

CS195: Computer Vision

Image Filtering

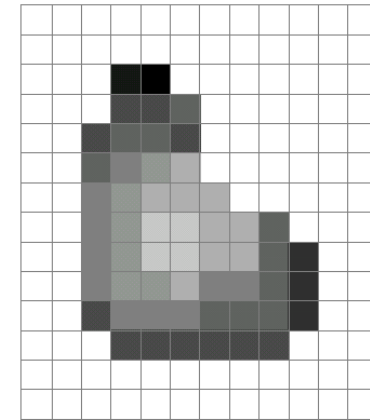
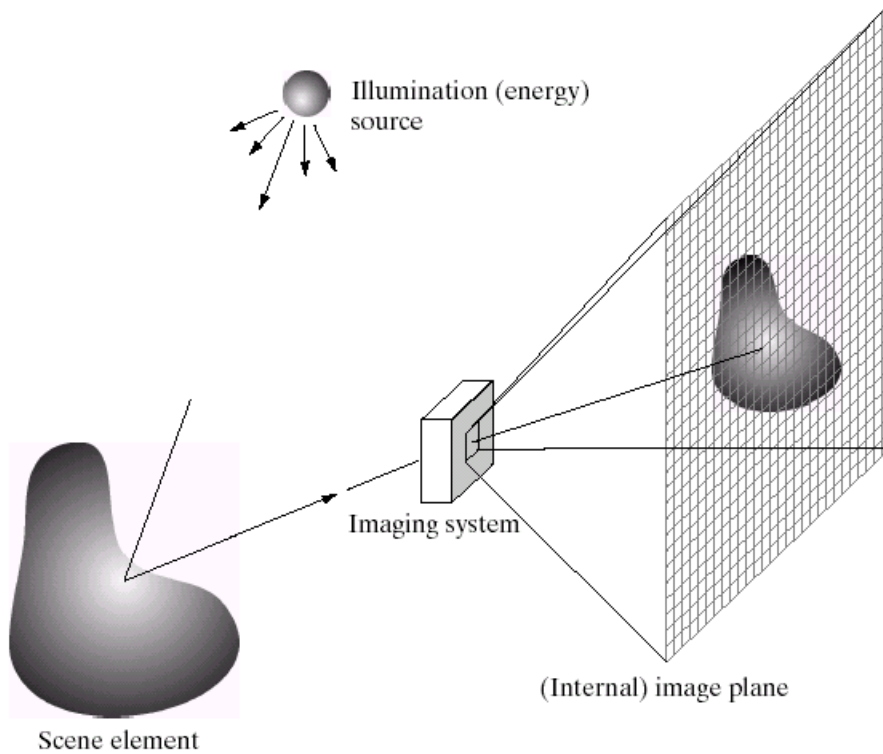
February 01, 2022

Md Alimoor Reza

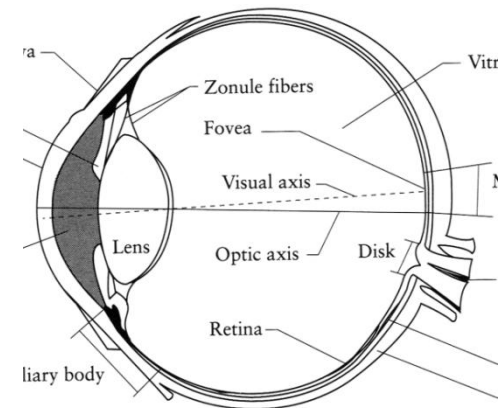
Assistant Professor of Computer Science



What is an image?

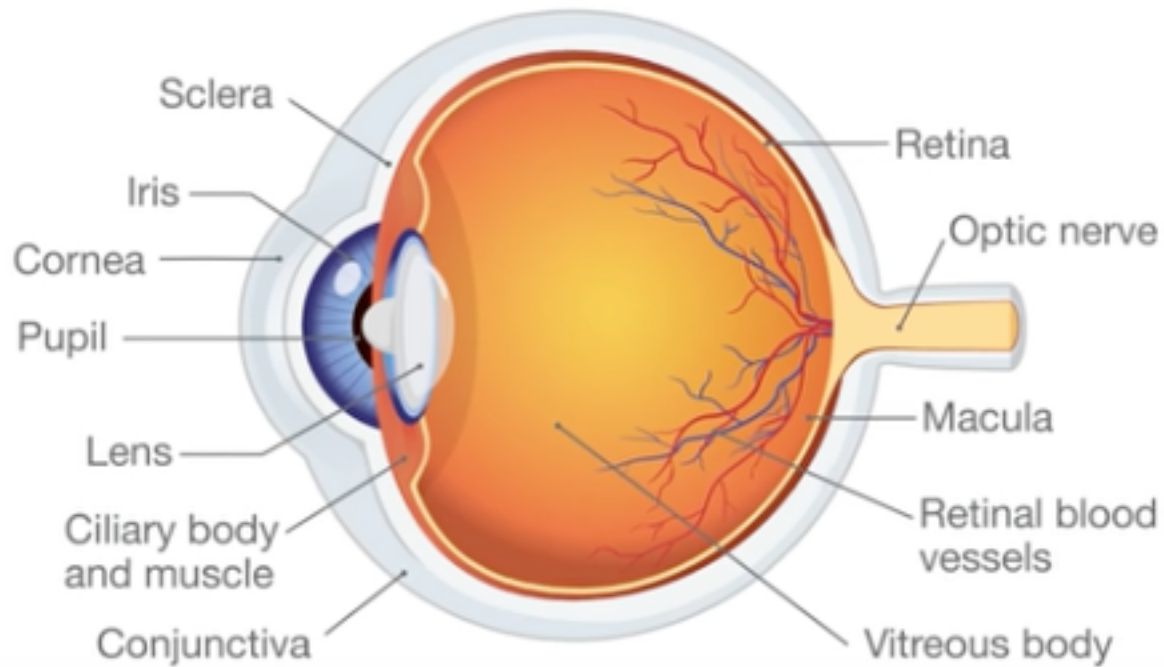


Digital Camera

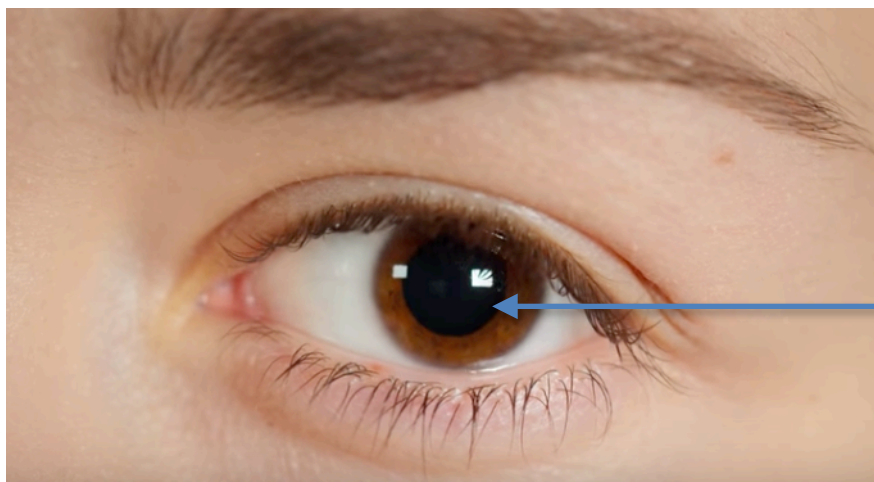


The Eye Source: A. Efros

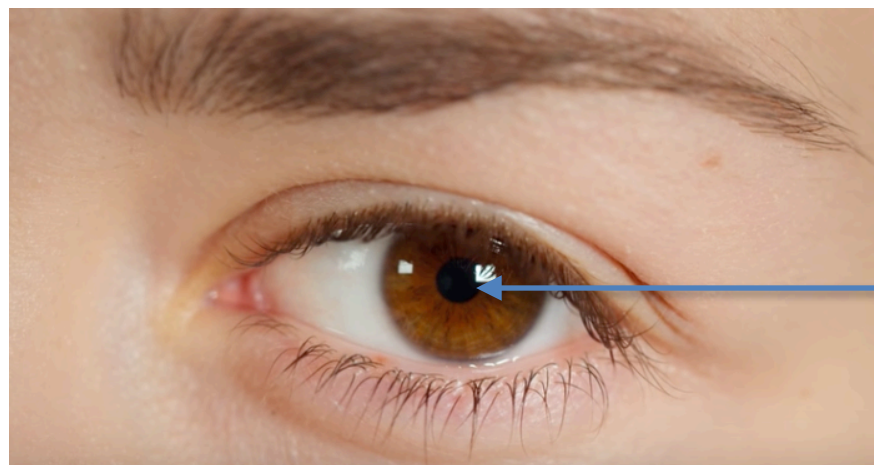
Human vision



Human vision

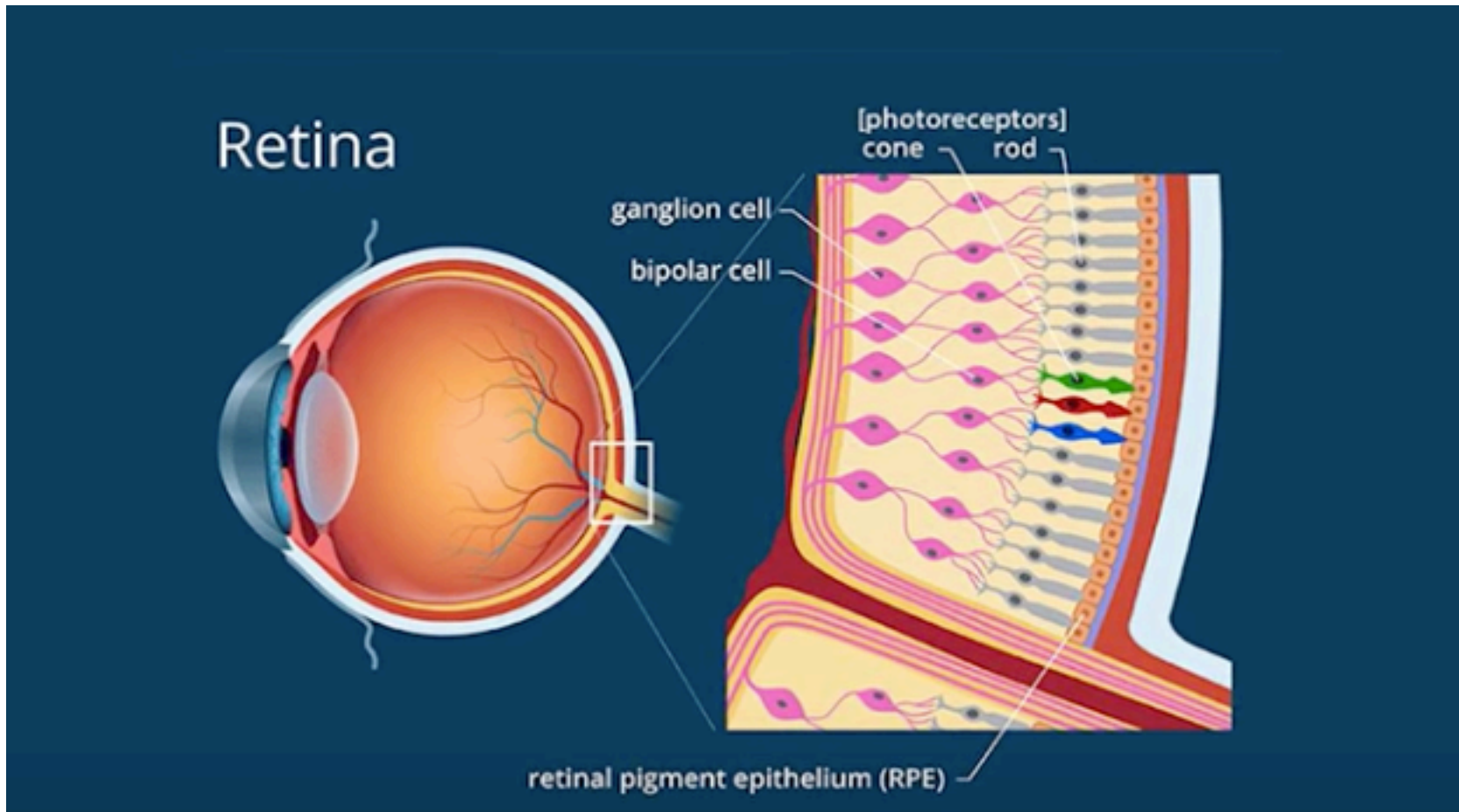


In the dark

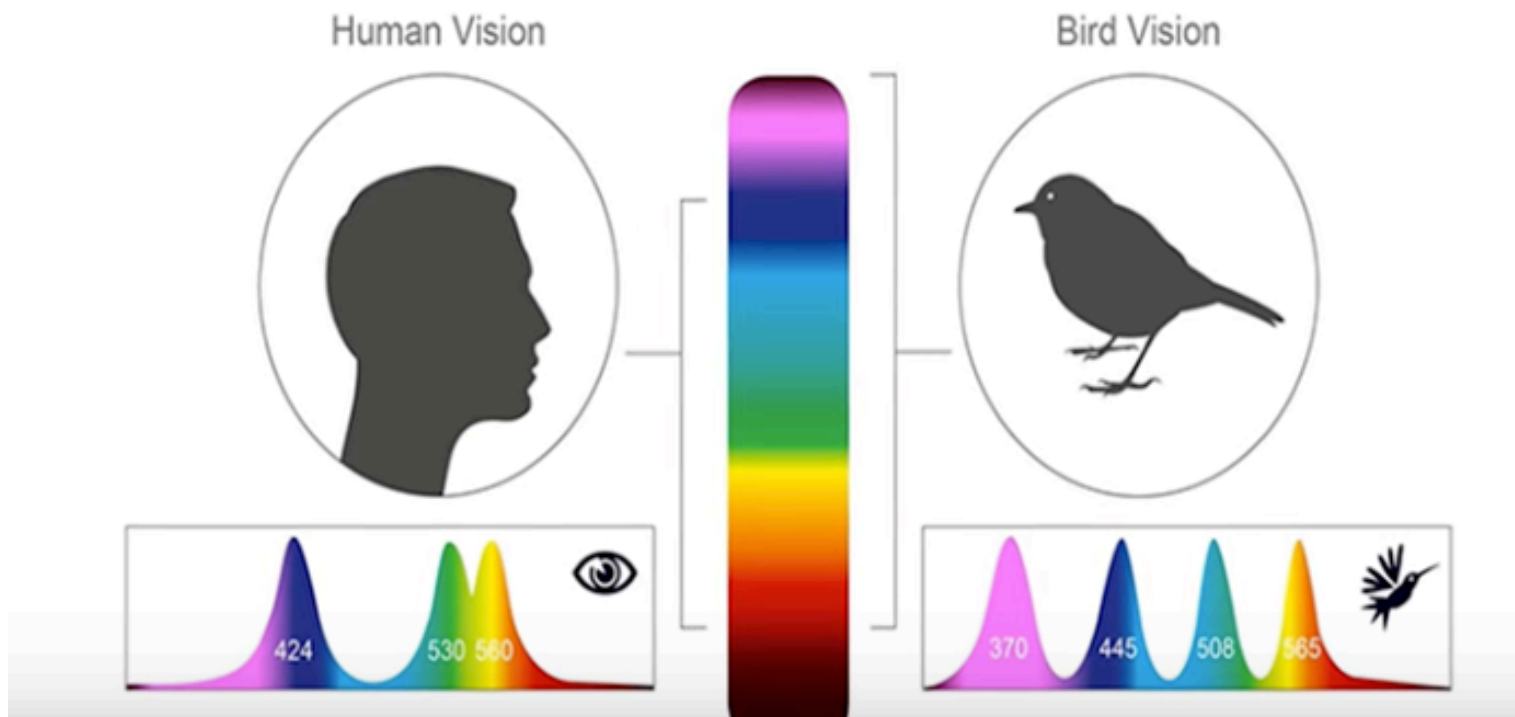


In bright light

Human vision

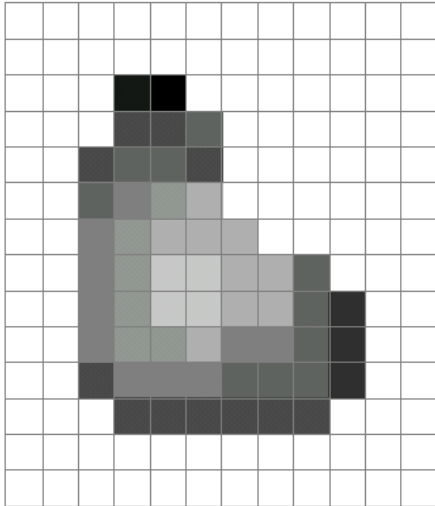


Human vision



What is an image?

- A grid of intensity values



=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

Red

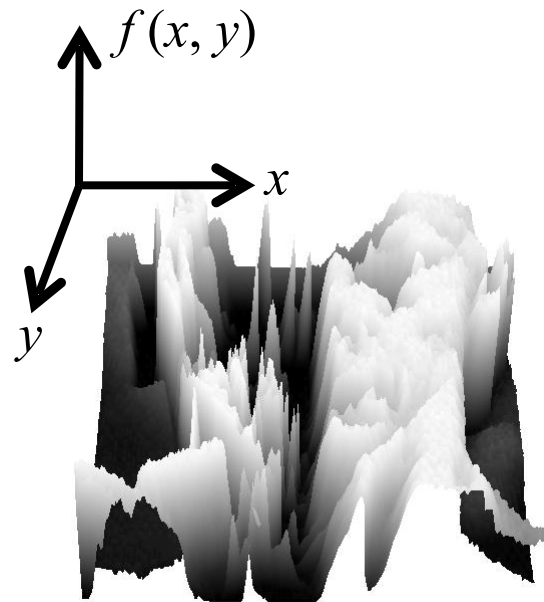


Green

Blue

What is a (grayscale) image?

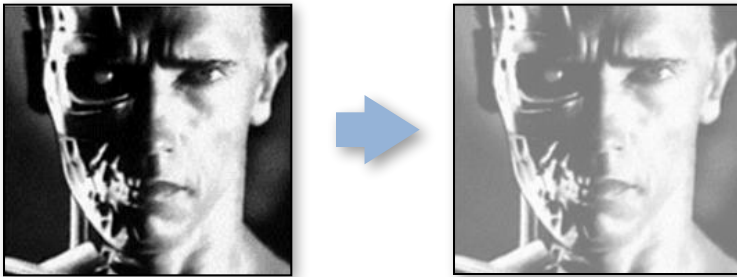
- A **2-d function** $f: \mathbb{R}^2 \rightarrow \mathbb{R}$
 - $f(x, y)$ gives the **grayscale intensity** at position (x, y)



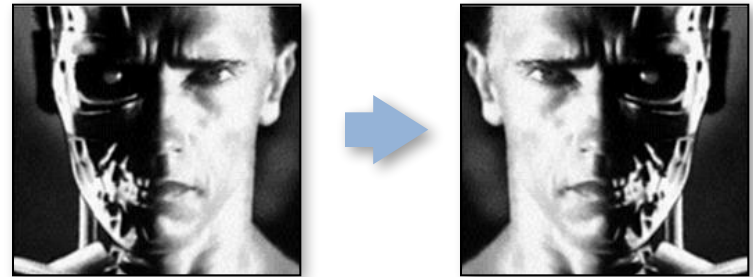
- A **digital image** is a discrete (**domain is sampled, range is quantized**) version of this function

Image transformations

- As with any function, we can apply operators to an image



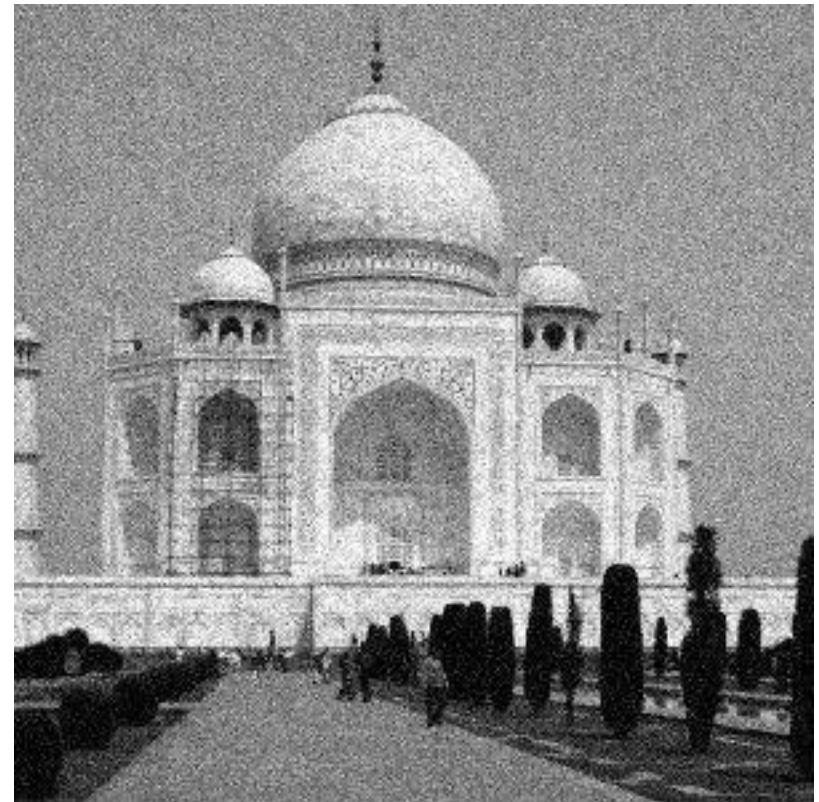
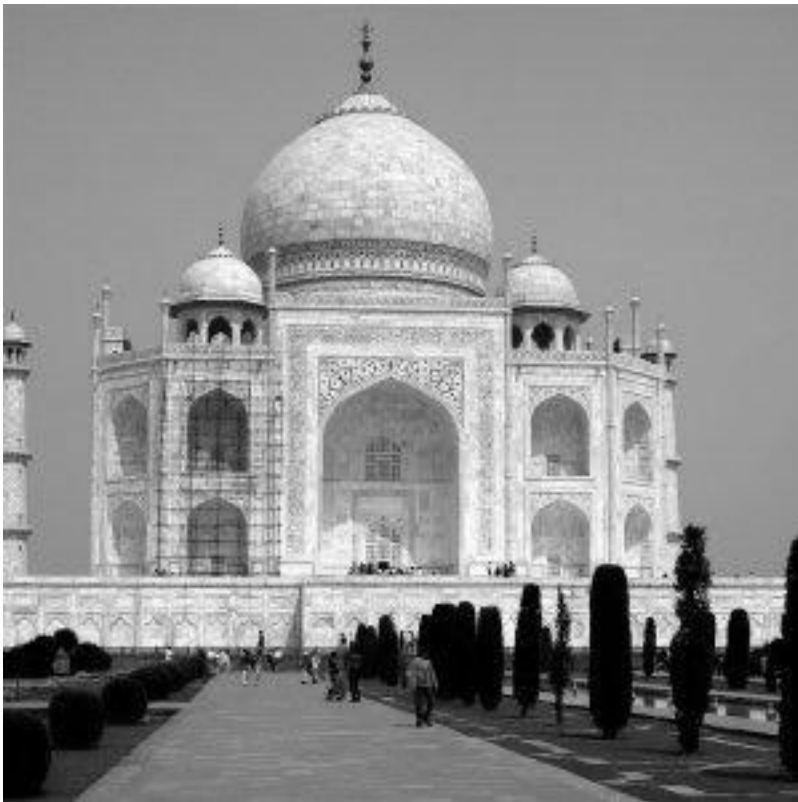
$$g(x,y) = f(x,y) + 20$$



$$g(x,y) = f(-x,y)$$

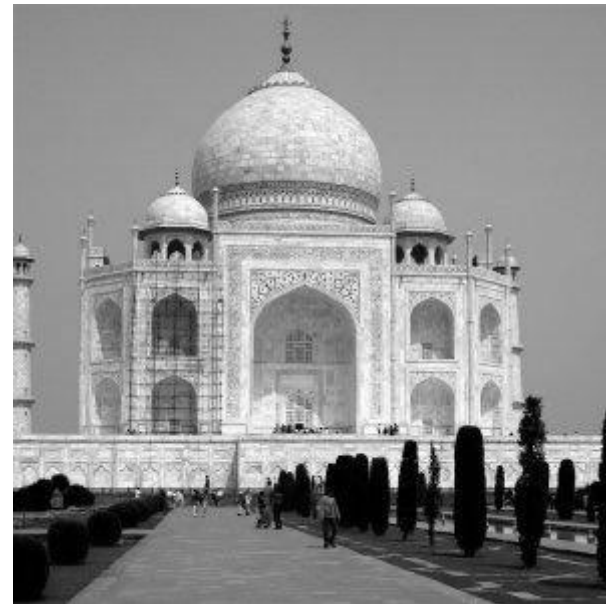
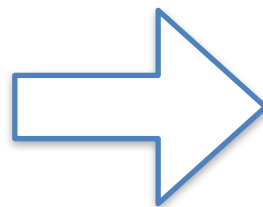
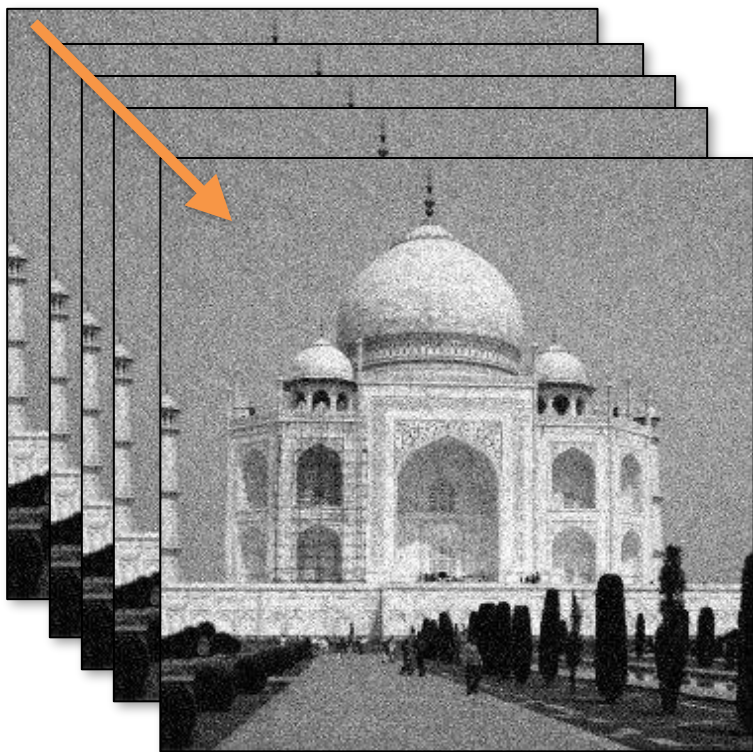
Image noise

- Imaging systems (sensor, lens, compression algorithm, etc.) add unwanted noise



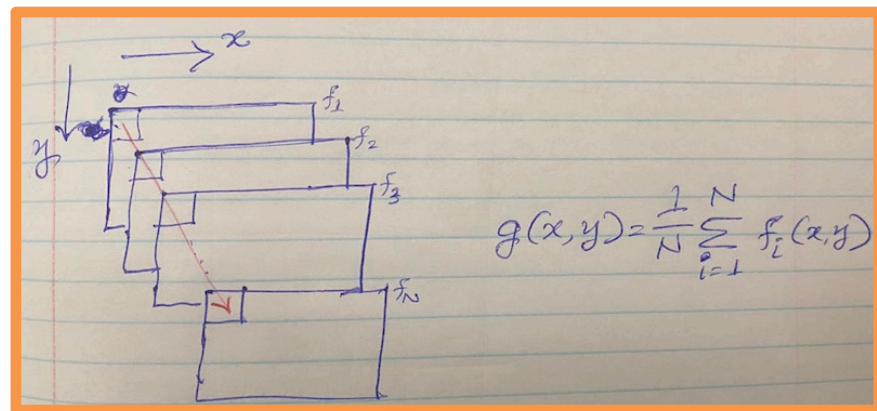
Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?



Take lots of images and average them!

What's the next best thing?



Question: What assumptions need to be correct for proposed averaging method?

- Given a camera and a still scene, we assume a particular noise model
 - Additive Gaussian Noise

$$f_{\text{noise}}(x, y) = f_{\text{actual}}(x, y) + \epsilon$$

where, $\epsilon = \mathcal{N}(0, \sigma^2)$

zero mean sigma standard deviation

$$\mathcal{N}(0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

10	5	3
4	5	1
1	1	7

Local image data

Some function



	7	

Modified image data

Mean filtering

 $F[.,.]$

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

 $G[.,.]$

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

Mean filtering

$F[.,.]$

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

$G[.,.]$

	0	10							

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

Mean filtering

 $F[.,.]$

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

 $G[.,.]$

	0	10	20						

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

Mean filtering

 $F[.,.]$

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

 $G[.,.]$

	0	10	20	30					

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

Mean filtering

 $F[.,.]$

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

 $G[.,.]$

	0	10	20	30	30				

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

Mean filtering

$F[.,.]$

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

$G[.,.]$

	0	10	20	30	30				

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

Mean filtering

 $F[.,.]$

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

 $G[.,.]$

	0	10	20	30	30				
							?		
					50				

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

Mean filtering

 $F[.,.]$

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

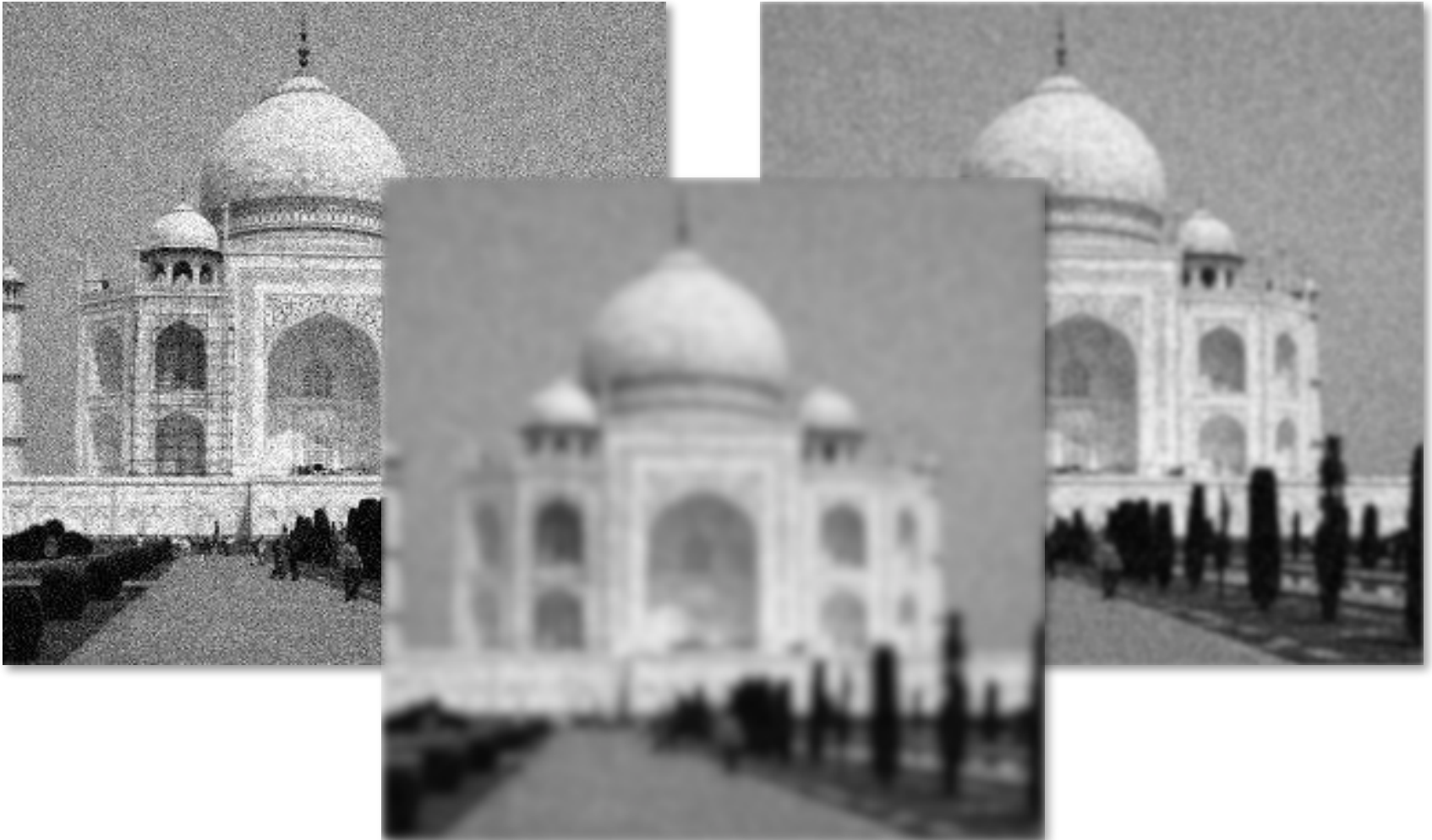
 $G[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$G(i, j) = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F(u + i, v + j)$$

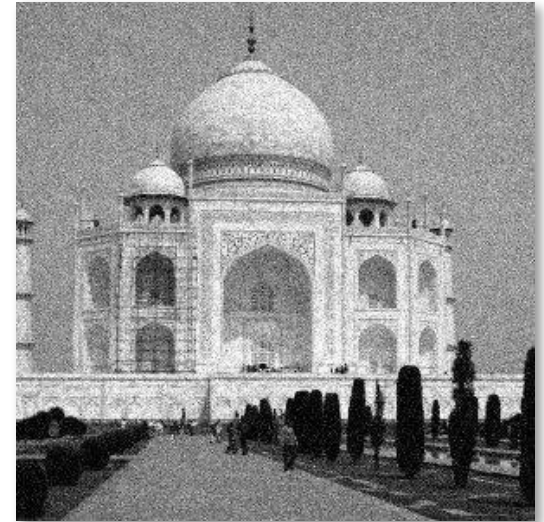
Mean filtering

- Larger k \Rightarrow more blurring



Mean filtering

- Larger k \Rightarrow more blurring



Mean filtering $k=1$



Mean filtering $k=3$



Mean filtering $k=5$

Cross-correlation

The mean filter is just a specific case of a *cross-correlation*, a very general operation.

Let F an image, H be a kernel (of size $2k+1 \times 2k+1$), then the cross-correlation of F with H is:

$$G = H \otimes F$$

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v]$$

Linear filtering

- More general version: linear filtering (cross-correlation, convolution)
 - Replace each pixel by a *linear combination* of its neighbors

• MEAN FILTER = SPECIFIC CASE OF CROSS-CORRELATION

$H = \text{KERNEL} \in \mathbb{R}^{2K+1}$

$G = H \otimes F$

$G[u, v] = \sum_{i=-K}^K \sum_{j=-K}^K H[i, j] * F[i+u, j+v]$

•

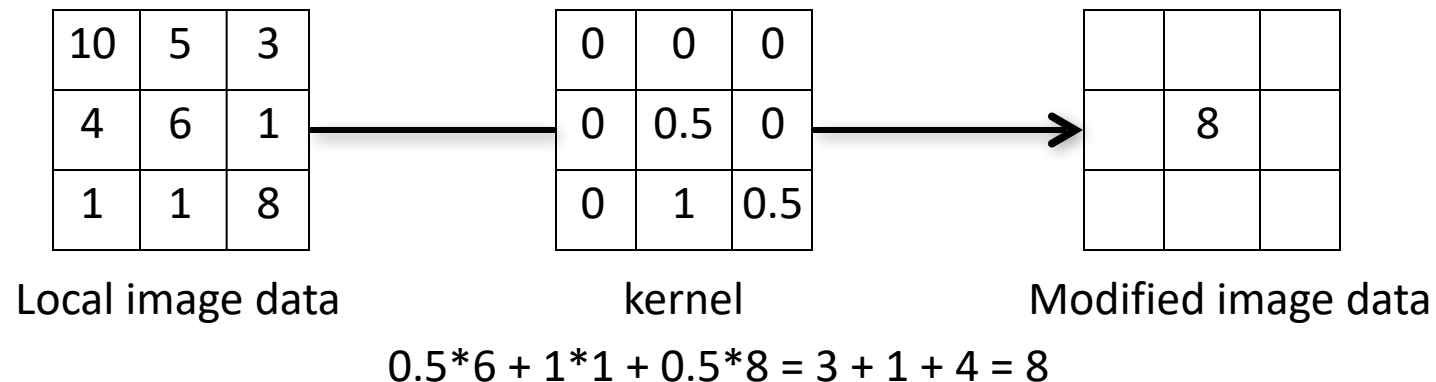
```

        graph TD
            LF[LINEAR FILTERING] --> CC[CROSS-CORR]
            LF --> CONV[CONVOLUTION]
            CC --> MF[MEAN FILTER]
            CC --> F3[1/3 FILTER]
            CC -.-> Dots1[...]
            CONV -.-> Dots2[...]
        
```

KERNEL / H
 MASK /
 FILTER /

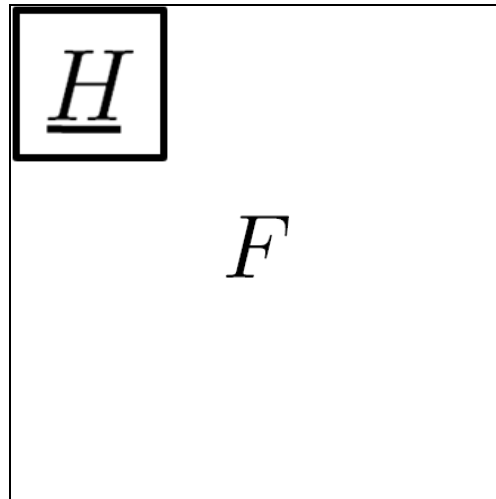
Linear filtering

- More general version: linear filtering (cross-correlation, convolution)
 - Replace each pixel by a *linear combination* of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)



Source: L. Zhang

Cross correlation



Adapted from F. Durand

Cross-correlation examples

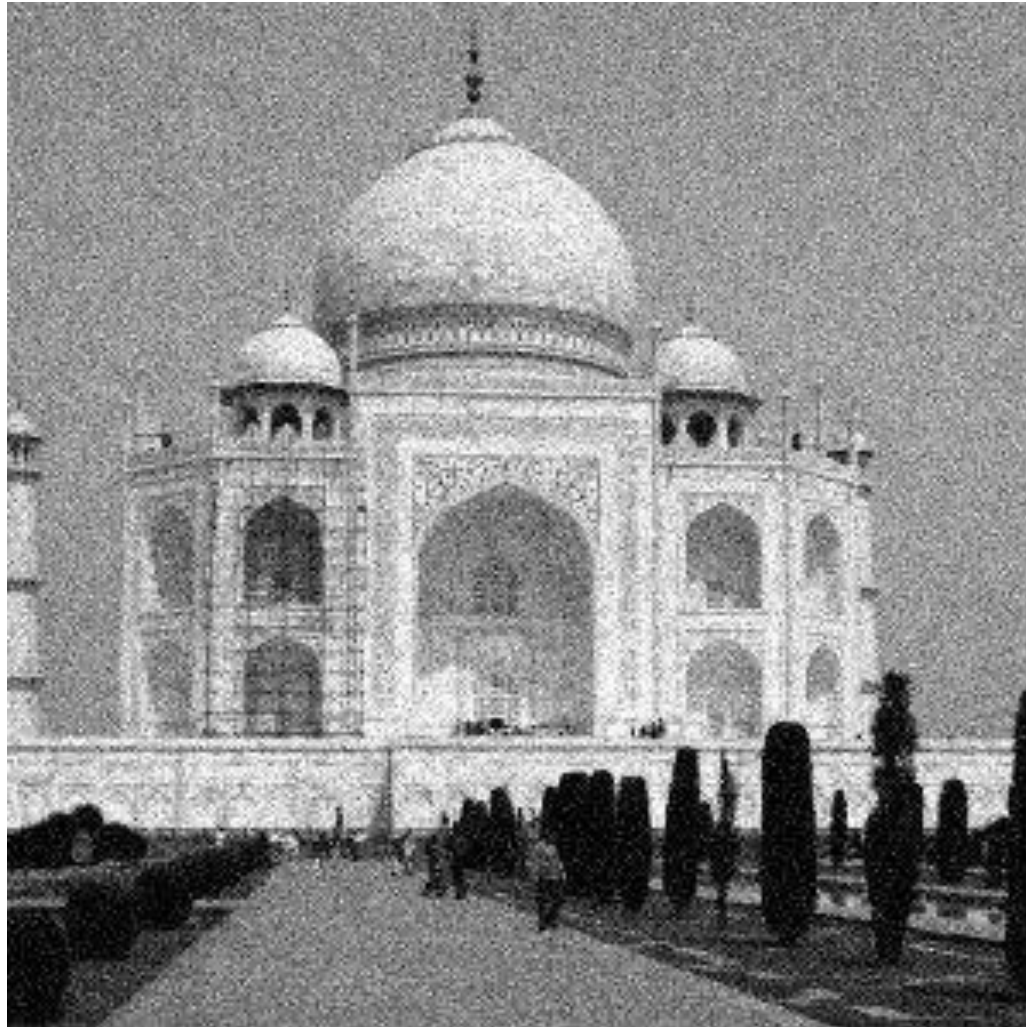
$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \otimes \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & a & b & c & 0 \\ \hline 0 & d & e & f & 0 \\ \hline 0 & g & h & i & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} =$$

H F

$$\begin{array}{|c|c|c|} \hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array} \otimes \begin{array}{|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} =$$

H F

Coding activity: cross-correlation



Coding activity: cross-correlation

```
# Author's name: Md Alimoor Reza
# Author's contact: md.reza@drake.edu
# Date: (February 01, 2022)

# copy these two lines if want to mount your drive and read data from your google-drive
'''
from google.colab import drive
drive.mount('/content/drive')
'''

import matplotlib.pyplot as plt
from PIL import Image, ImageOps
import numpy as np
import pdb
import sys

img_pil = Image.open('/content/drive/MyDrive/cs195_sp22/week2/tajmahal_noisy.png')
img_pil_array = np.asarray(img_pil)

plt.figure(figsize=(4,4)) # figure size (8 inch, 8 inch)
plt.imshow(img_pil_array, cmap='gray')
plt.title('Tajmahal noisy')
img_pil_array.shape

k = 1
kernel_size= 2*k+1 # kernel of size (2k+1, 2k+1)
H = np.ones((kernel_size, kernel_size))
H = H/(kernel_size**2)

sum(H.flatten())

rows = img_pil_array.shape[0]
cols = img_pil_array.shape[1]
print(H, '\n', rows, cols)
```

Coding activity: cross-correlation

```
# kernel introspection
'''
for v_row in range(-k,k+1):
    for u_col in range(-k,k+1):
        cur_kernel_value = H[v_row + k, u_col + k]
        # small trick to adjust the indexing so that does not go outside (negative indexing is not permitted)
        print(cur_kernel_value, ", ", end="")
    print('')
'''

# computer cross-correlation
F = img_pil
new_img = img_pil

for y in range(kernel_size, rows-kernel_size):
    for x in range(kernel_size, cols-kernel_size):

        # compute cross-correlation centered at pixel location (x, y) s
        old_pixel_value = F.getpixel((x, y))
        new_pixel_value = 0.0

        for v_row in range(-k,k+1):
            for u_col in range(-k,k+1):
                # YOUR CODE HERE (at most 3 three lines)

        # update the value at location (x,y) with newly computed pixel value
        new_img.putpixel((x, y), round(new_pixel_value))
        #print('old_value=', old_pixel_value, ': new_value=', round(new_pixel_value))

fig, axes = plt.subplots(1,2, figsize=(16, 16))
ax = axes[0]
ax.imshow(img_pil_array, cmap='gray')
ax.set_title('Original noisy image')
ax = axes[1]
ax.imshow(new_img, cmap='gray')
ax.set_title('Mean filtered image with k = ' + str(k))
new_img.save('tajmahal_mean_filter_k_' + str(k) + '.png')
```


More activities

- In class activities: Implement cross-correlation Filtering with the following Kernel.

.33	0	0
.33	0	0
.33	0	0

Kernel 1

0	0	.33
0	.33	0
.33	0	0

Kernel 2

.33	.33	.33
0	0	0
0	0	0

Kernel 3

.33	0	0
0	.33	0
0	0	.33

Kernel 4