CS181: Computer Vision

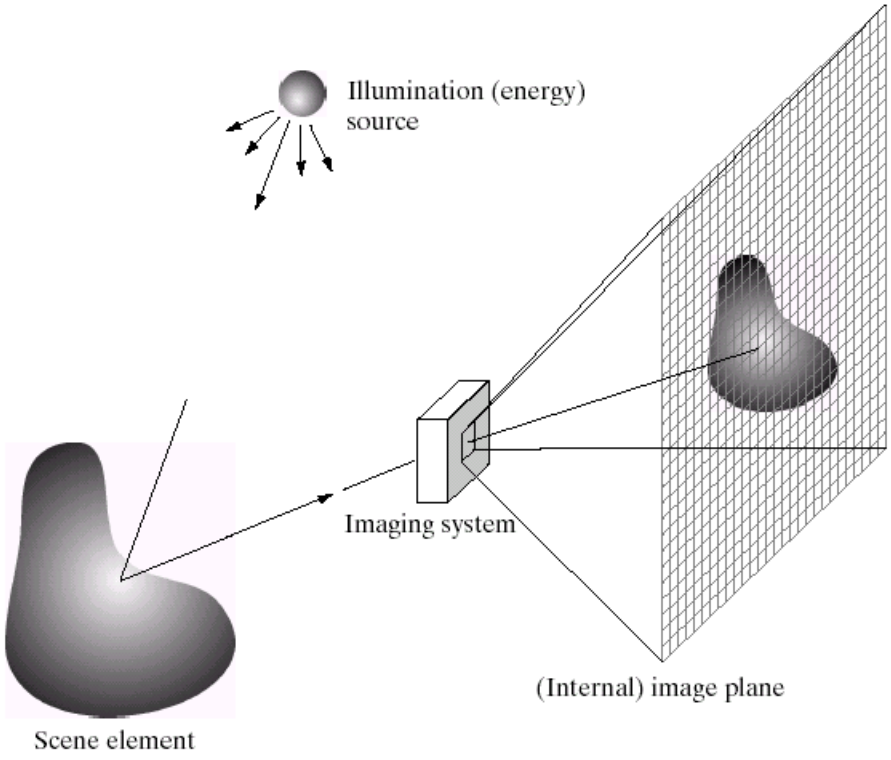
Image Filtering

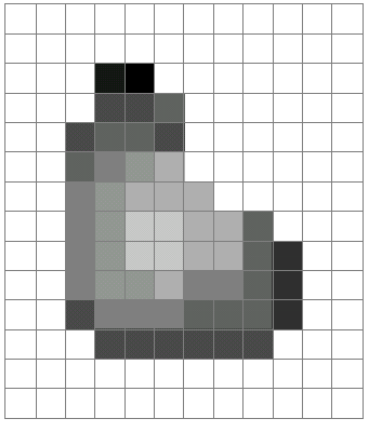
Cross-correlation

August 27, 2026

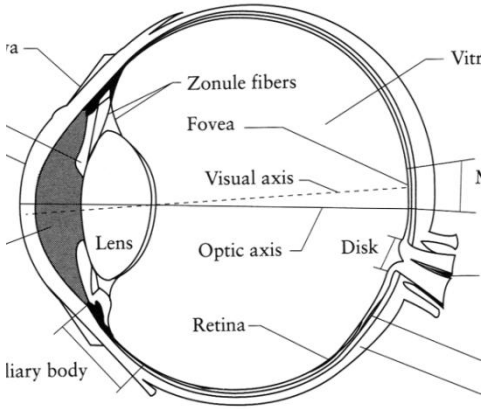
Md Alimoor Reza
Associate Professor of Computer Science

What is an image?



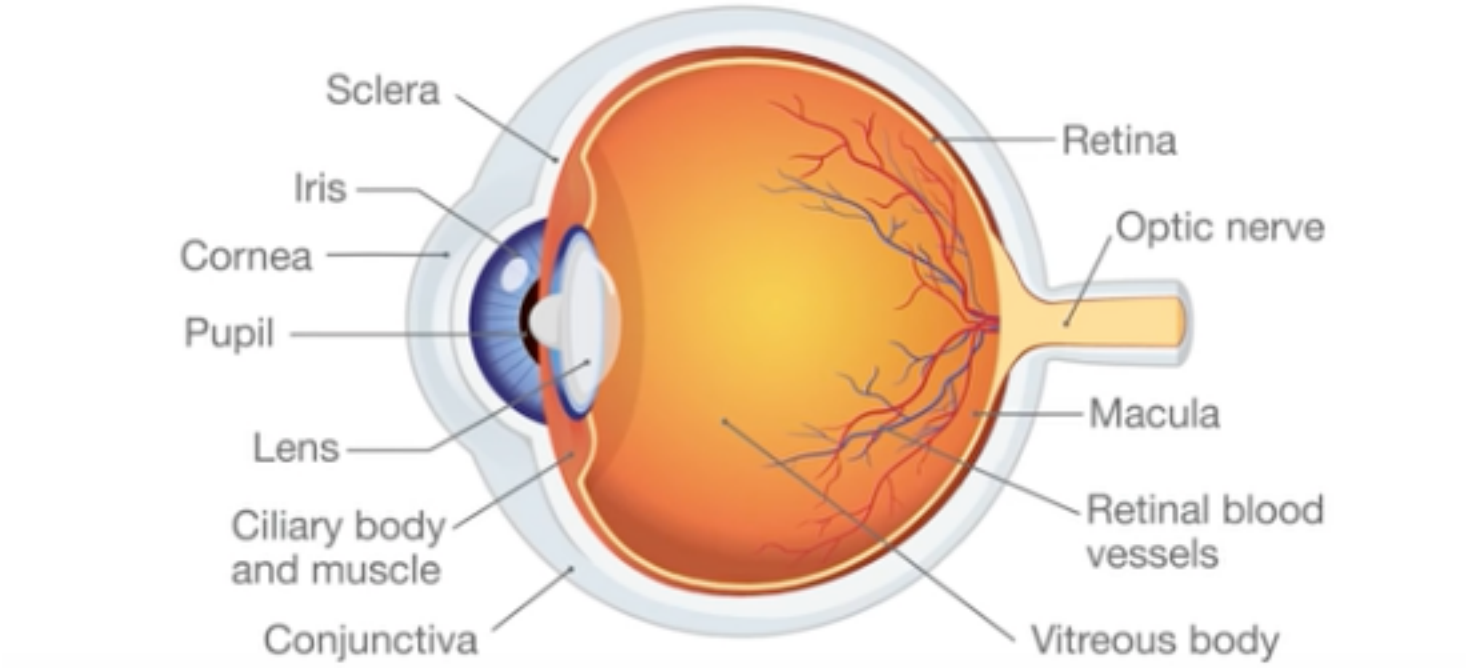


Digital Camera

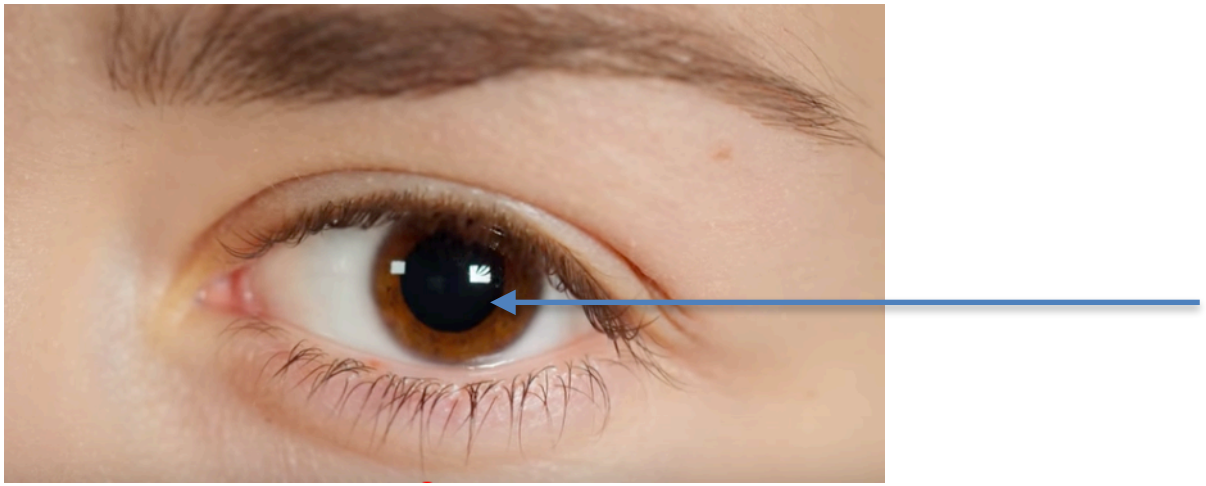


The Eye **Source: A. Efros**

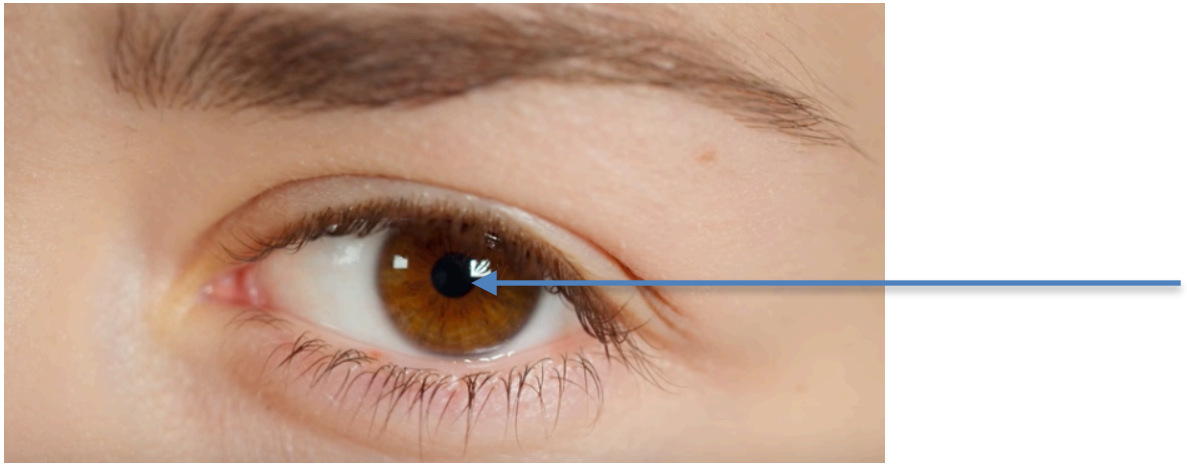
Human vision



Human vision

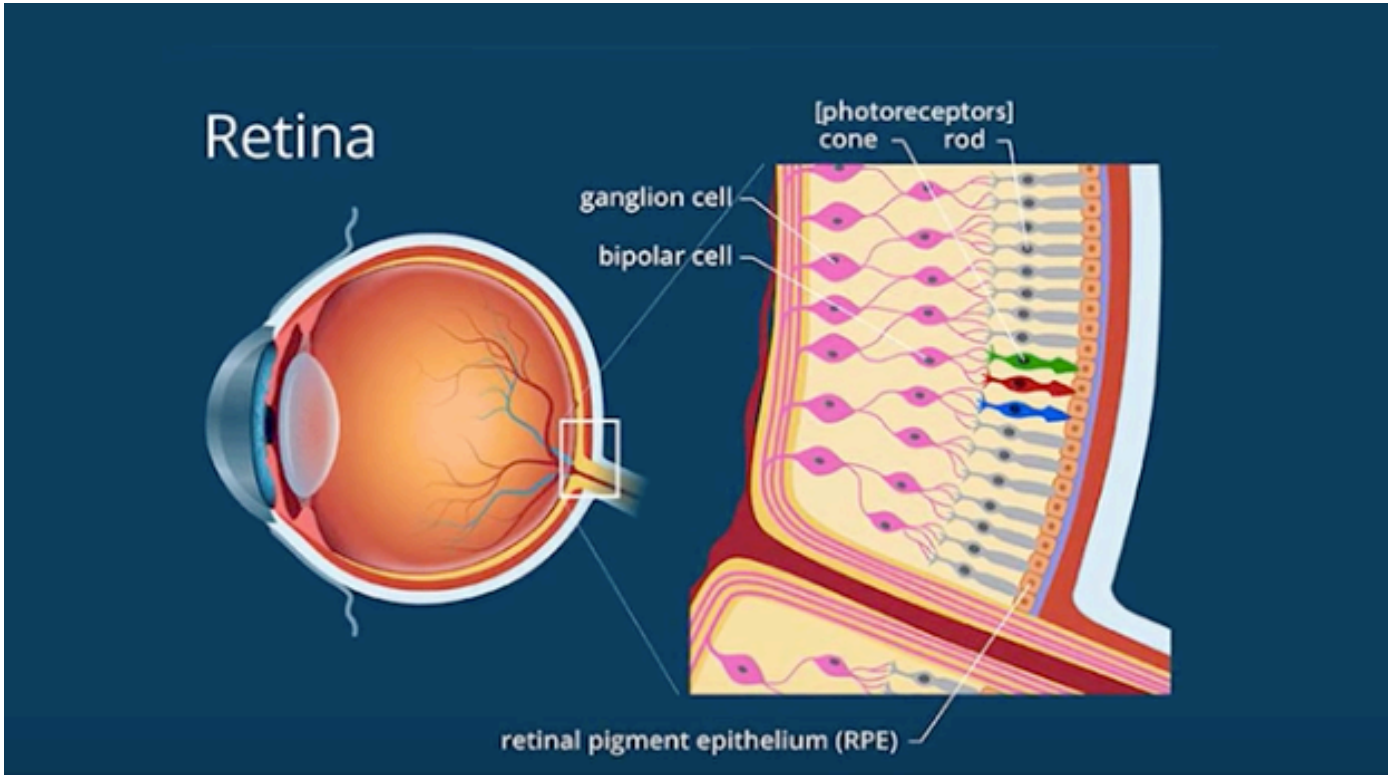


In the dark

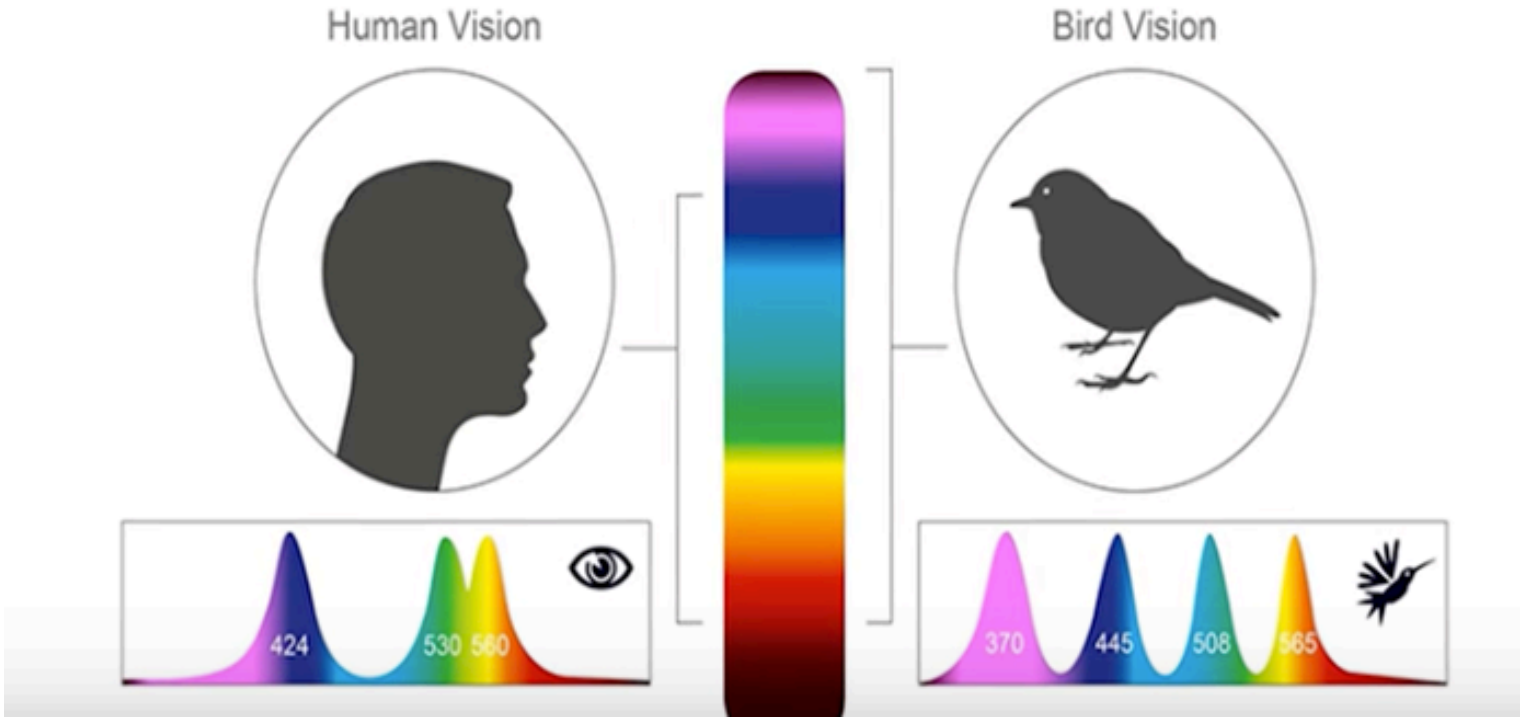


In bright light

Human vision

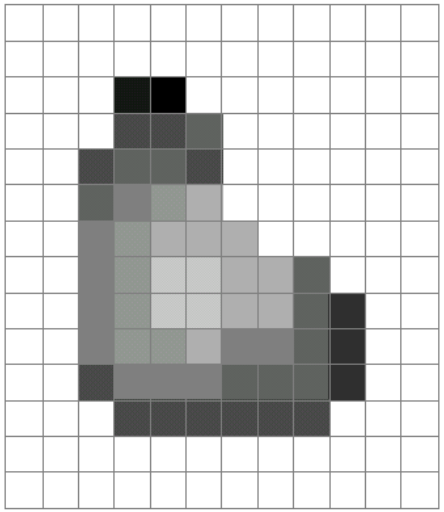


Human vision



What is an image?

- A grid of intensity values



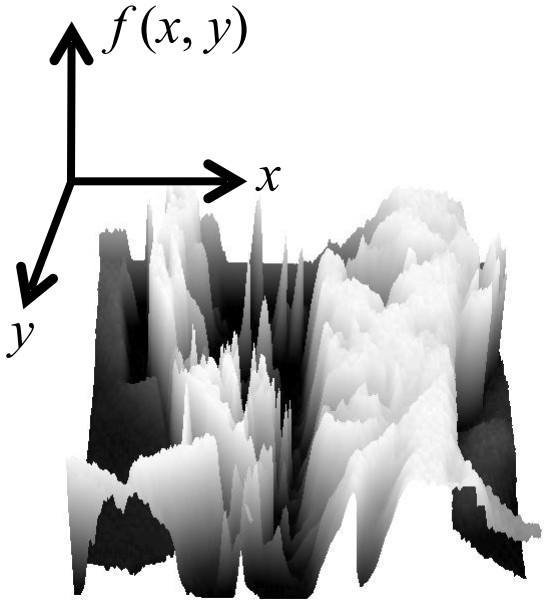
=

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)

What is a (grayscale) image?

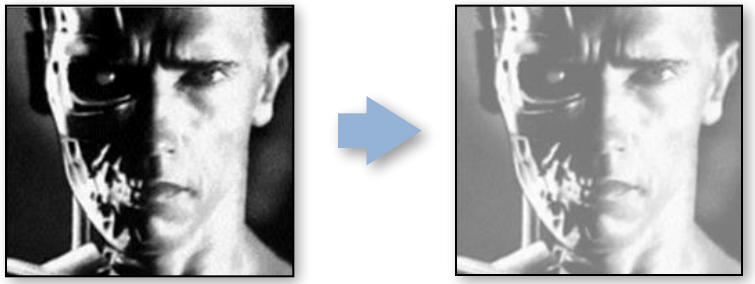
- A **2D function** $f: \mathbb{R}^2 \to \mathbb{R}$
  - $f(x,y)$ gives the **grayscale intensity** at position (x,y)



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

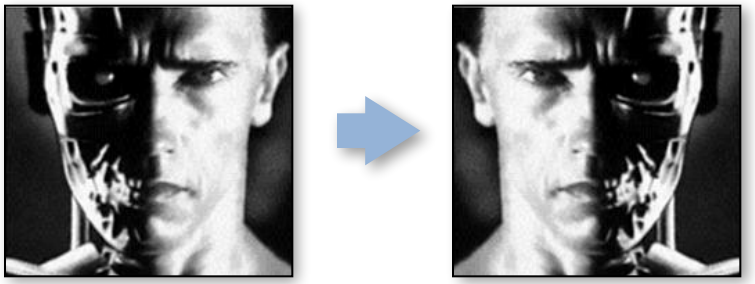
Image transformations

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



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

Becomes brighter than the original



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

Create mirror reflection of the original

Practice Time, Your Turn

Exercise#1
Practice Time, Your Turn

Group activity#1: Modifying the pixel values within a fixed rectangle

Now, let's tweak the code a bit to create a black rectangle (instead of a square) at any location within the image.

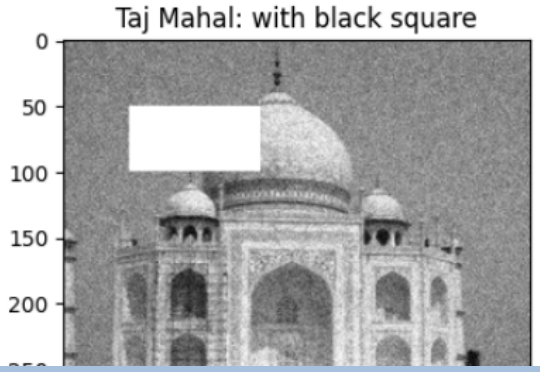
Choose any size you like for your *rectangle* by defining the **width** and **height** variables.

[ ]

```
# Your code here
# ...
# ...
# ...
```

34]

```
Grayscale image size: (361, 355)
image size: (361, 355)
```



Red

Green

Color image

Blue

Exercise#2
Practice Time, Your Turn

Group activity #2: Change the color of the bush and tree regions to green

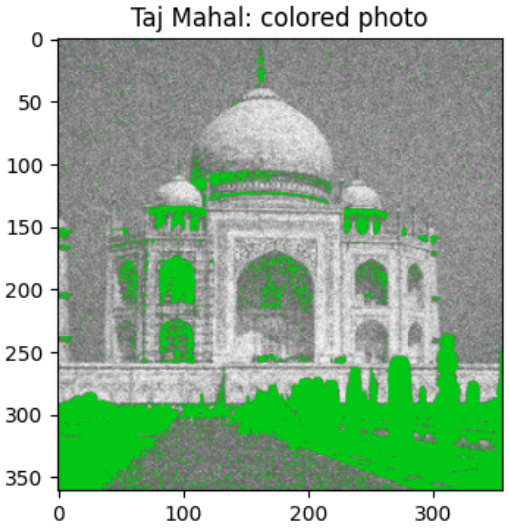
Start by making a new image with the same size as the **'Taj Mahal'** image.

Then, change the bushes and tree regions in the picture to green pixels.

[31]

```
# Your code here
# ...
# ...
# ...
```

```
Grayscale image size: (361, 355)
Text(0.5, 1.0, 'Taj Mahal: colored photo')
```



Exercise#3
Practice Time, Your Turn

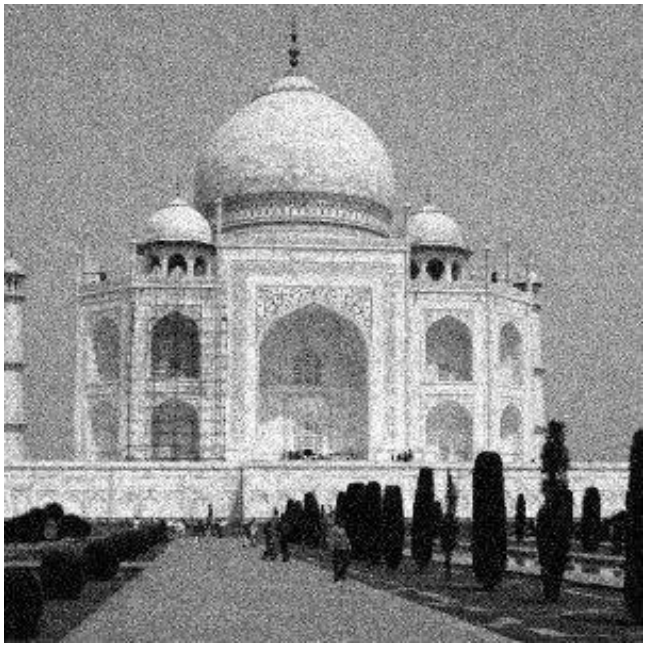
Group activity #3: Draw any other shapes (like circles, ellipses, etc.) at random locations.

> Reference: PIL.ImageDraw different shapes

```
# Your code here
# ...
# ...
```

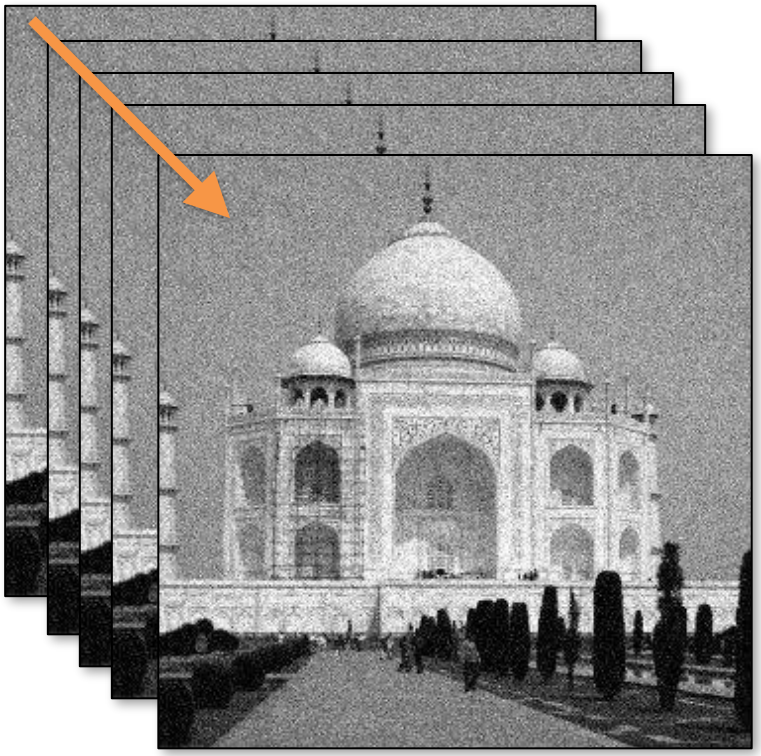
Image noise

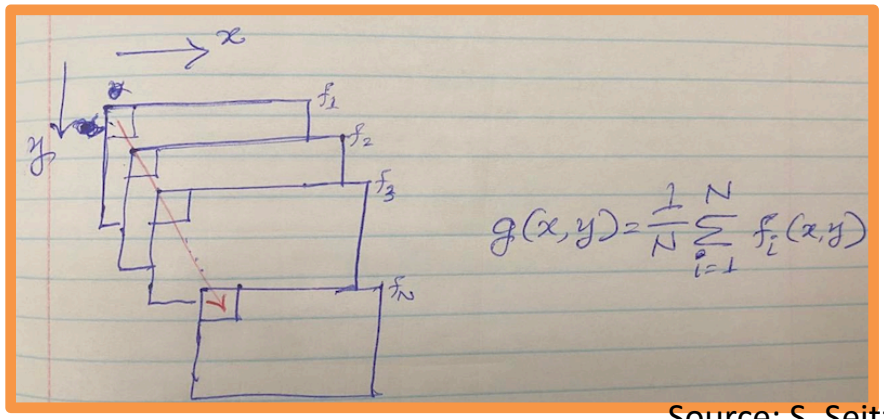
- Imaging systems (sensor, lens, compression algorithm, etc.) add unwanted noise



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?

Source: S. Seitz

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

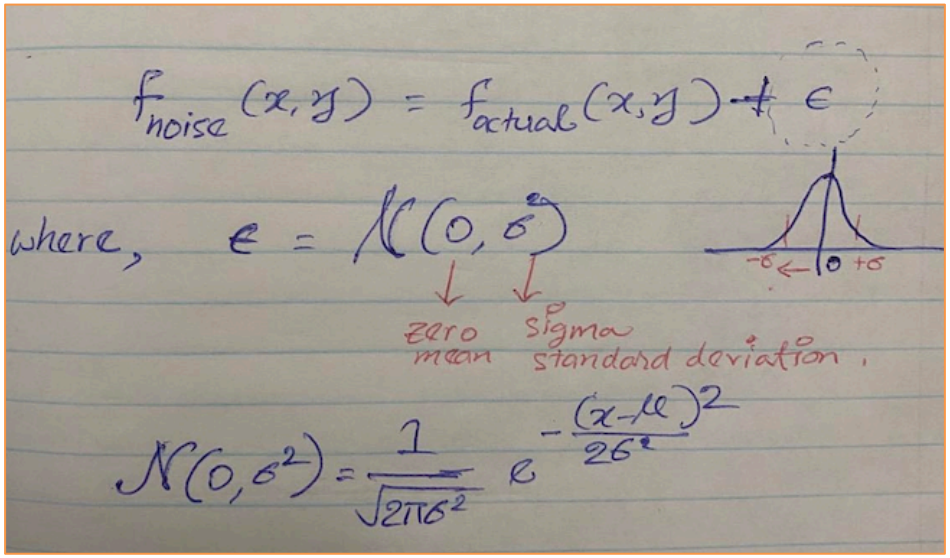


Image filtering

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

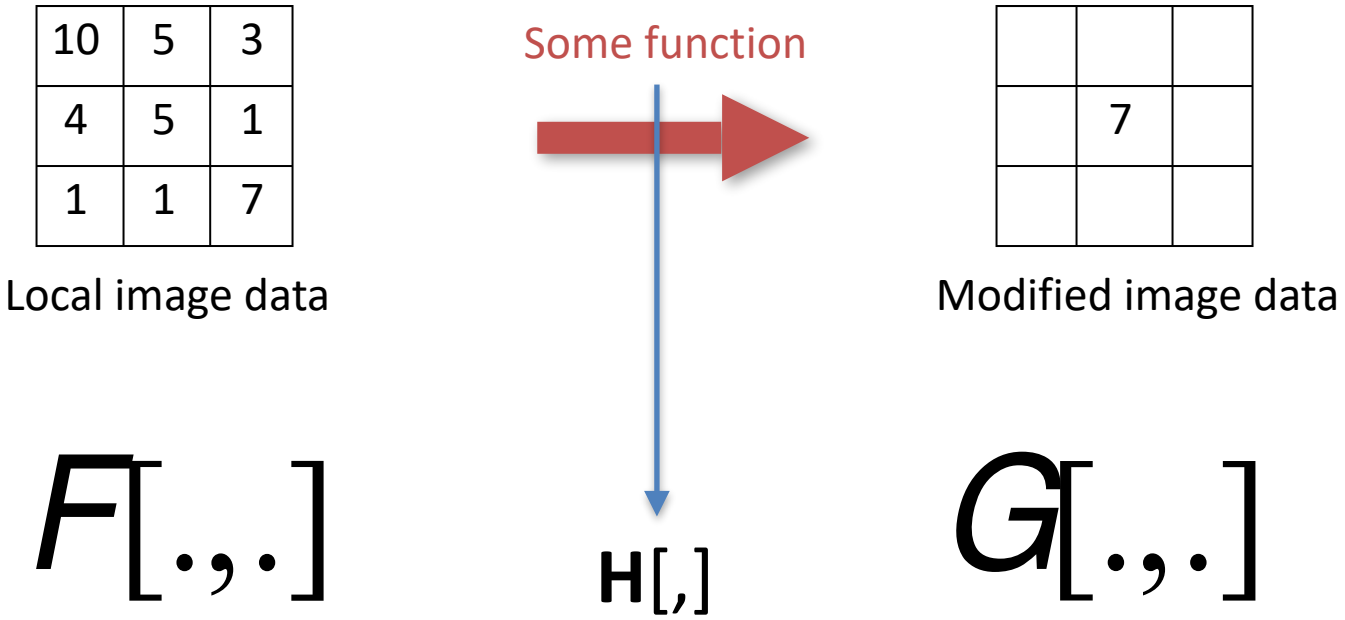
Local image data

Modified image data

Source: L. Zhang

Image filtering

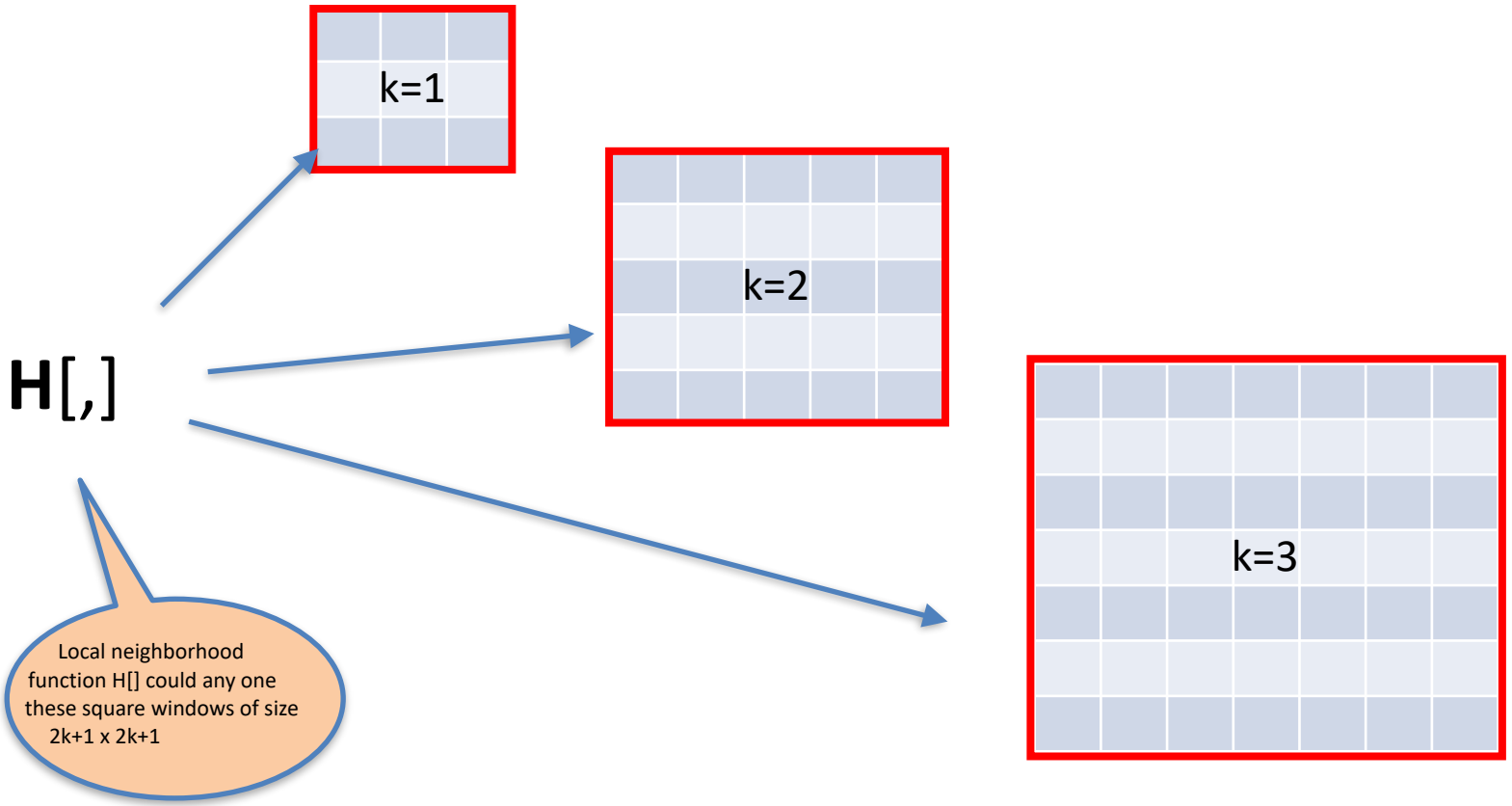
- Modify the pixels in an image based on some function of a local neighborhood of each pixel



Source: L. Zhang

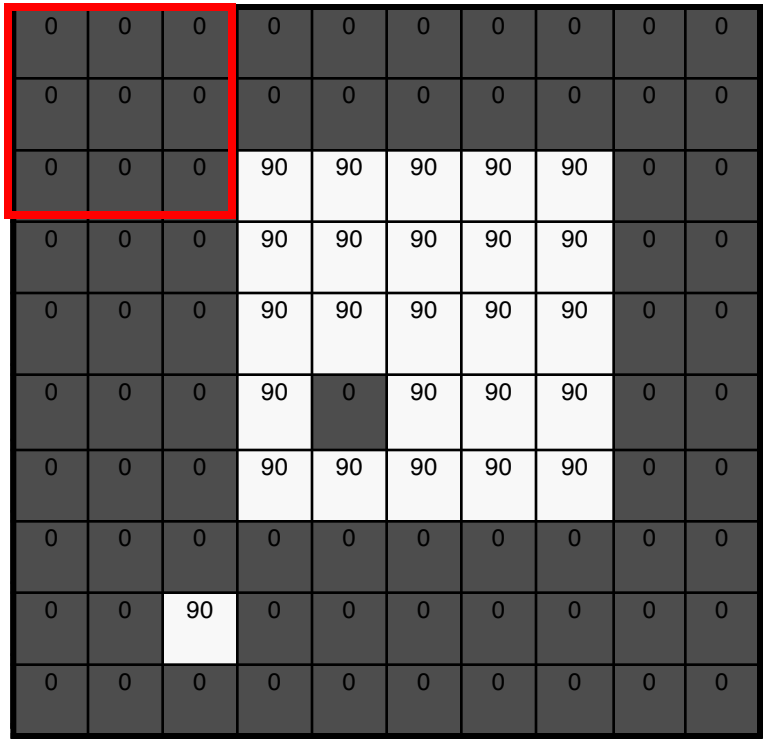
Image filtering

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

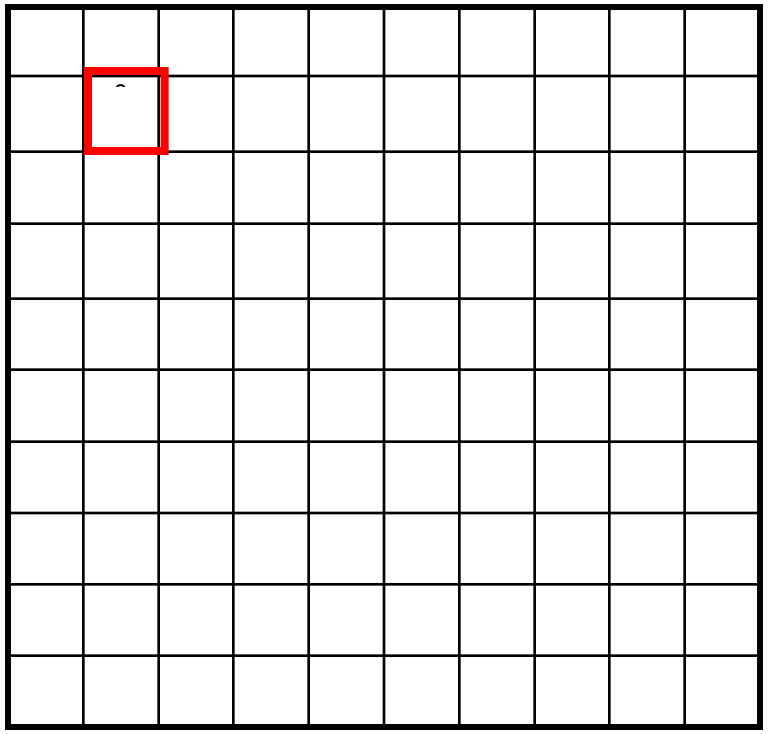


Mean filtering

$F[.,.]$



$G[.,.]$



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

Credit: S. Seitz

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

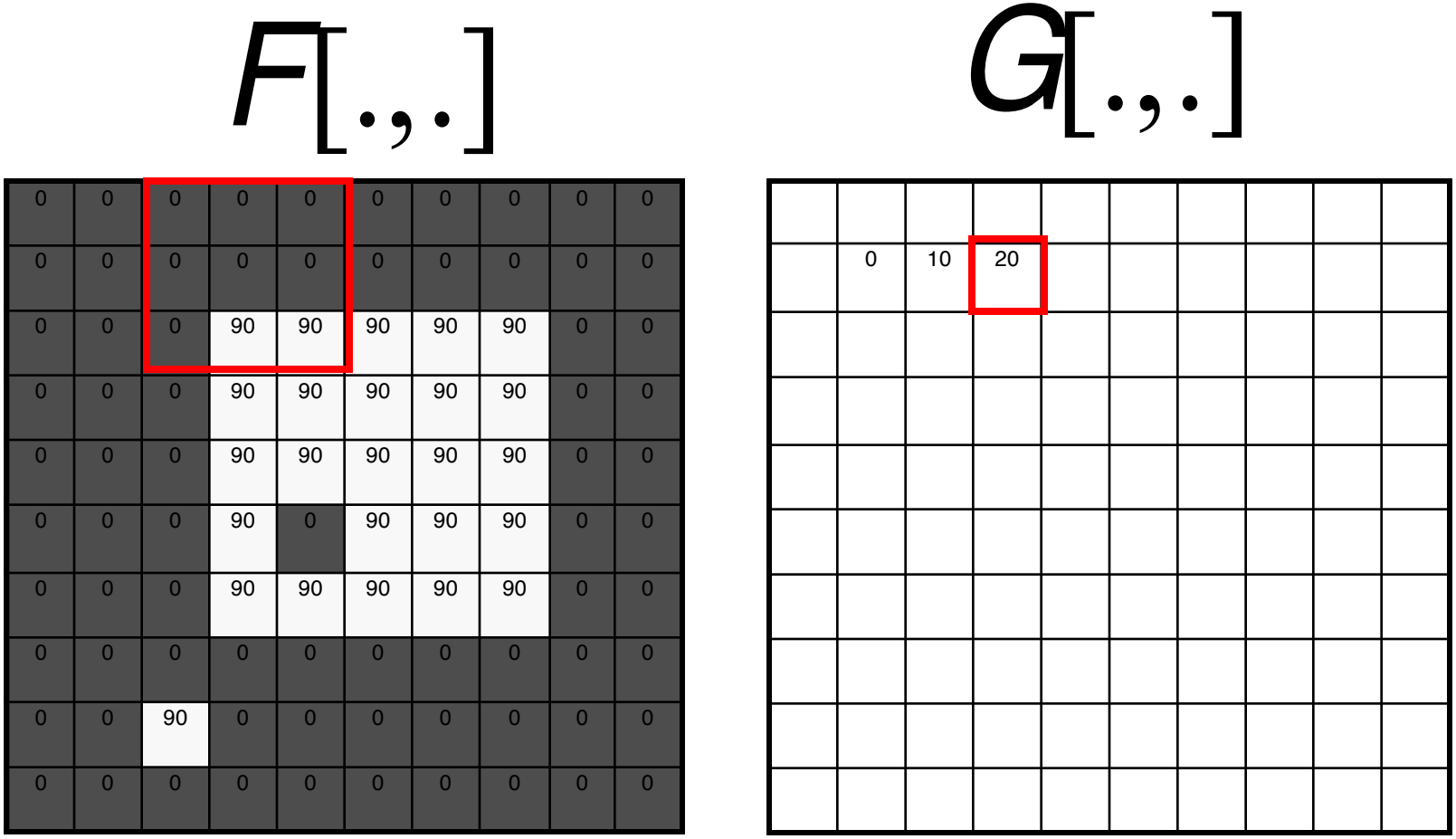
$G[.,.]$

	0	10							

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

Credit: S. Seitz

Mean filtering



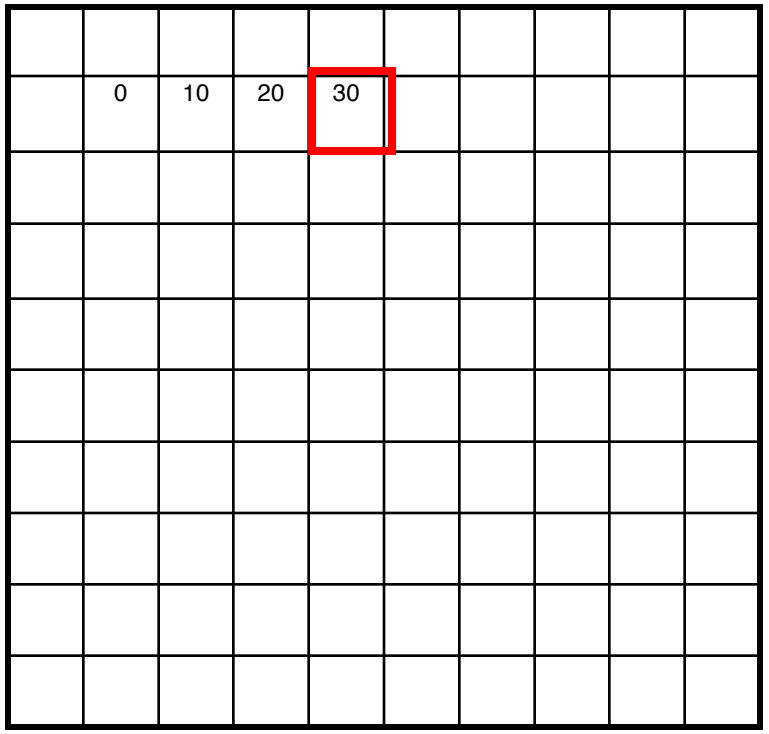
$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u+i, v+j)$$

Credit: S. Seitz

Mean filtering

$F[.,.]$

$G[.,.]$



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

Credit: S. Seitz

Mean filtering

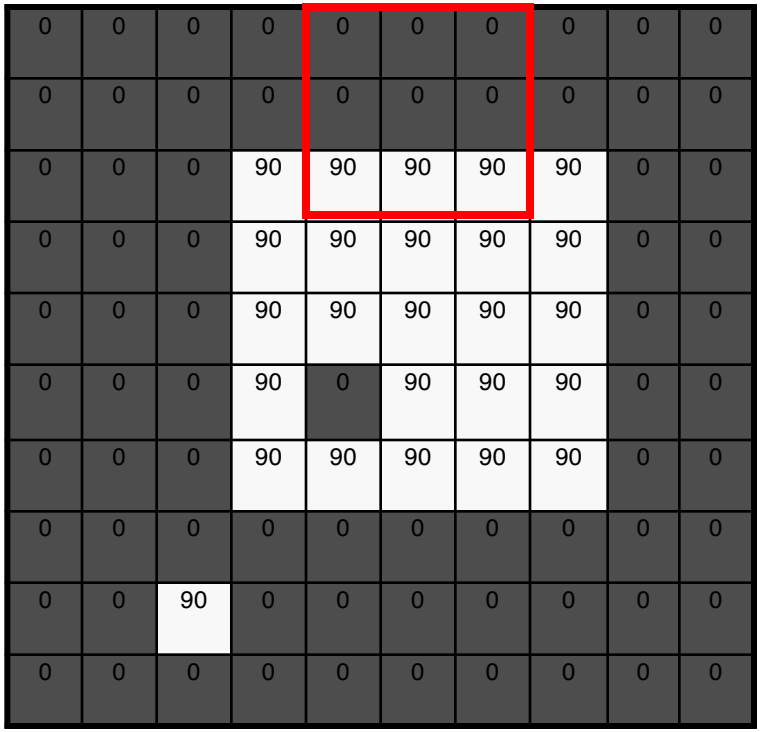
H

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

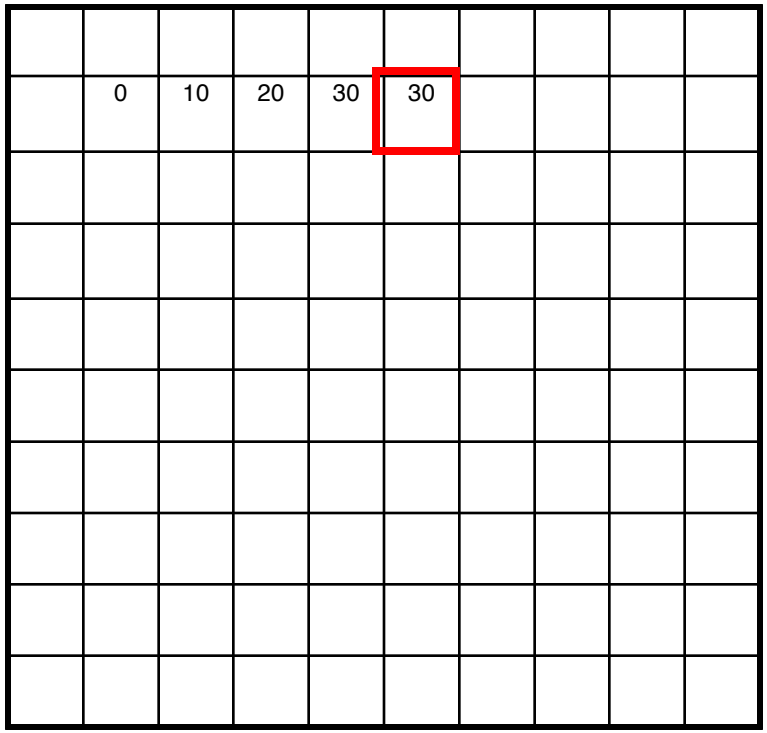
F

0	0	0
0	0	0
90	90	90

$F[.,.]$



$G[.,.]$

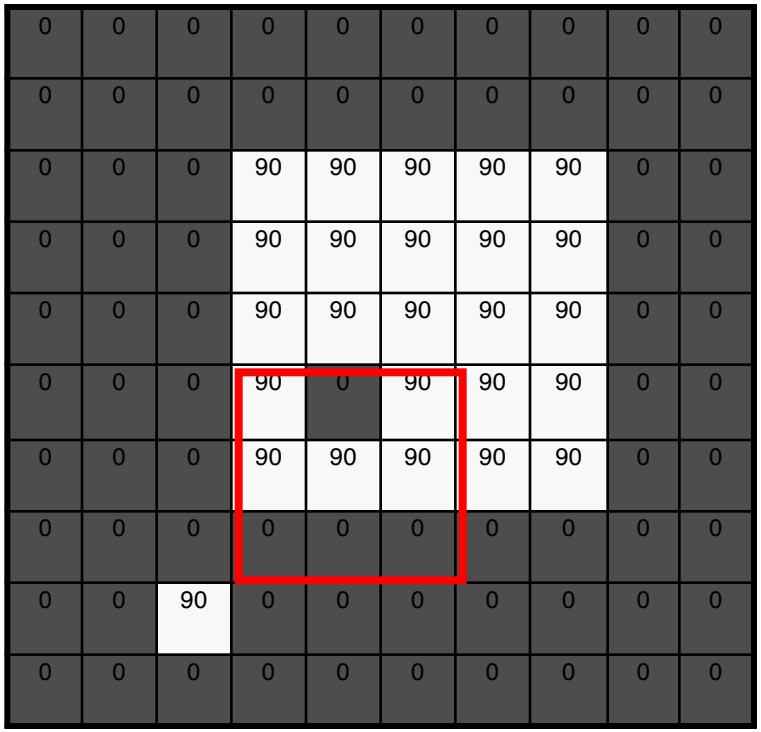


$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u+i, v+j)$$

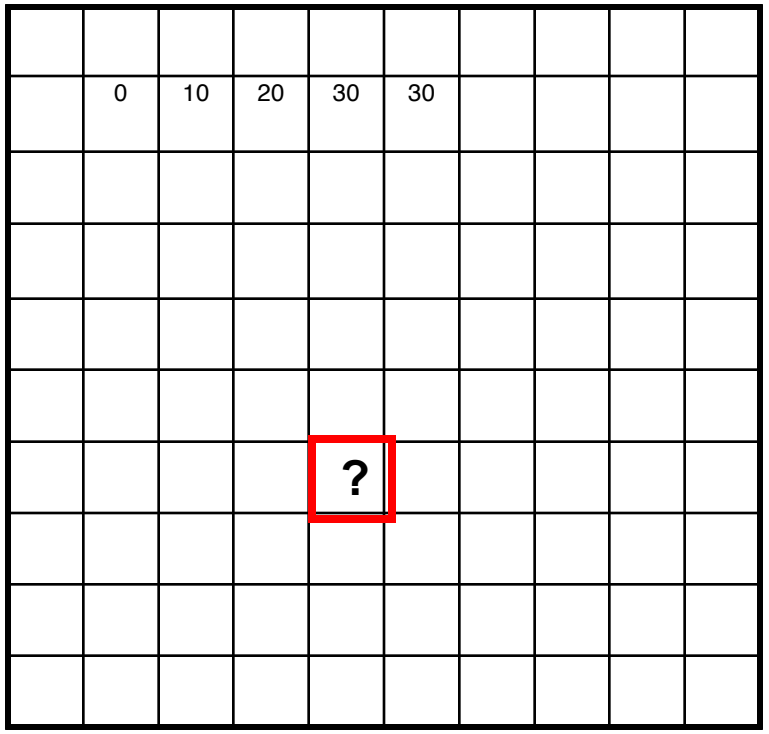
Credit: S. Seitz

Mean filtering

$F[.,.]$



$G[.,.]$



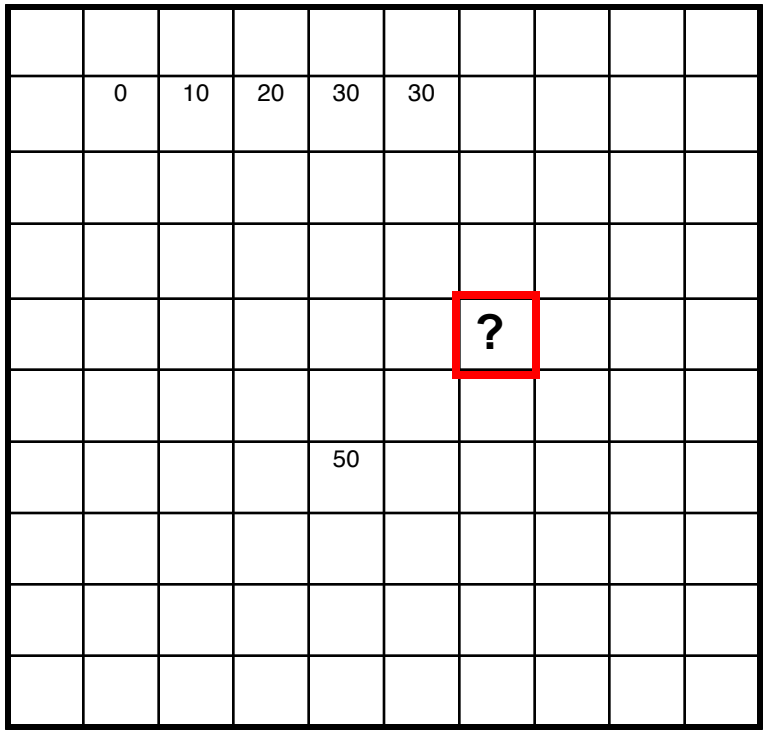
$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u+i, v+j)$$

Credit: S. Seitz

Mean filtering

$F[.,.]$

$G[.,.]$

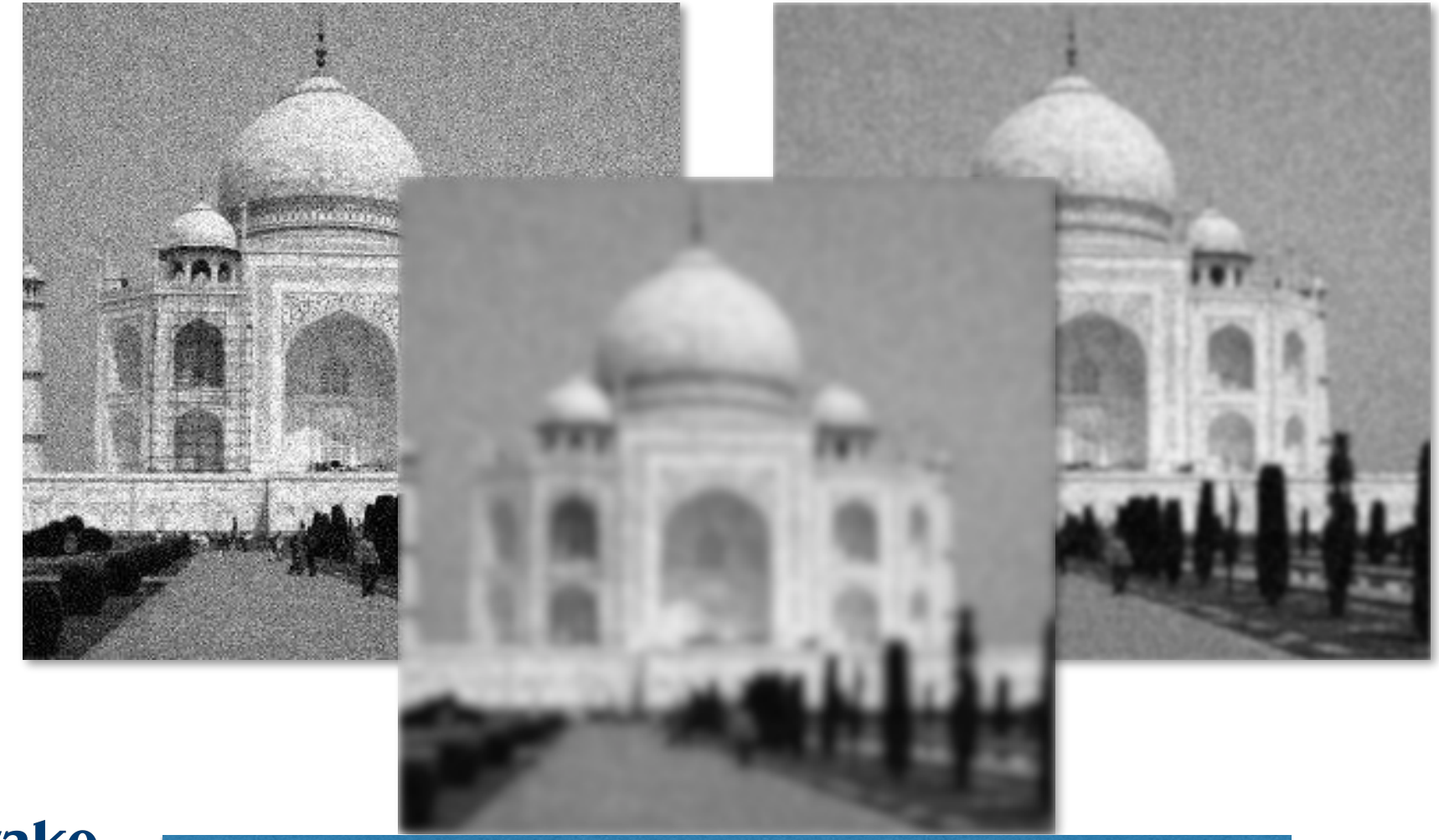


$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u+i, v+j)$$

Credit: S. Seitz

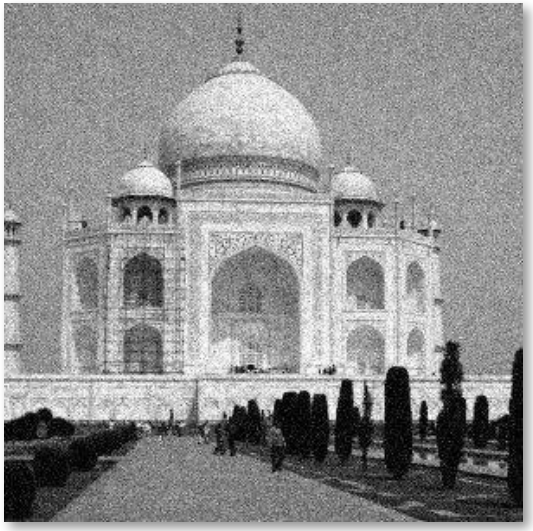
Mean filtering

- Larger k => more blurring



Mean filtering

- Larger k => 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 x 2k+1), then the cross-correlation of F with H is:

$$G = H \otimes F$$

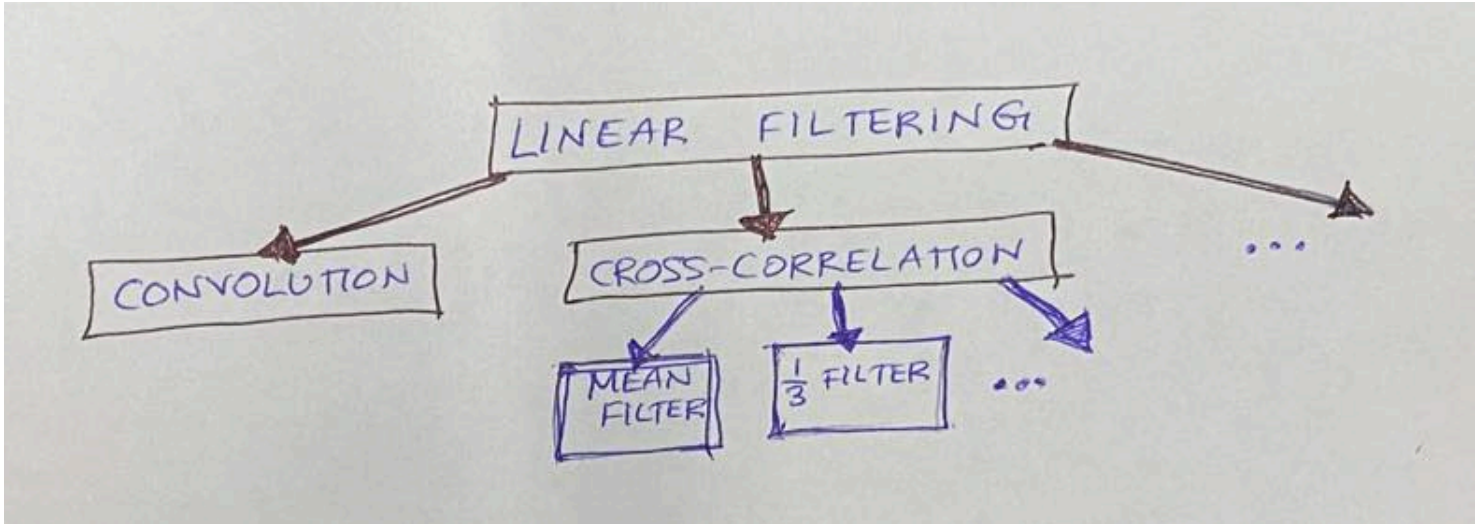
H			F		
0	0	0	0	0	0
0	0	0	0	0	0
1/9	1/9	1/9	90	90	90

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

0+0+0+0+0+0+(1/9*90)+(1/9*90)+(1/9*90) = 30

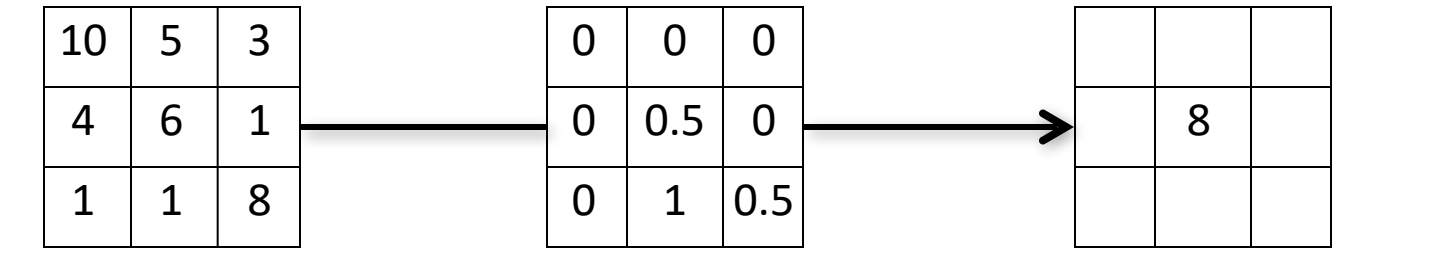
Linear filtering

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



Linear filtering

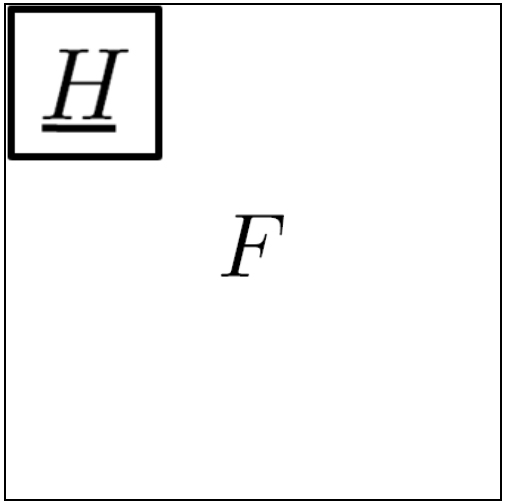
- More general version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a *linear combination* of its neighbors
- The prescription for the linear combination is called the "kernel" (or "mask", "filter")



$0.5*6 + 1*1 + 0.5*8 = 3 + 1 + 4 = 8$

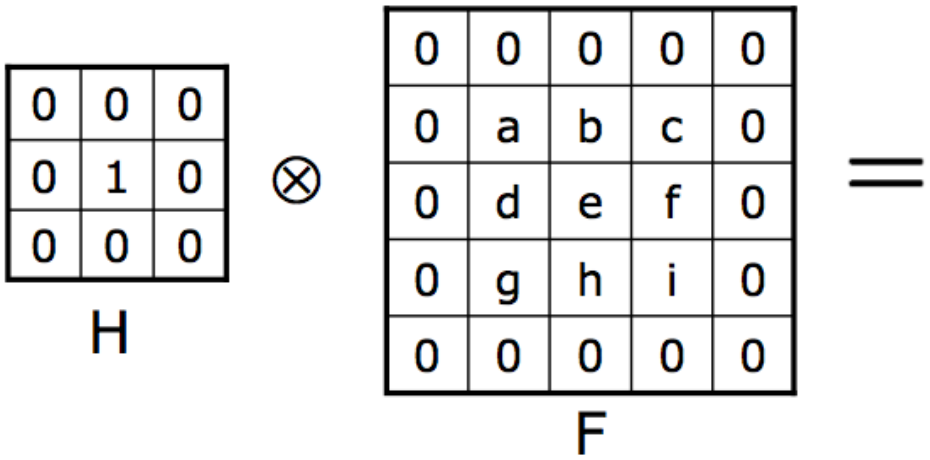
Source: L. Zhang

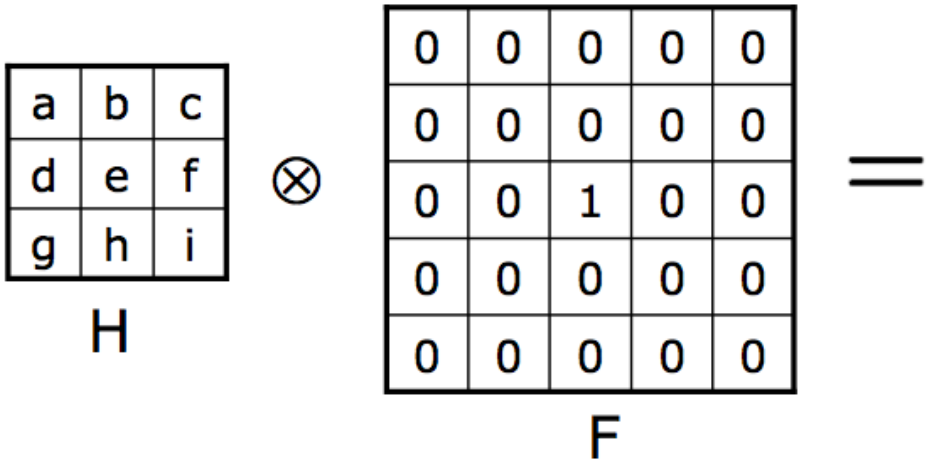
Cross correlation



Adapted from F. Durand

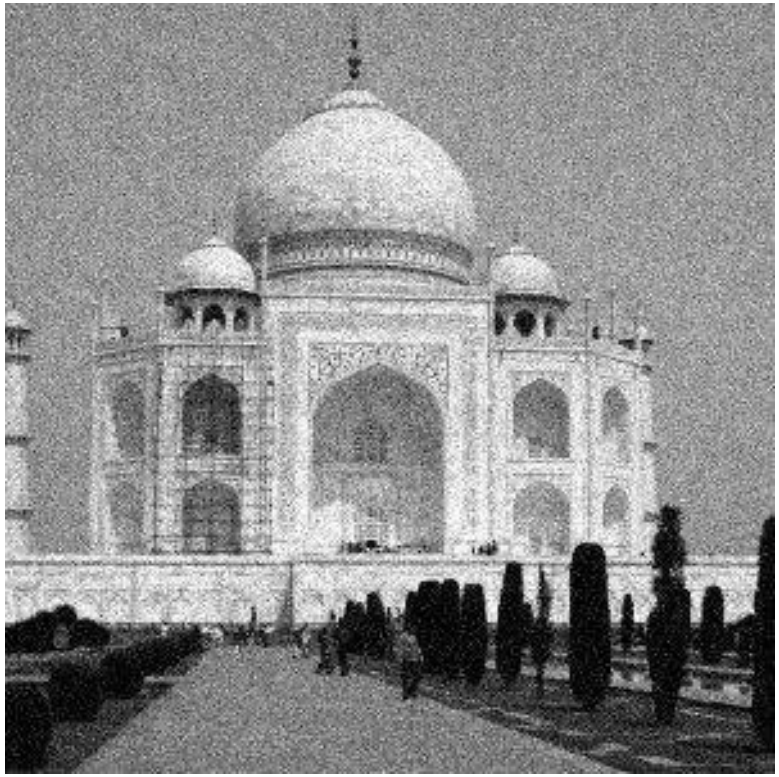
Cross-correlation examples





Coding activity: cross-correlation

- Reduce noise using cross-correlation from the given still scene ‘Tajmahal’ and others.



Exercise#4
Practice Time, Your Turn

Group activity#4: Cross-rorrelation computation

Great job learning how to manipulate images using various PIL operations! Now, let's take it a step further by completing the cross-correlation code. You may only need to add up to three lines of code. Once you've completed it, try reducing the noise in the given still scene of the Taj Mahal using your cross-correlation code. You're almost there!

Recall, that `img_pil.getpixel()`:

> takes as arguments a pixel location specified by `(x, y)`

and `img_pil.putpixel()`:

> takes two arguments: the first is a pixel location specified by `(x, y)`, and the second is the new intensity value specified by an integer value `new_intensity`.

Coding activity: cross-correlation

```
  print('')
'''

# compute 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)
    old_pixel_value = F.getpixel((x, y)) # we don't need it anymore

    new_pixel_value = 0.0
    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
        cur_pixel_value  = F.getpixel((x + u_col, y + v_row))
        # MODIFICATION 1: calculate the updated value of pixel
        # your code ...

    # update the value at location (x,y) with newly computed pixel value
    # MODIFICATION 1: change the pixel value in the 'new_img' with the calculate new value
    # your code ...
```

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 x 2k+1), then the cross-correlation of F with H is:

$$G = H \otimes F$$

H			F		
1/9	1/9	1/9	0	0	0
0	0	0	0	0	0
0	0	0	90	90	90

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

0+0+0+0+0+0+(1/9*90)+(1/9*90)+(1/9*90) = 30