box_filter)
```

In [8]:
```
img = Image.fromarray(result_image[0], 'L')
# img.show()
img.save('box_filter3.png')
```

Oringinal image:

does seem to work well in the face of these challenges is the human visual system. It makes eminent sense, therefore, to attempt to understand the strategies this biological system employs, as a first step towards eventually translating them into machine-based algorithms. With this objective in mind, we review here 19 important results regarding face recognition by humans. While these observations do not constitute a coherent theory of face recognition in human vision (we simply do not have all the pieces yet to construct such a theory), they do provide useful hints and constraints for one. We believe that for this reason, they are likely to be useful to computer vision researchers in guiding their ongoing efforts. Of course, the success of machine vision systems is not dependent on a slavish imitation of their biological counterparts. Insights into the functioning of the latter serve primarily as potentially fruitful starting points for computational investigations.

Result:

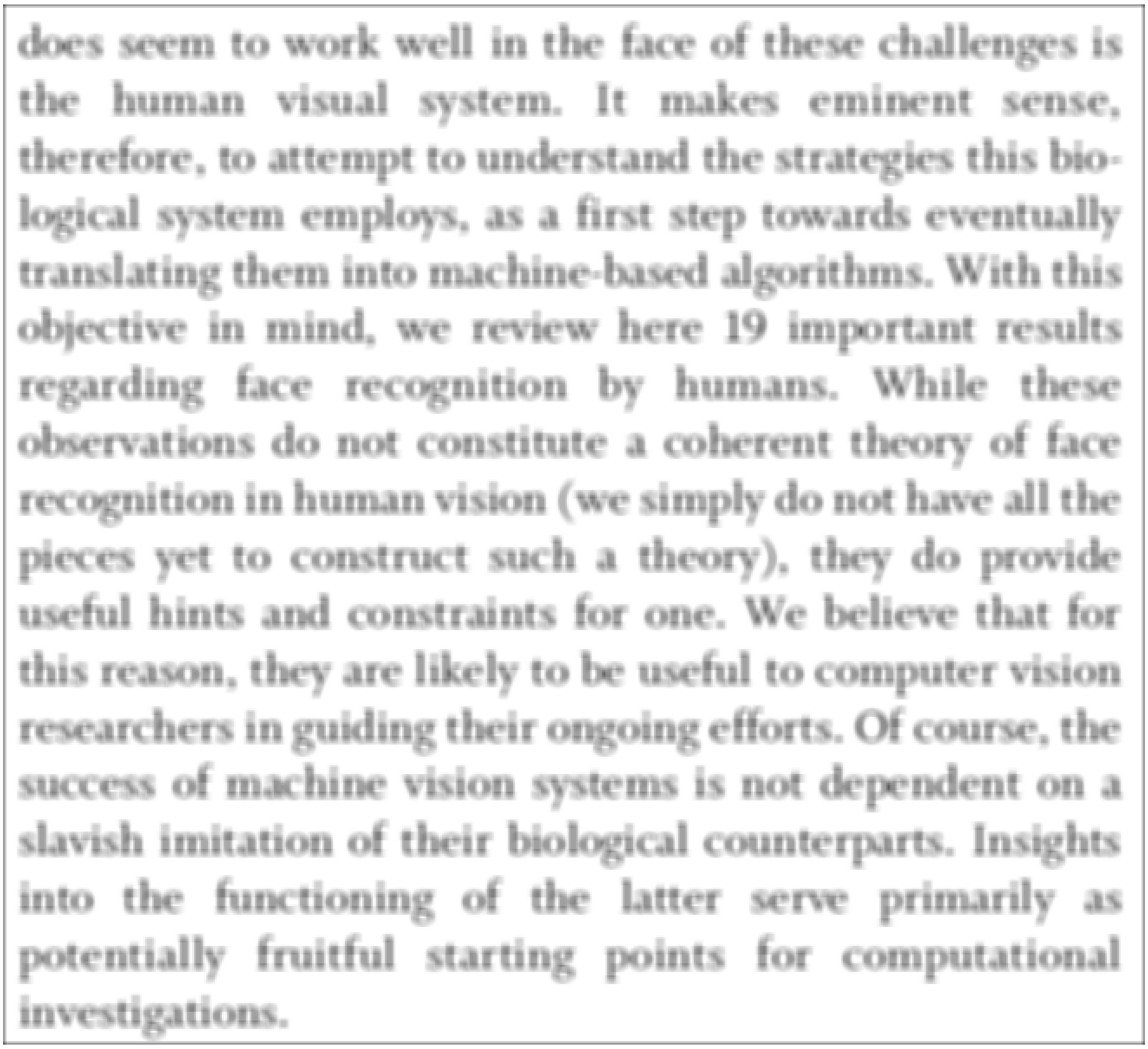

does seem to work well in the face of these challenges is the human visual system. It makes eminent sense, therefore, to attempt to understand the strategies this biological system employs, as a first step towards eventually translating them into machine-based algorithms. With this objective in mind, we review here 19 important results regarding face recognition by humans. While these observations do not constitute a coherent theory of face recognition in human vision (we simply do not have all the pieces yet to construct such a theory), they do provide useful hints and constraints for one. We believe that for this reason, they are likely to be useful to computer vision researchers in guiding their ongoing efforts. Of course, the success of machine vision systems is not dependent on a slavish imitation of their biological counterparts. Insights into the functioning of the latter serve primarily as potentially fruitful starting points for computational investigations.

Let's try some larger filter.

In [9]:
```
box_filter5 = np.ones((5,5),dtype=np.int)
print(box_filter5)
```

```
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
```

In [10]:
```
result_image_box5 = filtering('characters.png',box_filter5)
img = Image.fromarray(result_image_box5[0], 'L')
# img.show()
img.save('box_filter5.png')
```


does seem to work well in the face of these challenges is the human visual system. It makes eminent sense, therefore, to attempt to understand the strategies this biological system employs, as a first step towards eventually translating them into machine-based algorithms. With this objective in mind, we review here 19 important results regarding face recognition by humans. While these observations do not constitute a coherent theory of face recognition in human vision (we simply do not have all the pieces yet to construct such a theory), they do provide useful hints and constraints for one. We believe that for this reason, they are likely to be useful to computer vision researchers in guiding their ongoing efforts. Of course, the success of machine vision systems is not dependent on a slavish imitation of their biological counterparts. Insights into the functioning of the latter serve primarily as potentially fruitful starting points for computational investigations.

We can see it worked very well, and, as the filter size increases the blurring effect increases.

I will explain the theory now.

Let's say we have a image of size 3*3

In [11]:
```
np.random.seed(0)
test_image = np.random.random((3,3))
test_image = test_image*100
print(test_image)
```

```
[[54.88135039 71.51893664 60.27633761]
 [54.4883183  42.36547993 64.58941131]
 [43.75872113 89.17730008 96.36627605]]
```

And we have a filter of 3*3

```
In [12]: print(box_filter)
```

```
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

To let the filter go through all the elements in the image, we need to expand the image.

```
In [13]: test_image_ex = expend_image((test_image,3,3),box_filter)
         print(test_image_ex[0])
```

```
[[ 0  0  0  0  0]
 [ 0 54 71 60  0]
 [ 0 54 42 64  0]
 [ 0 43 89 96  0]
 [ 0  0  0  0  0]]
```

We can see it is surrounded by 0s now.
Then, we got all the size 3*3 sub-matrix from the expanded image and mutiply the data with the filter. We got a new 3*3 matrix, and we can sum all the elements in that matirx and divided by 1/(k^2), which k is the size of the kernal, k=3 for this situation.
The first 3*3 matrix will be:

```
In [14]: print(test_image_ex[0][0:3,0:3])
```

```
[[ 0  0  0]
 [ 0 54 71]
 [ 0 54 42]]
```

And the result matrix will be:

```
In [15]: tes_res = test_image_ex[0][0:3,0:3] * box_filter
         print(tes_res)
```

```
[[ 0  0  0]
 [ 0 54 71]
 [ 0 54 42]]
```

Now, we sum the elements and divided it by the kernal sum.

```
In [16]: tes_res.sum()/(box_filter.sum())

Out[16]: 24.555555555555557
```

Finally, we can apply this result value back to where it should be. In this example the 54 will replaced by 25

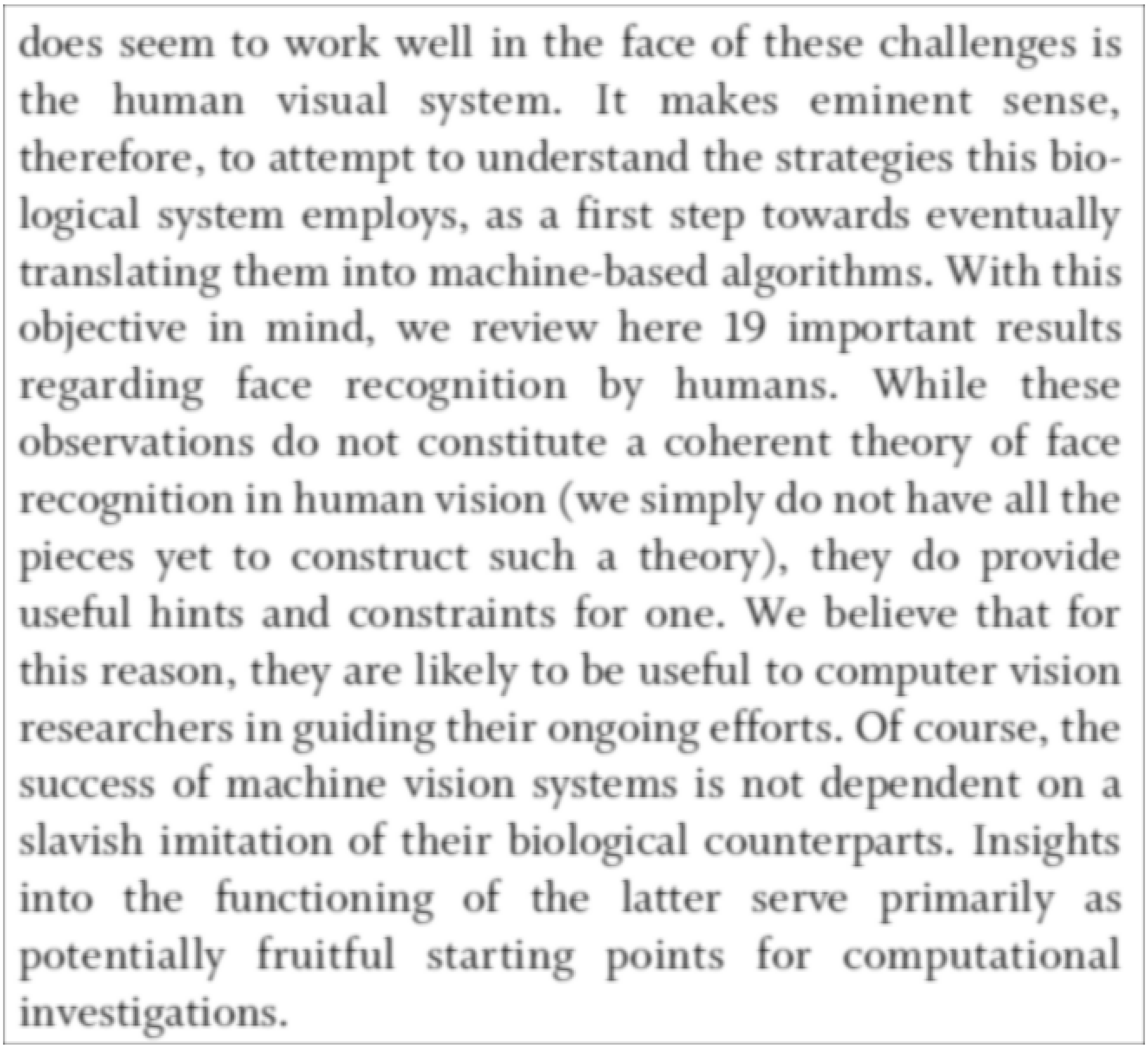
Now we have the functions we can try other filters.
Let's make a 5*5 'Gaussian' filter:

```
In [17]: gaussian_filter5 = np.array([[1,4,6,4,1],[2,8,12,8,2],[6,24,32,24,6],[2,8,12,8,2],[1,
         4,6,4,1]])
         print(gaussian_filter5)
```

```
[[ 1  4  6  4  1]
 [ 2  8 12  8  2]
 [ 6 24 32 24  6]
 [ 2  8 12  8  2]
 [ 1  4  6  4  1]]
```

And appley the filtering:

```
In [18]: result_image_gaussian5 = filtering('characters.png',gaussian_filter5)
         img = Image.fromarray(result_image_gaussian5[0], 'L')
         # img.show()
         img.save('gaussian_filter5.png')
```



does seem to work well in the face of these challenges is the human visual system. It makes eminent sense, therefore, to attempt to understand the strategies this biological system employs, as a first step towards eventually translating them into machine-based algorithms. With this objective in mind, we review here 19 important results regarding face recognition by humans. While these observations do not constitute a coherent theory of face recognition in human vision (we simply do not have all the pieces yet to construct such a theory), they do provide useful hints and constraints for one. We believe that for this reason, they are likely to be useful to computer vision researchers in guiding their ongoing efforts. Of course, the success of machine vision systems is not dependent on a slavish imitation of their biological counterparts. Insights into the functioning of the latter serve primarily as potentially fruitful starting points for computational investigations.

It seems well, too. Compared to the same size of the box filter, I think the box filter is more blurring, but in this filter, it blurring the iamge with some gradient, it is not mixing all the pixels together, but cutting the edges out.