Lecture 2
Operations on Images

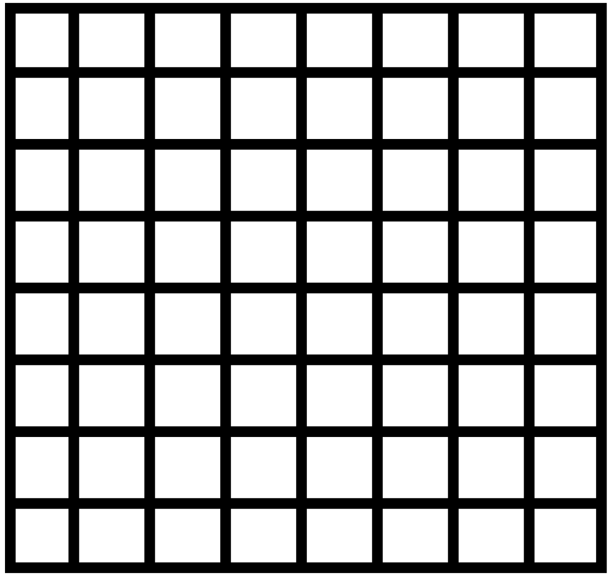
ECE 1360/2360

Learning Objectives:

- Mathematical operations
- Bitwise/Logical operations
- Dealing with overflow
- Masking and thresholding

An image is a matrix of information (tensor for color images)

X



Math based operations

$$Y = a * X$$

$$Y = fn(X)$$

$$Y = H * X * H'$$

Morphological operations

E.g. Dilate/Erode

```
For (i=0; i<n; i++)
        do something
end
```

Brightness

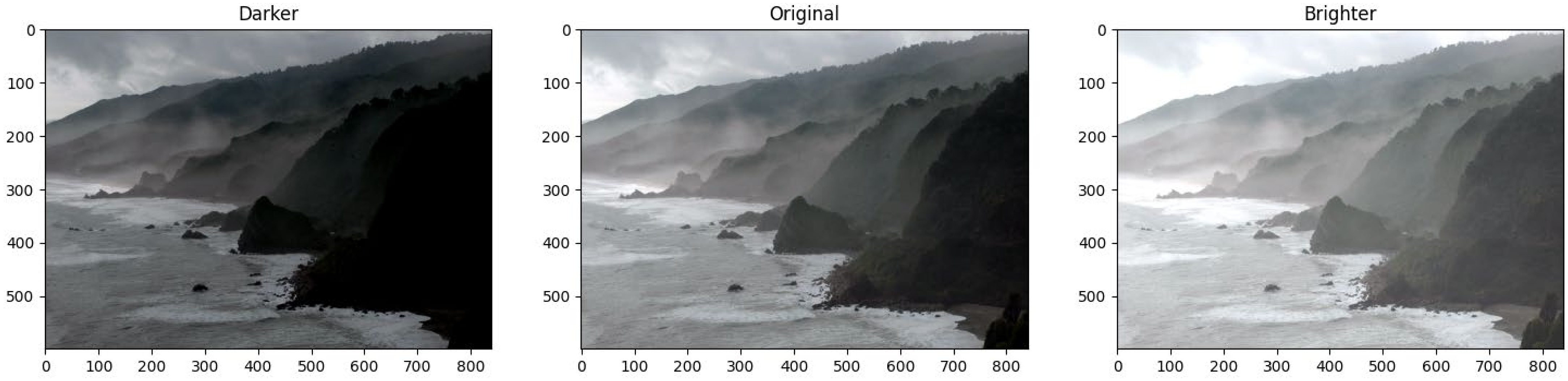
$$Y = X + a$$

Changing the brightness of an image means adding/subtracting a scalar from the image

From Lecture1/04_Image_Enhancement.ipnb

```
matrix = np.ones(img_rgb.shape, dtype="uint8") * 50
img_rgb_brighter = cv2.add(img_rgb, matrix)
img_rgb_darker = cv2.subtract(img_rgb, matrix)

# Show the images
plt.figure(figsize=[18, 5])
plt.subplot(131); plt.imshow(img_rgb_darker); plt.title("Darker");
plt.subplot(132); plt.imshow(img_rgb); plt.title("Original");
plt.subplot(133); plt.imshow(img_rgb_brighter);plt.title("Brighter");
```



Contrast

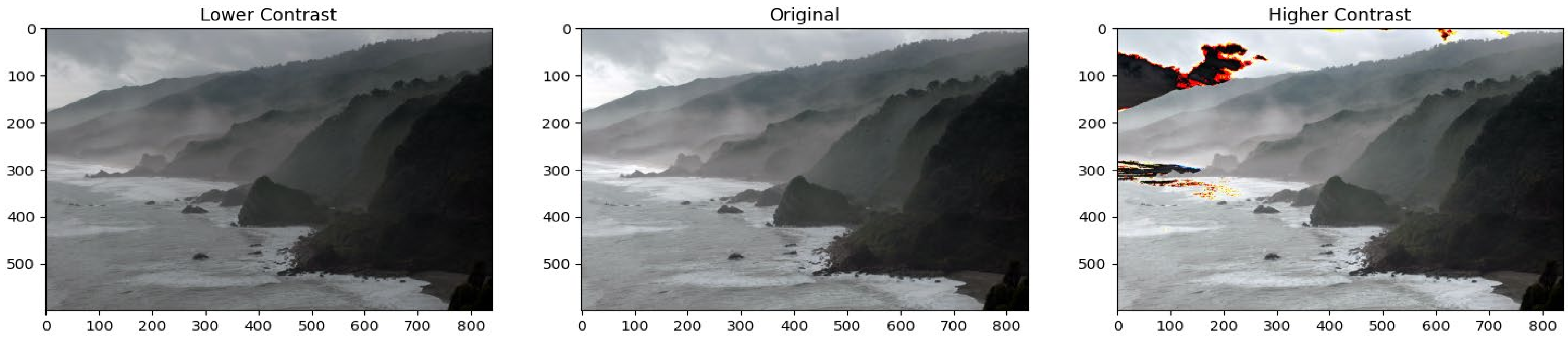
$$Y = a * X$$

Contrast is the range of values in the image. So multiplying by a scalar changes the contrast

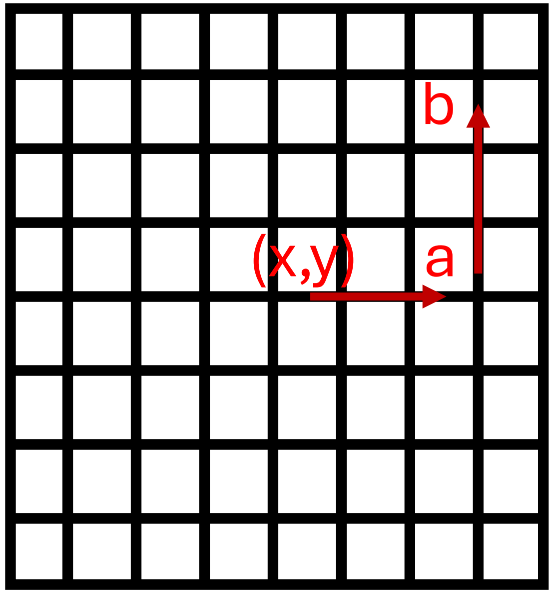
From Lecture1/04_Image_Enhancement.ipnb

```
matrix_low_contrast = np.ones(img_rgb.shape) * 0.8
matrix_high_contast = np.ones(img_rgb.shape) * 1.2
img_rgb_darker = np.uint8(cv2.multiply(np.float64(img_rgb), matrix_low_contrast))
img_rgb_brighter = np.uint8(cv2.multiply(np.float64(img_rgb), matrix_high_contast))

# Show the images
plt.figure(figsize=[18,5])
plt.subplot(131); plt.imshow(img_rgb_darker); plt.title("Lower Contrast");
plt.subplot(132); plt.imshow(img_rgb); plt.title("Original");
plt.subplot(133); plt.imshow(img_rgb_brighter);plt.title("Higher Contrast");
```



$$Y = H * X * H'$$



Point Spread Function

General

$$h(x, a, y, b)$$

Operation varies depending on which position (x,y) in the image

Shift invariant

$$h(a - x, b - y)$$

Operation is independent of the position (x,y) in the image

2D convolution as a matrix operation

$$\begin{matrix} a & b & c & d & e & f \\ g & h & \dots & & & \end{matrix}$$

< 6 x 6 >

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

< 3 x 3 >

$$\left.\boxed{\begin{matrix} 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 \end{matrix}}\right\} n = (6 - 3) + 1$$

$$\boxed{\begin{matrix} 4 & 5 & 6 & 0 & 0 & 0 \\ 0 & 4 & 5 & 6 & 0 & 0 \\ 0 & 0 & 4 & 5 & 6 & 0 \\ 0 & 0 & 0 & 5 & 6 & 7 \end{matrix}}$$

$$\boxed{\begin{matrix} 7 & 8 & 9 & 0 & 0 & 0 \\ 0 & 7 & 8 & 9 & 0 & 0 \\ 0 & 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 7 & 8 & 9 \end{matrix}}$$

2D convolution as a matrix operation

$$
\left.\begin{array}{cccccccccccccccccccccccccccccccccccc}
1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0&0&&&&&&&&&&&&&&&&&&\\
0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0&&&&&&&&&&&&&&&&&&\\
0&0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&&&&&&&&&&&&&&&&&&\\
0&0&0&1&2&3&0&0&0&5&6&7&0&0&0&7&8&9&&&&&&&&&&&&&&&&&&\\
&&&&&&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0&0&&&&&&&&&&&&\\
&&&&&&0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0&&&&&&&&&&&&\\
&&&&&&0&0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&&&&&&&&&&&&\\
&&&&&&0&0&0&1&2&3&0&0&0&5&6&7&0&0&0&7&8&9&&&&&&&&&&&&\\
&&&&&&&&&&&&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0&0&&&&&&\\
&&&&&&&&&&&&0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0&&&&&&\\
&&&&&&&&&&&&0&0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&&&&&&\\
&&&&&&&&&&&&0&0&0&1&2&3&0&0&0&5&6&7&0&0&0&7&8&9&&&&&&\\
&&&&&&&&&&&&&&&&&&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0&0\\
&&&&&&&&&&&&&&&&&&0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0&0\\
&&&&&&&&&&&&&&&&&&0&0&1&2&3&0&0&0&4&5&6&0&0&0&7&8&9&0\\
&&&&&&&&&&&&&&&&&&0&0&0&1&2&3&0&0&0&5&6&7&0&0&0&7&8&9
\end{array}\right. = H
$$

2D convolution as a matrix operation

$$\begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ \vdots \end{matrix} = \mathrm{g}$$

<16 x 36 >   <36 x 1 >

$$H * g' = x$$

$$reshape(x) \rightarrow < 4\ x\ 4 >$$

2D convolution as a matrix operation

In MATLAB: conv2(B,A,'valid')

N = (6 – 3) +1

If you want the same size output as input image:

N=6

= (8 – 3)

so need to pad the image to <8x8>

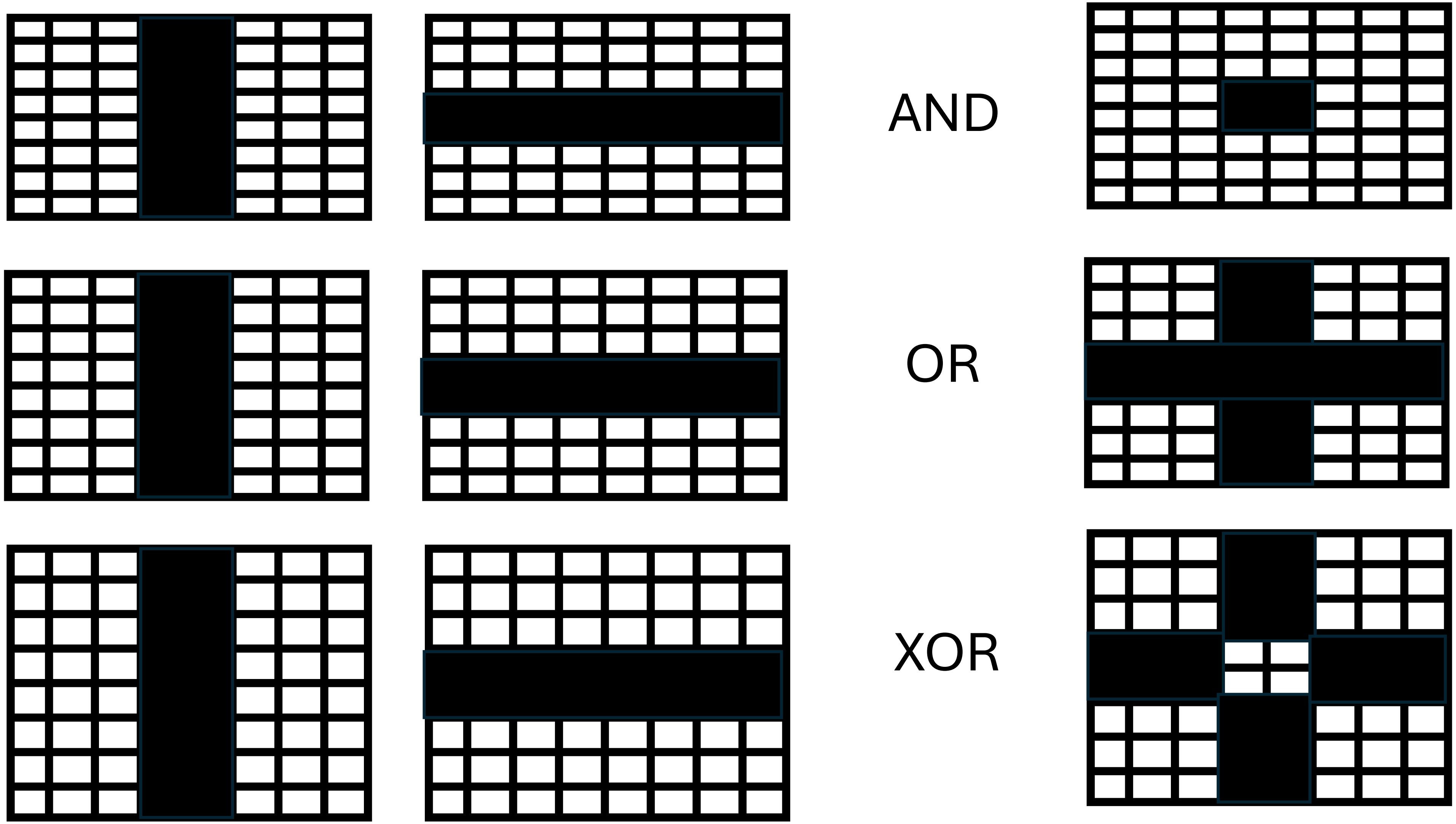
```
H = kron(eye(N),reshape(A’,1,[] )
g =  reshape(B’,1,[] )
x = reshape(H*g’ , N, N )
```

2D convolution as a matrix operation

```
kernel= np.ones((3,3))/9

img_smooth = cv2.filter2D(img,-1,kernel)
```

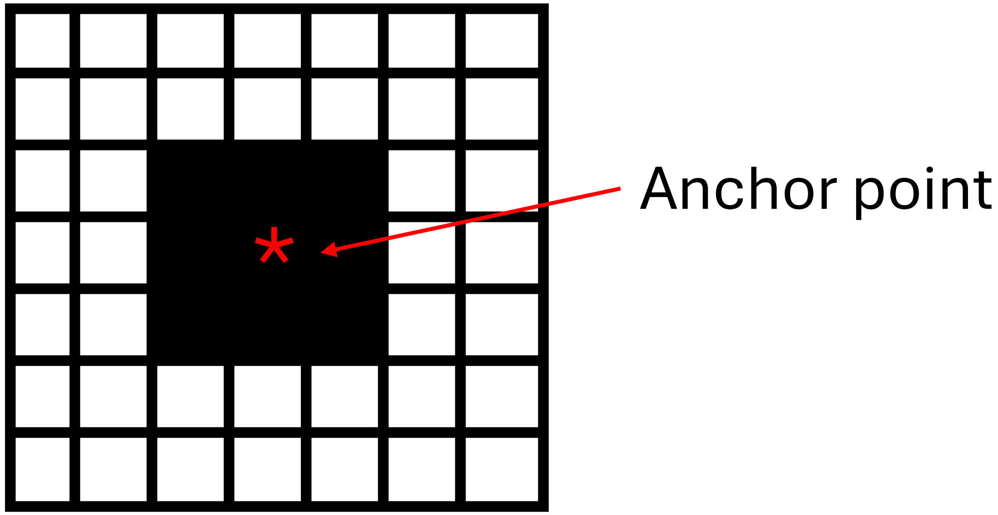
Boolean (Bit-wise) operations



Morphological Operations
Erosion/Dilation

Kernel is Boolean (mask)

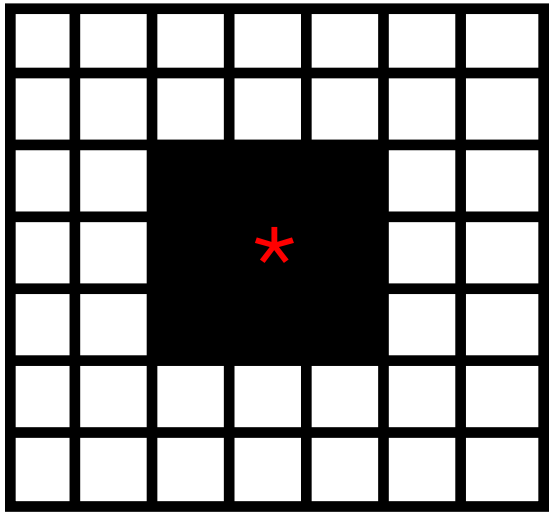
Structure/Kernel



Erosion/Dilation

Anchor point

Structure/Kernel

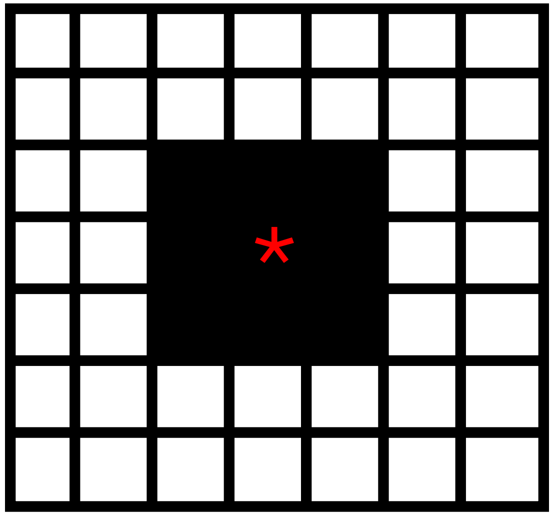


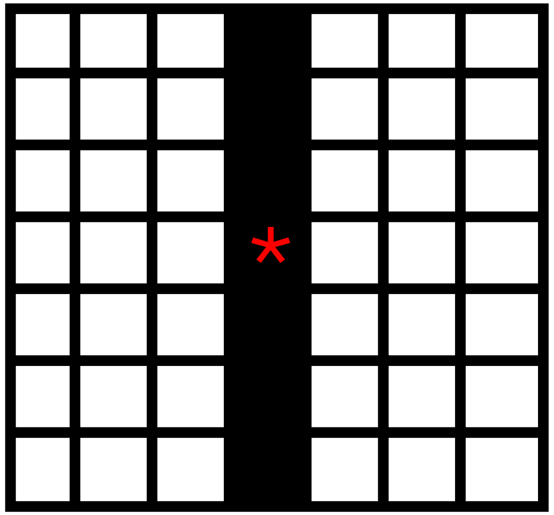
For {i,j} in image:
    move kernel anchor to {i,j}
    list = find (kernel ==true)
    newVal{i,j} = fcn (image[list])

Dilation - MAX (image[list])
Erosion - MIN (image[list])

Denoising - MEDIAN (image[list])
Smoothing - MEAN (image[list])

Erosion/Dilation

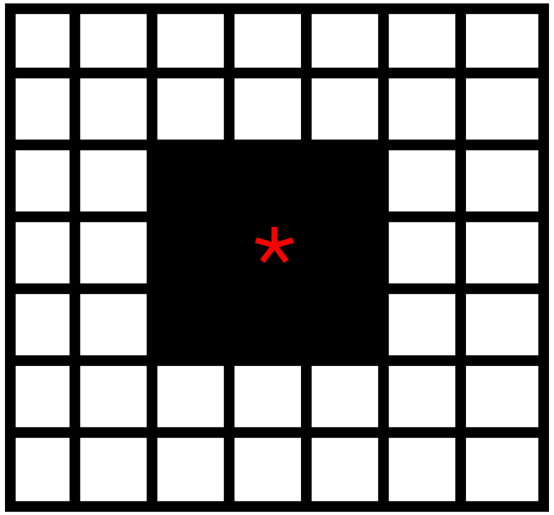




Convolutional (Filtering) Operations

Anchor point

Structure/Kernel



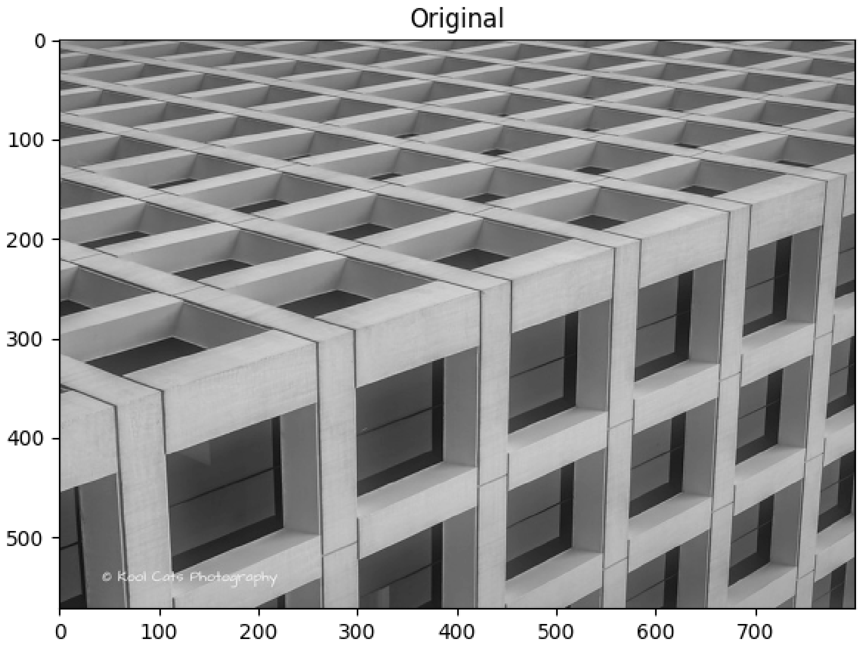
For {i,j} in image:

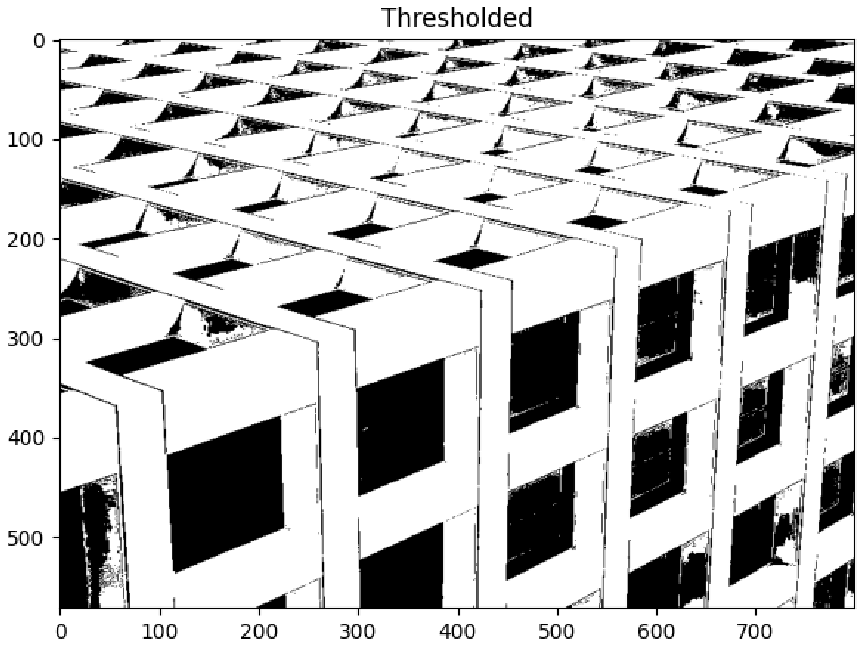
move kernel anchor to {i,j}

newVal{i,j} = sum(image .* kernel)

Thresholding

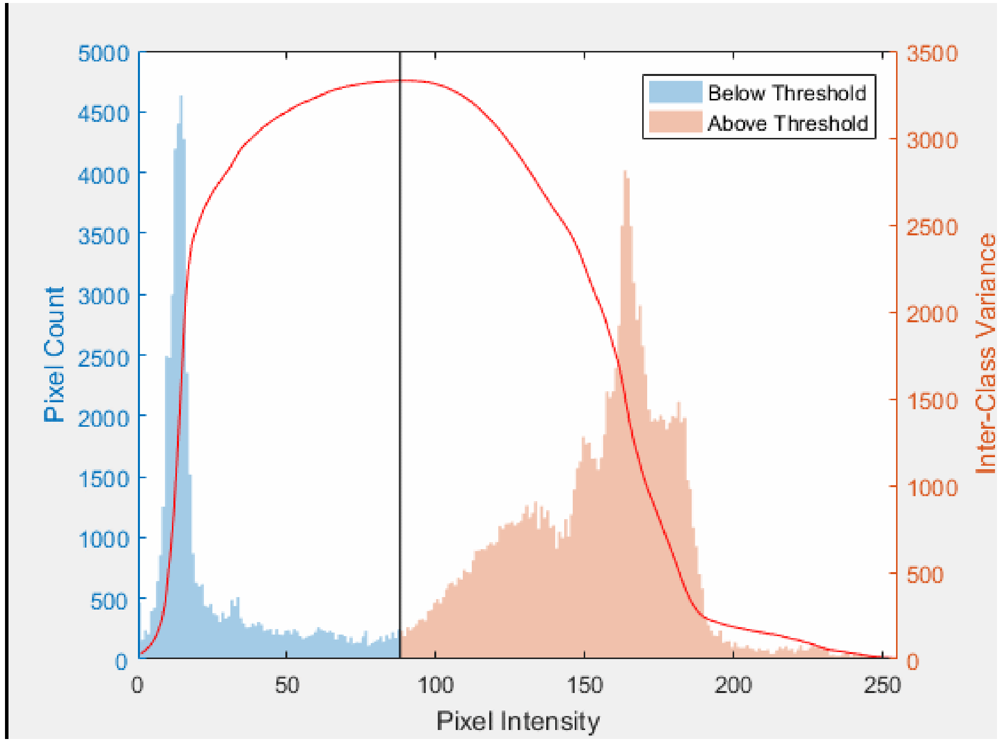
```
thres, img_out = cv2.threshold(img, thresh, replacement value, type)
```





Otsu’s thresholding

- Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Transactions on Systems, Man, and Cybernetics*. **9** (1): 62–66.



Variance values above thresh

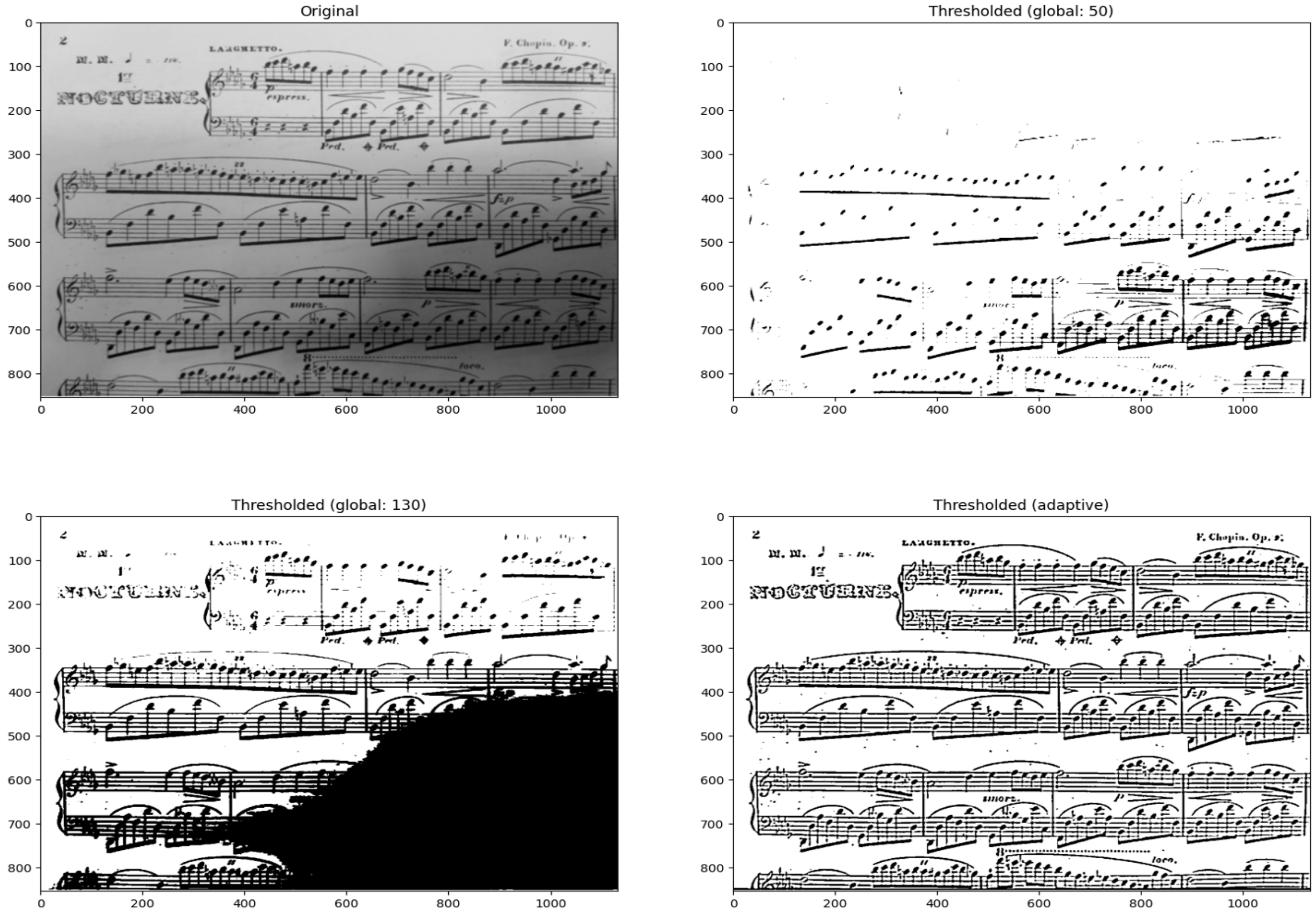
Variance values below thresh

$$\sigma^2 = w_0 \cdot {\sigma_0}^2 + w_1 \cdot {\sigma_1}^2$$

Probability of values below thresh

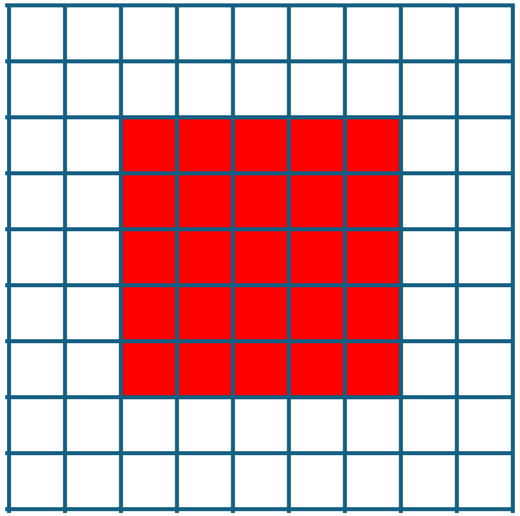
Probability of values above thresh

Adaptive thresholding



Adaptive thresholding

$$C = \left(\frac{1}{n * m} \sum_{i=1}^{n} \sum_{j=1}^{m} x[i,j] \right) > thres$$



$$C = \left(\frac{1}{\sum_{i=1}^{n} \sum_{j=1}^{m} G[i,j]} \sum_{i=1}^{n} \sum_{j=1}^{m} G[i,j] * x[i,j] \right) > thres$$