Introduction to Computer Vision in Python

Teodora Szasz

Image Analysis & Data Visualization Specialist

tszasz@uchicago.edu
Research Computing Center

What is Computer Vision?

Every picture tells a story

Every picture tells a story

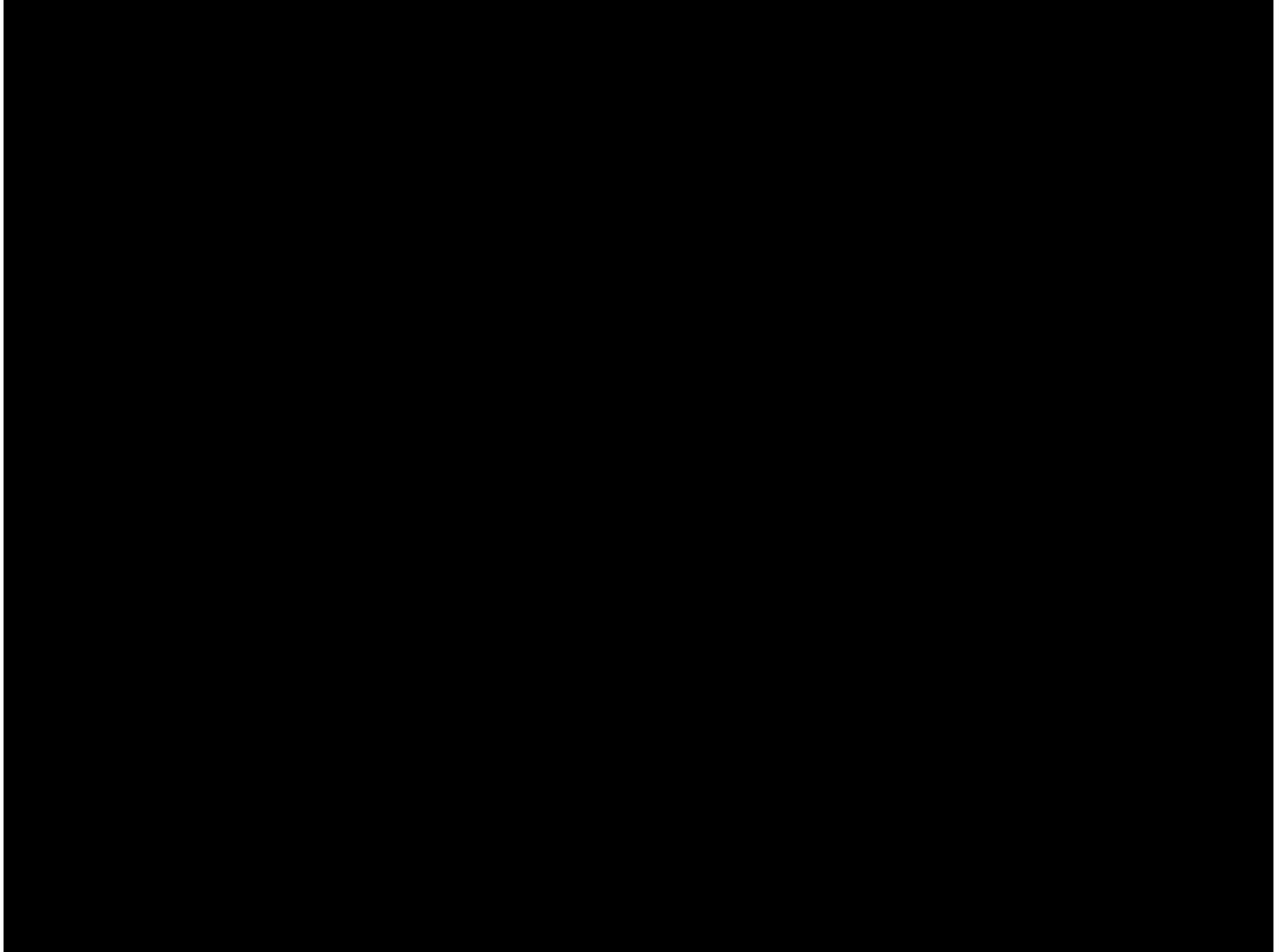
Goal of computer vision is to write computer programs that can interpret images

Why study Computer Vision?

- Images and movies are ubiquitous in both production and consumption
- Applications to manipulate images (movies) are a great need
- As are the systems to extract information from imagery
  - Surveillance
  - Building 3D models
  - Motion capture assisted

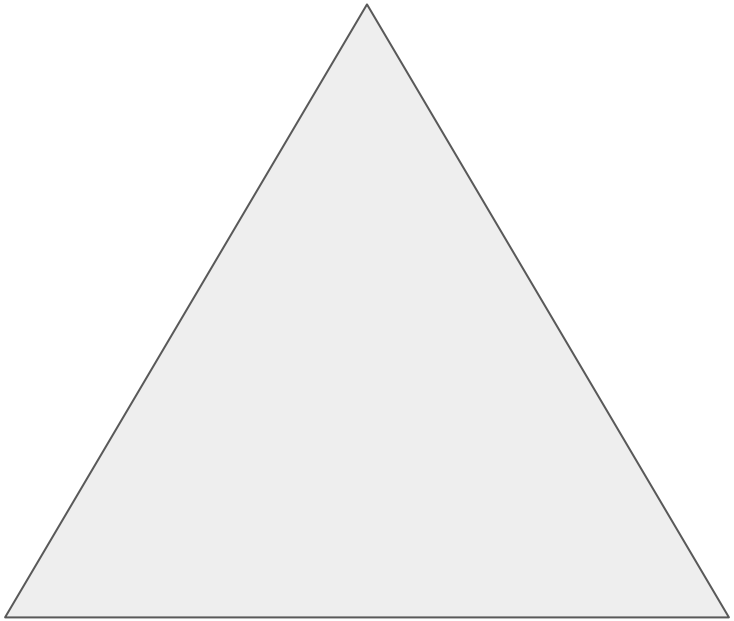
What is the state of the art in computer vision?


20
cans
Coca-Cola



Workshop outline

Computational Models (Math)



Algorithm

Real Images

- Introduction
- Filtering
- Template matching
- Edge detection
- Line detection - in a video

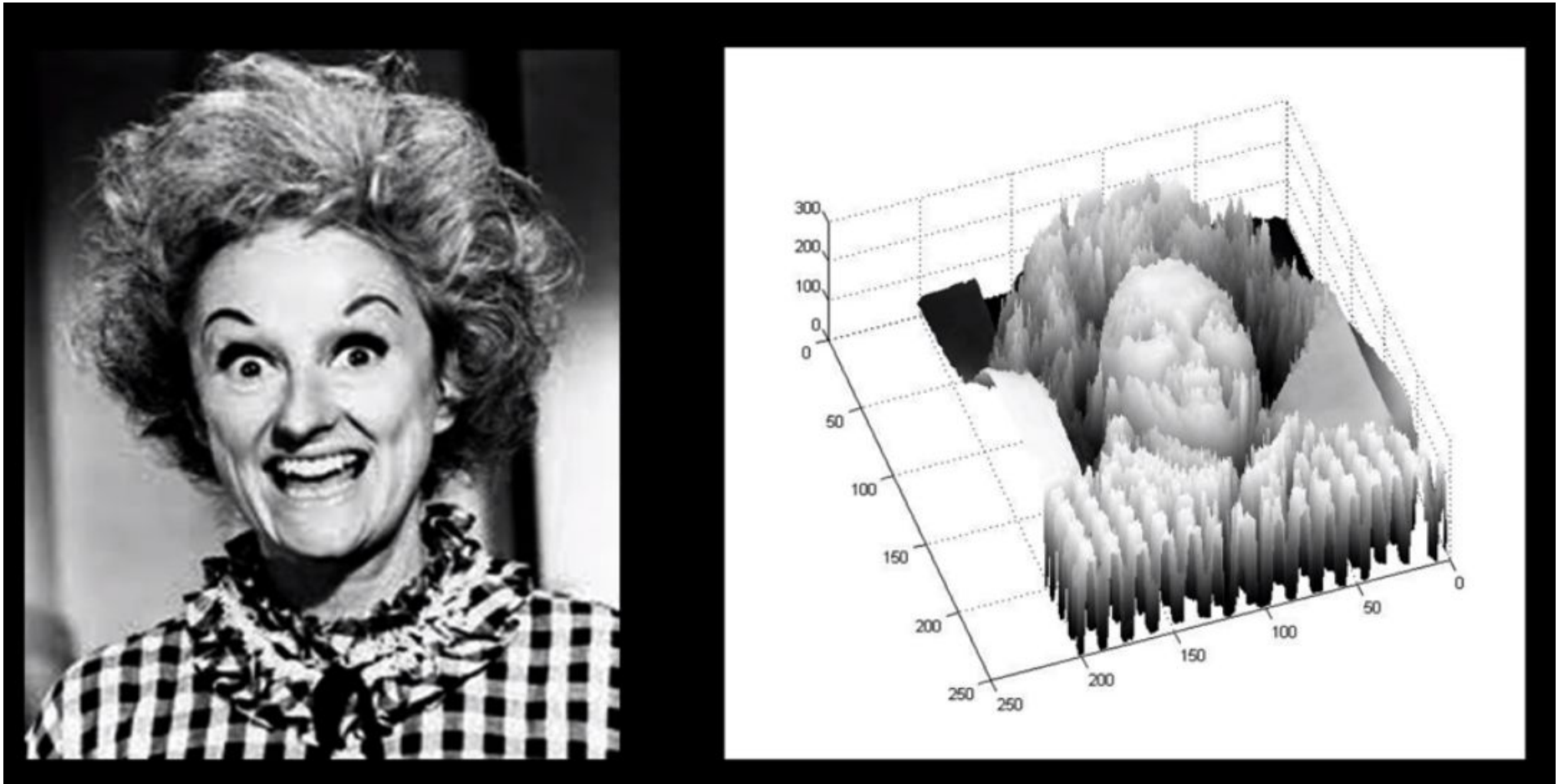
What is an image?

Images as functions

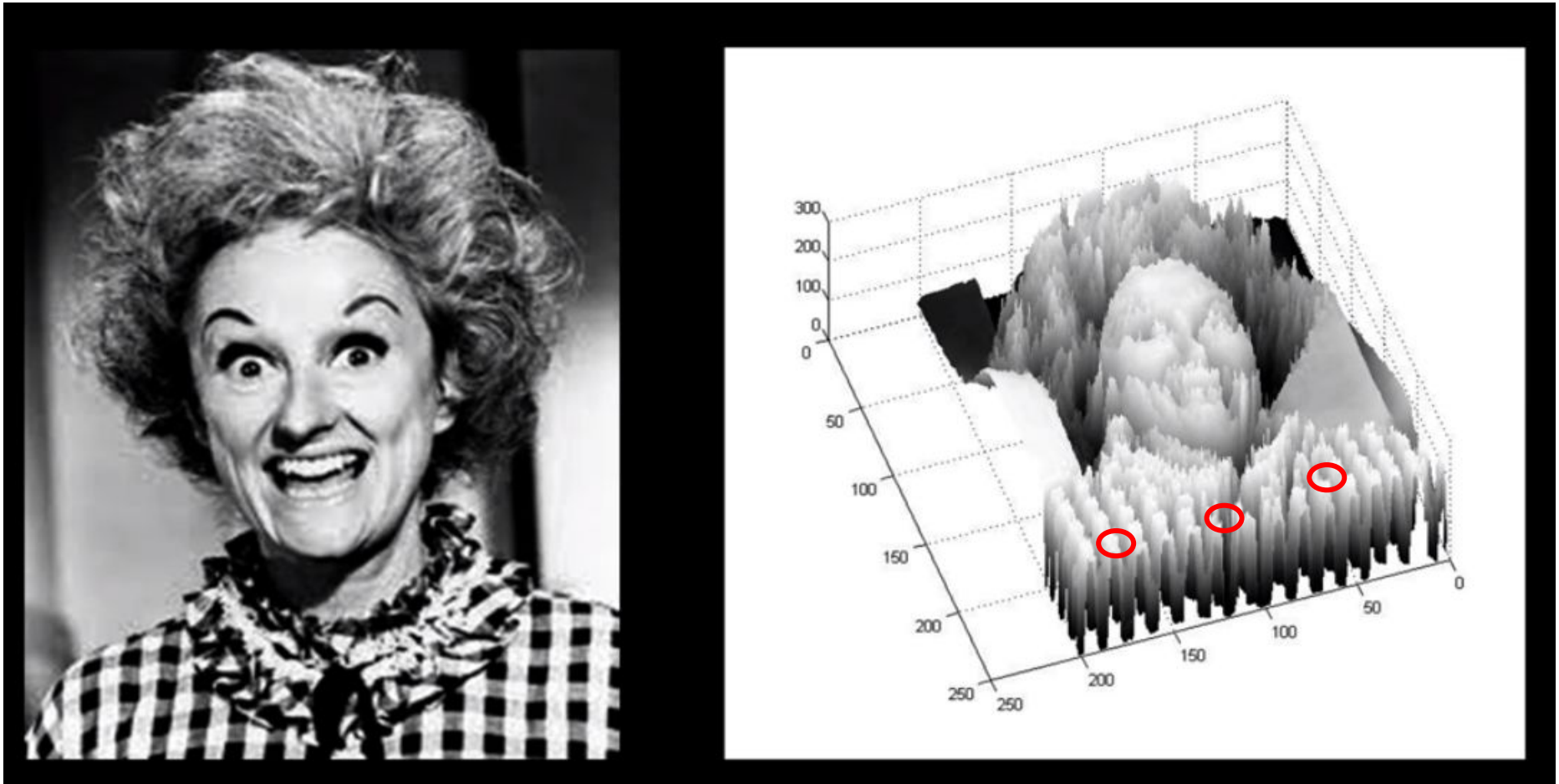
Images as functions

I(x,y)

Images as functions



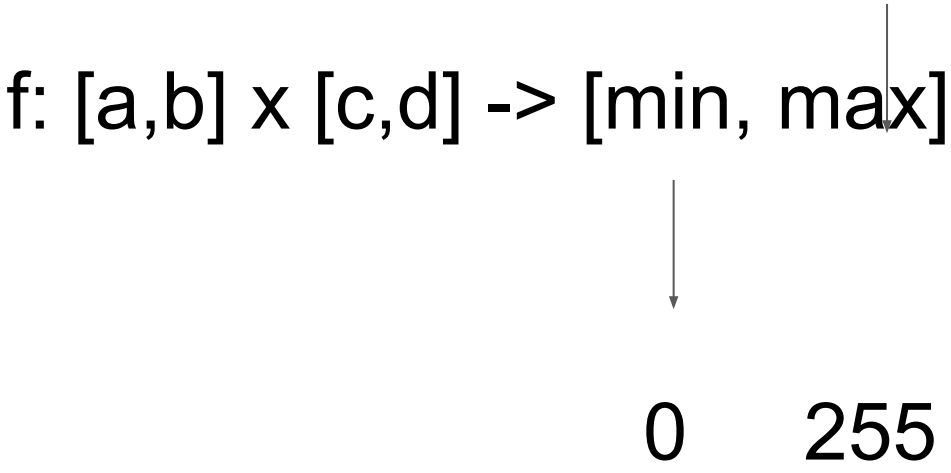
Images as functions



Images as functions

$$f: [a,b] \times [c,d] \rightarrow [min, max]$$

Images as functions



Color images as functions

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

The real Phyllis

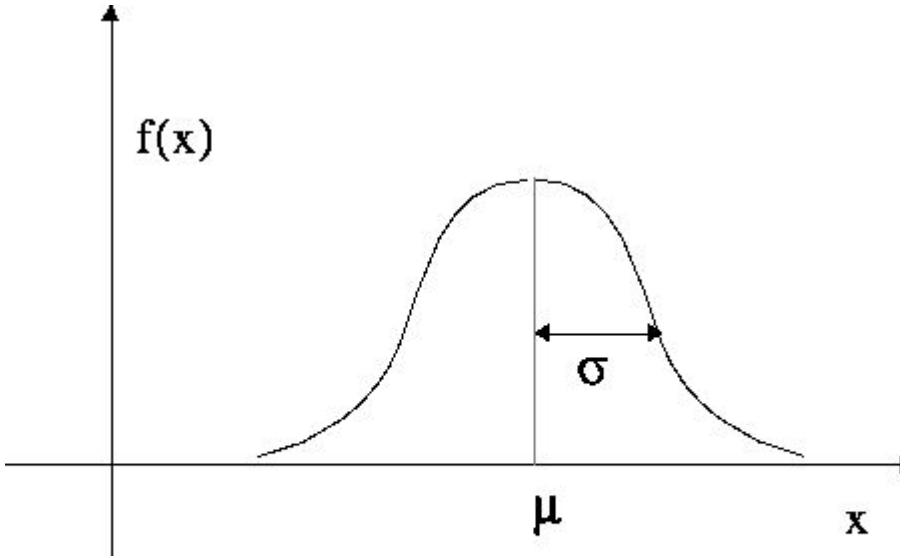
```
>> pd(40:60,30:40)

ans =

   152   122    99    83   122   120   154   150   123   141   112
   102   140   109   114   125   124    69   134   123   141   132
   138   160   135   109   104    89    91   145   128   102   154
   101   147   165    87    93    97   110   145   157   124   141
    58    68    96   115    80    98   137   160   145   168   166
    57   127    62    92   145   127    93   121   168   221   157
    69   108    74    71   156   119   106   140   156   161   158
   116   132   101    60   134   159   110   125   153   145   123
   109   119   130   113    80   176   121   108   111   152   133
   135    77   102   134   127   136   154   130   139   120   160
   175   127   112   145   153   125   160   126   103    94   166
   205   187   151    87   128   154   124   174    96   129   142
   206   211   207   171   153   146   173   194   125   129   164
   214   205   235   200   170   162   151   151   183   152   107
   225   199   211   203   125   145   154   181   201   184   137
   207   203   172   169   170   127   116    95   197   187   138
   171   208   150   157   184   153   109   119   148   182   138
   111   170   150   116   128   170   144   132   119   176   132
   101   172   168   130   112   131   116   136   129   137   121
   103   167   164   131   104   106    96   111   106   103   139
    92   136   146   138    92    63    73   101   120   126   134
```

Hands-on: Gaussian Noise

1_Gaussian_Noise.ipynb

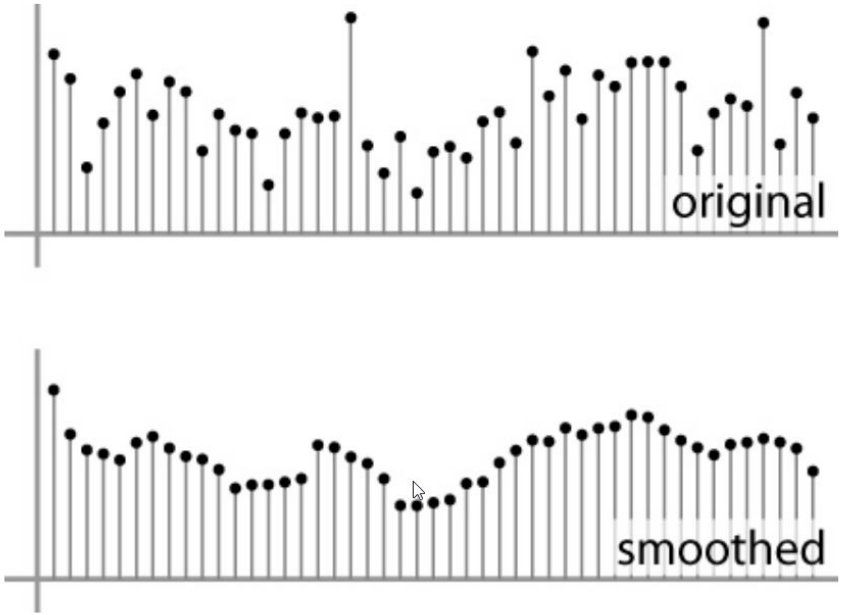


Hands-on: Gaussian Noise Image

2_Gaussian_Noise_Image.ipynb

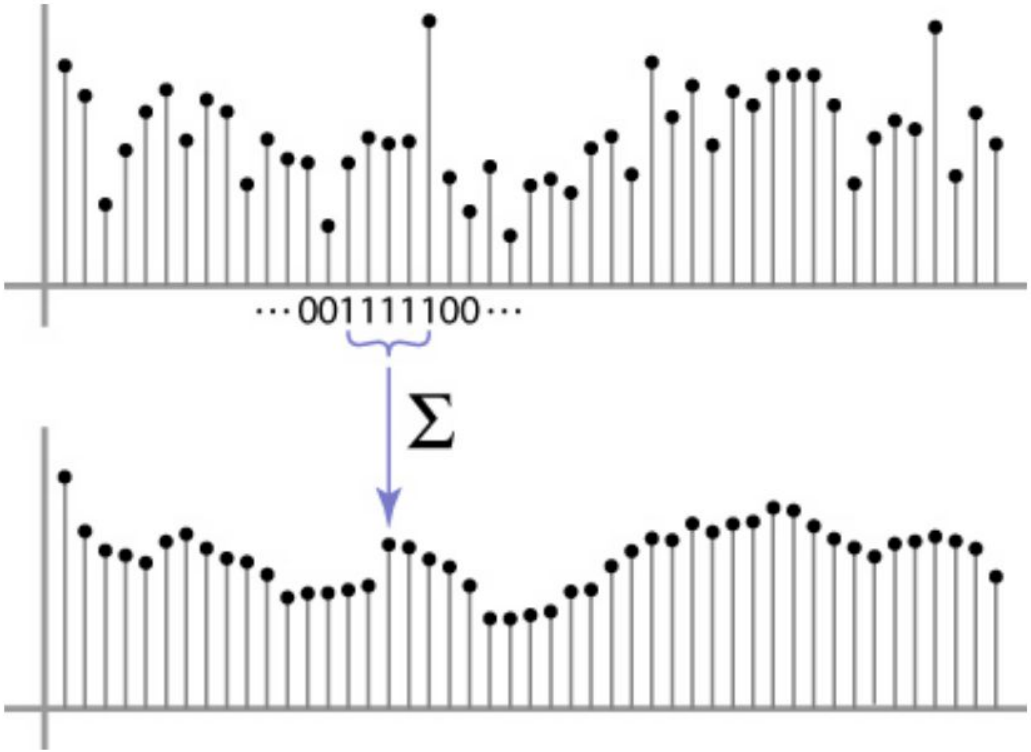
How do we remove noise in an image?

Smoothing -1D



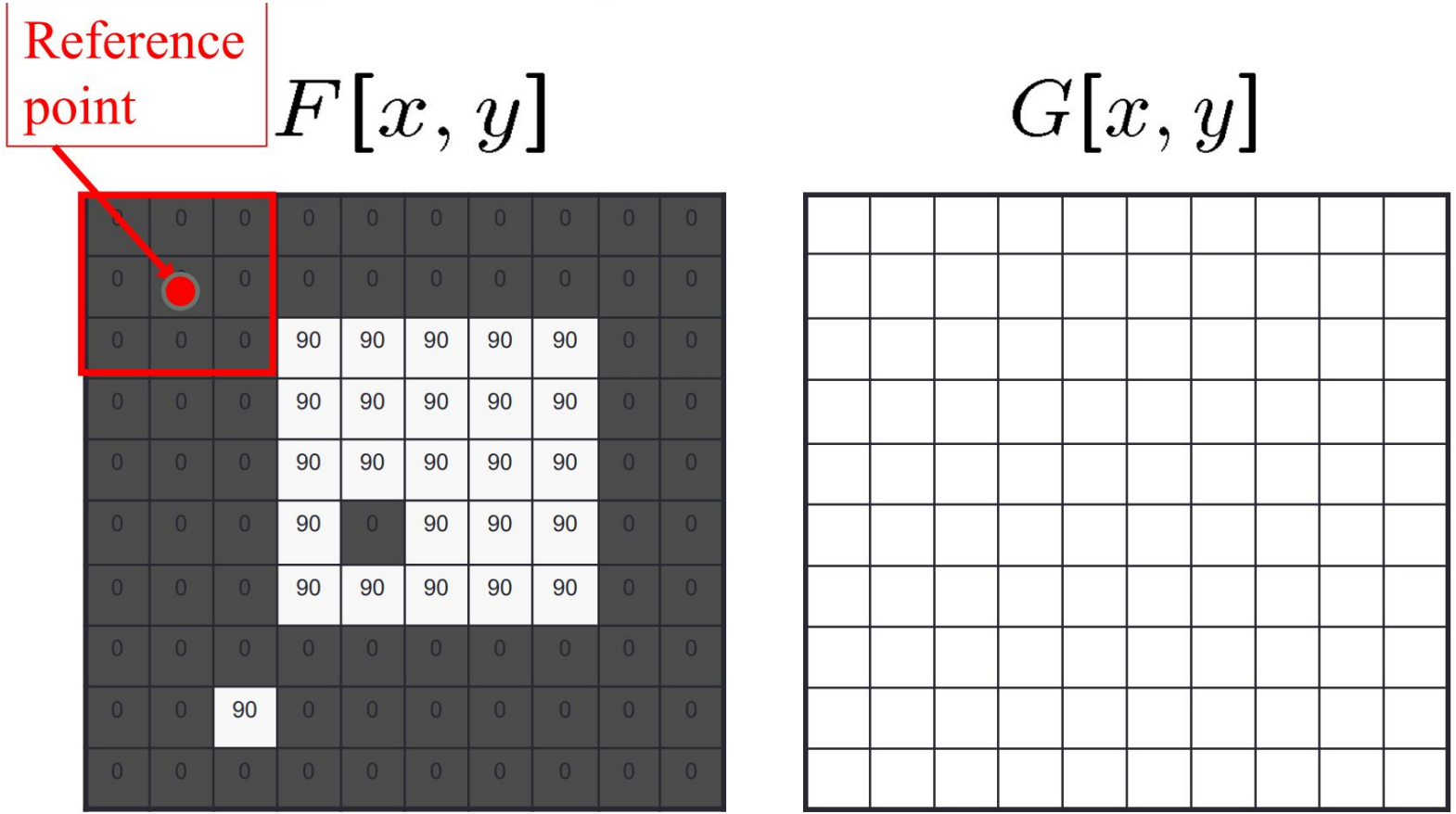
Source: S. Marschner

Weighted Moving Average



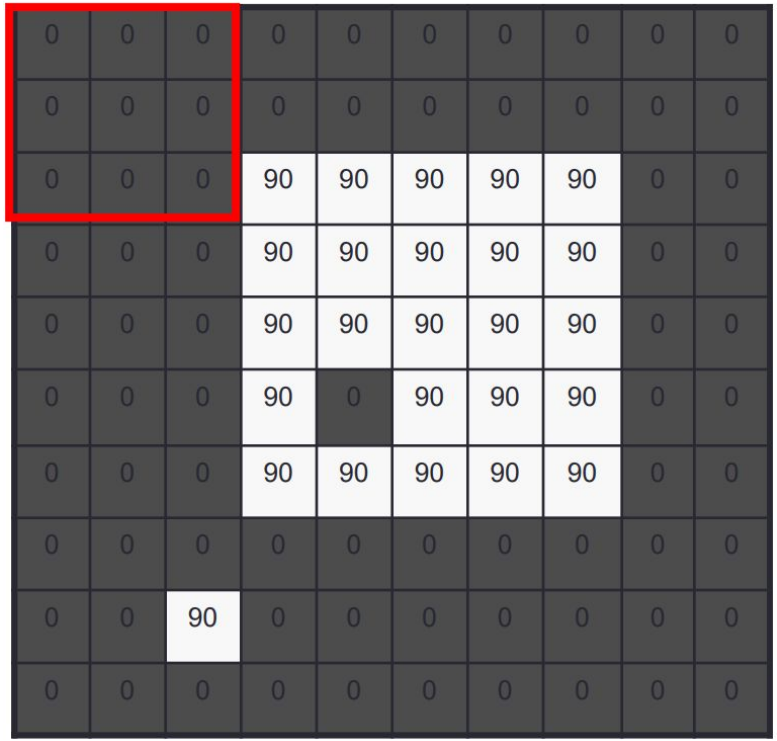
Source: S. Marschner

Weighted Moving Average in 2D

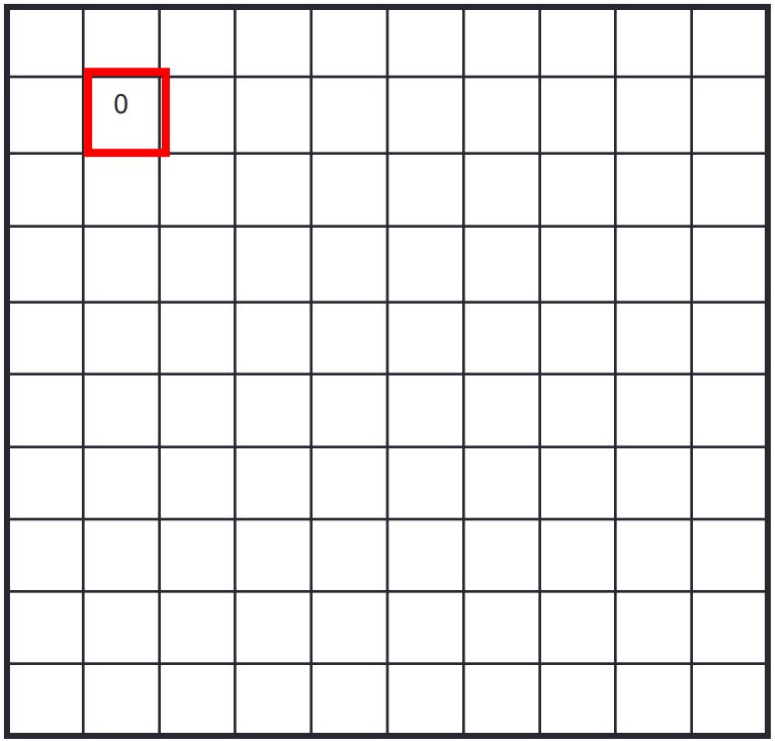


Weighted Moving Average in 2D

$F[x, y]$

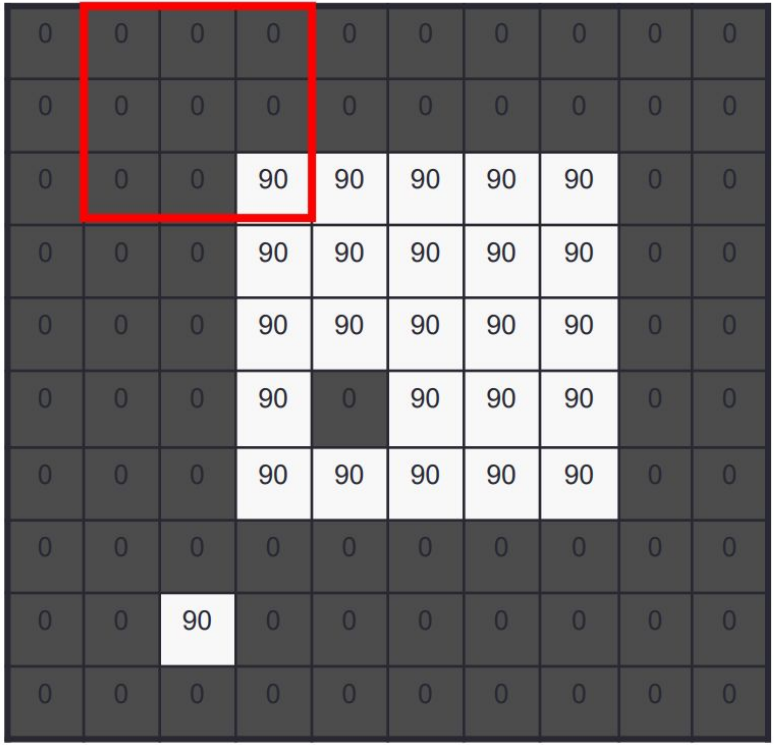


$G[x, y]$

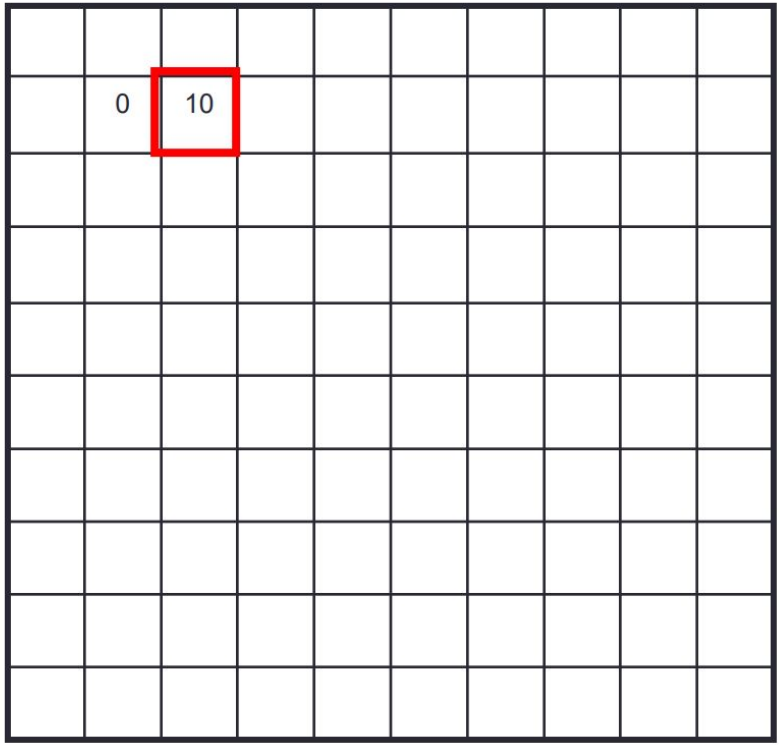


Weighted Moving Average in 2D

$F[x, y]$

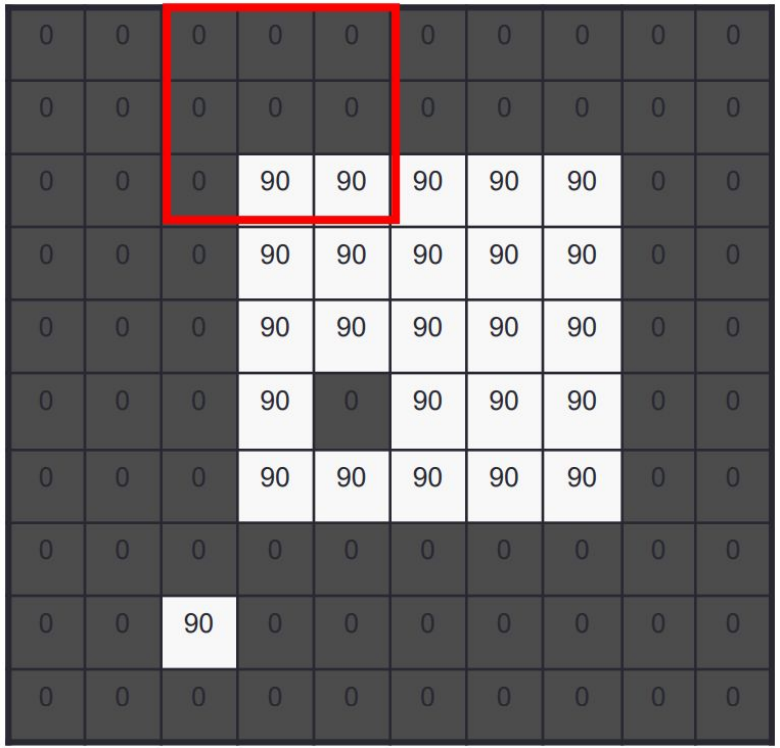


$G[x, y]$

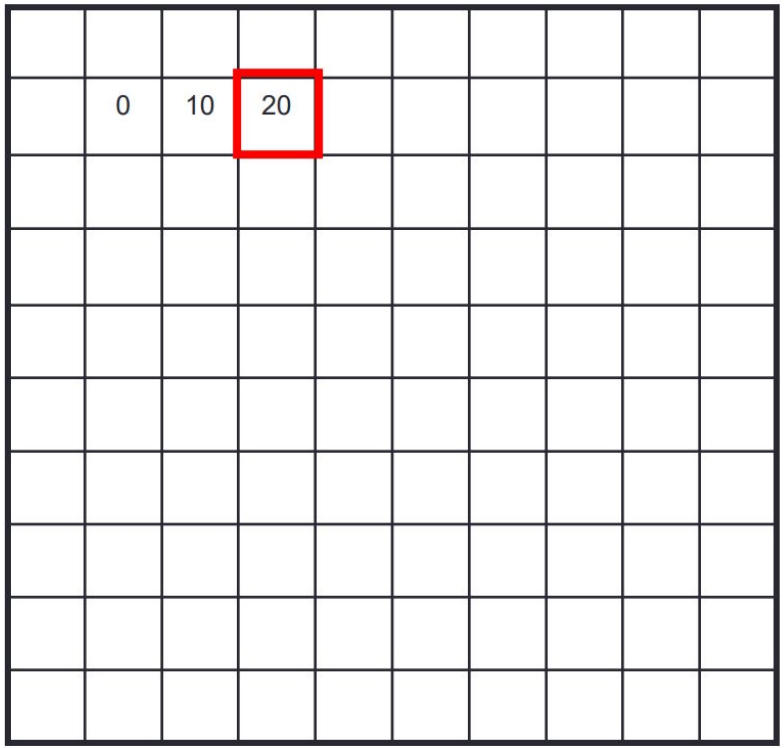


Weighted Moving Average in 2D

$F[x, y]$



$G[x, y]$



Weighted Moving Average in 2D

...

Weighted Moving Average in 2D

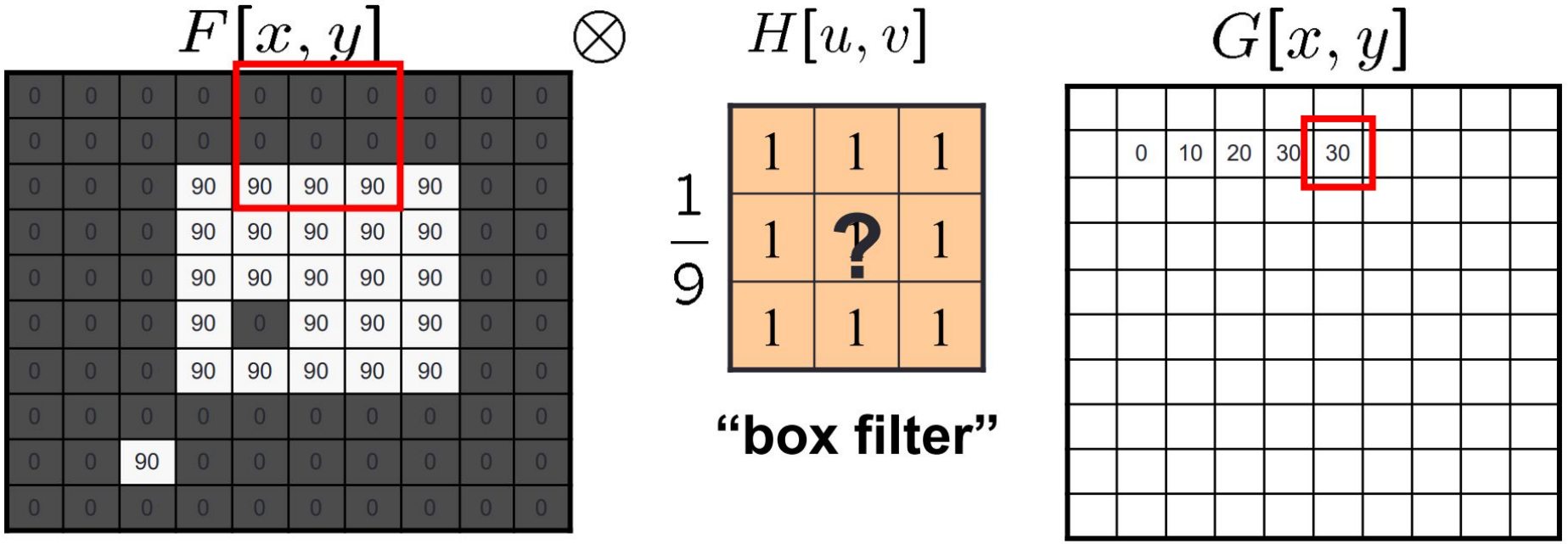
$F[x, y]$

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

$G[x, y]$

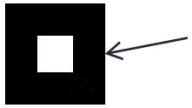
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Cross-correlation



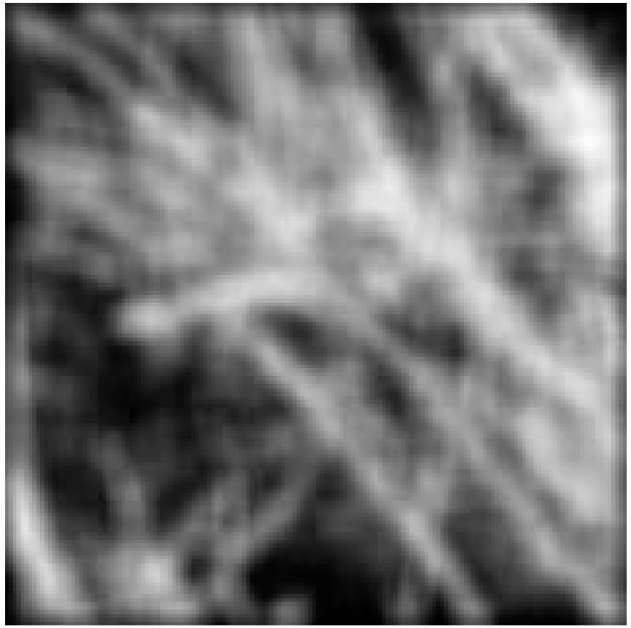
$$G = H \otimes F$$

Smoothing with a box filter



depicts box filter:
white = high value, black = low value

original



filtered

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

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

$F[x, y]$

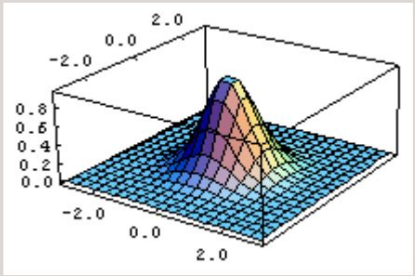
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

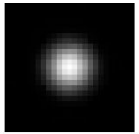
$H[u, v]$

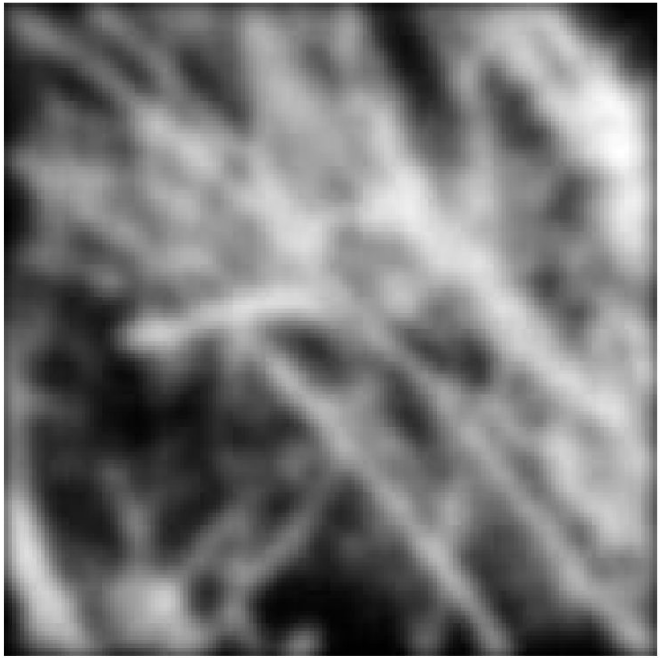
This kernel is an approximation of a Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Smoothing with Gaussian





Hands-on: Gaussian Filter

3_Gaussian_Filter.ipynb

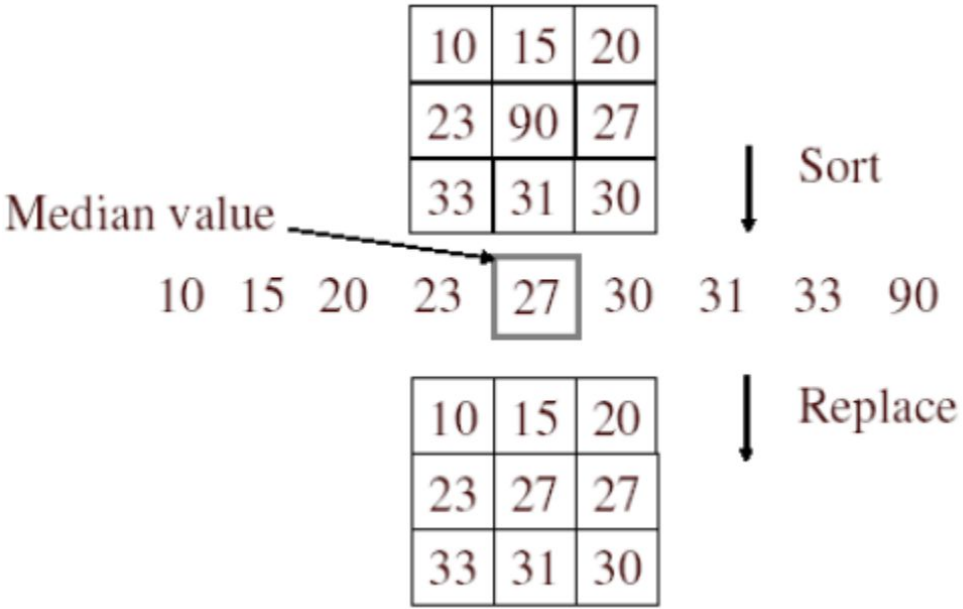
Hands-on: Linear Filter

4_Linear_Filter.ipynb

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

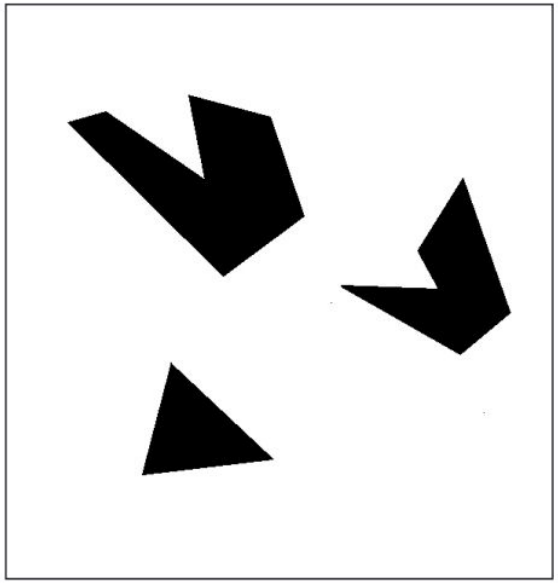
Hands-on: Linear Filter

5_Median_Filter.ipynb



How filters will allow us to extract features?

Template matching

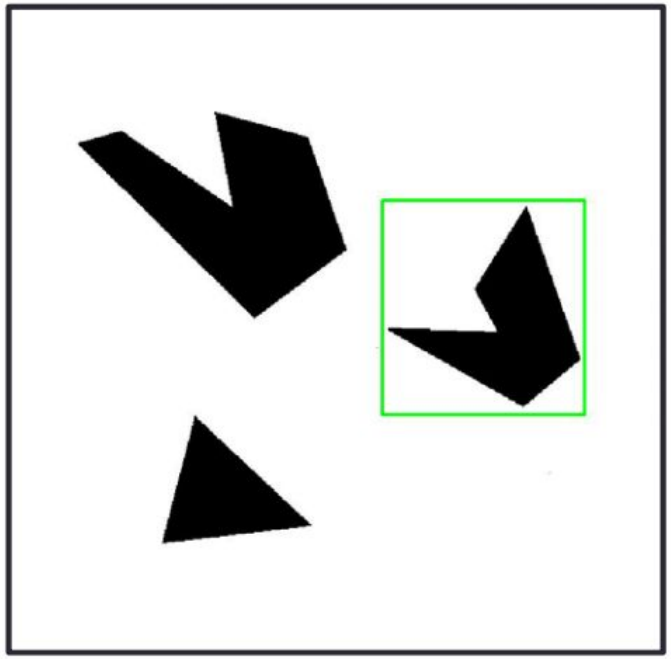


Scene

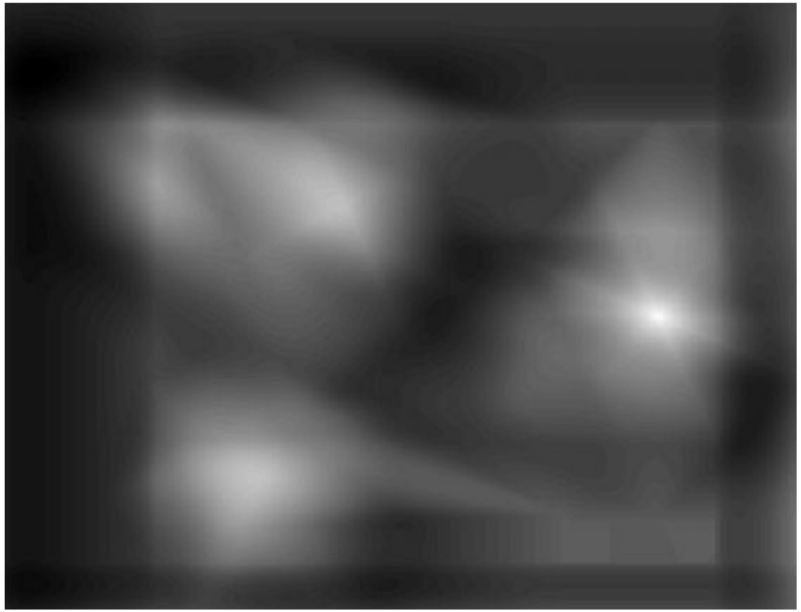
Template (mask)

A toy example

Template matching



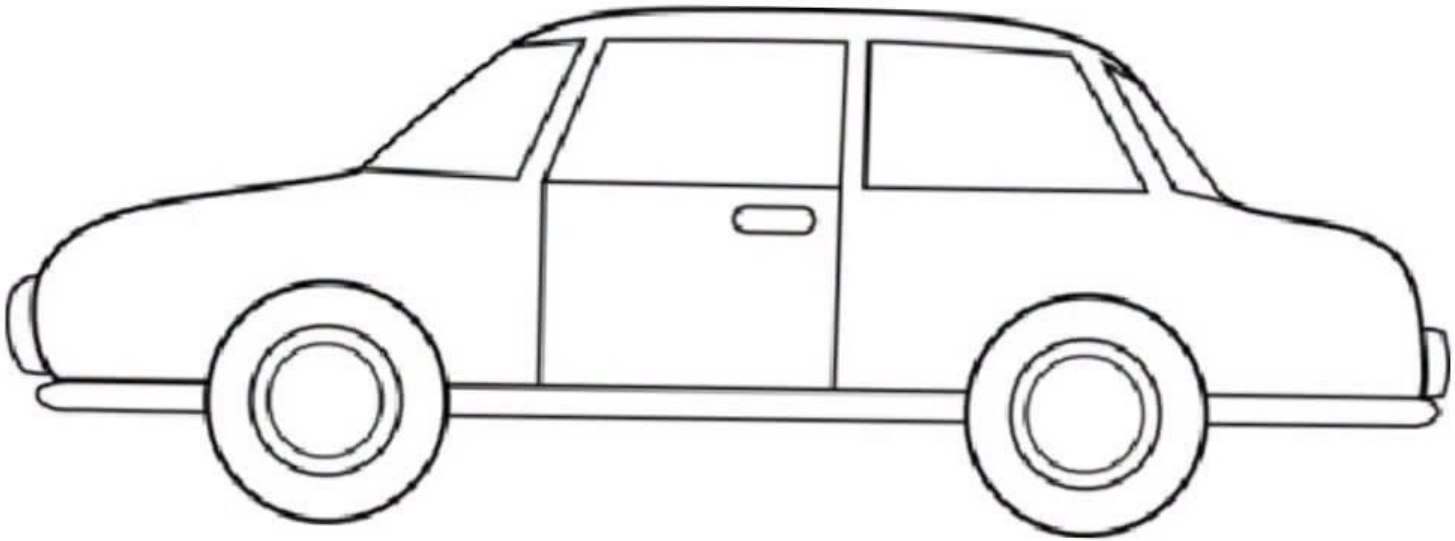
Detected template

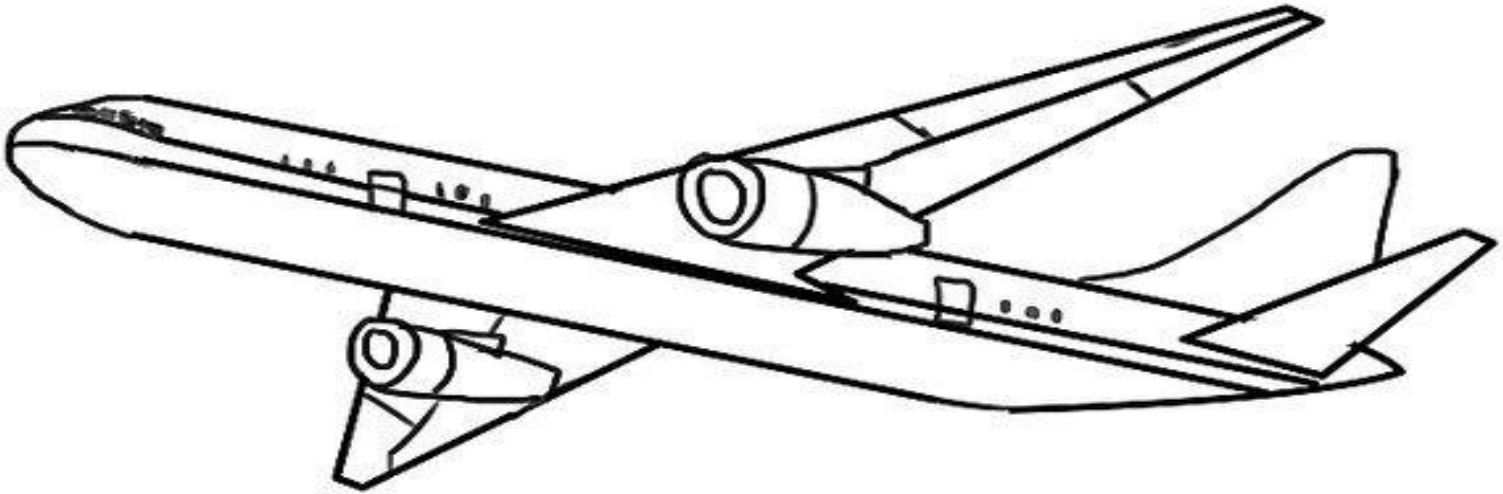


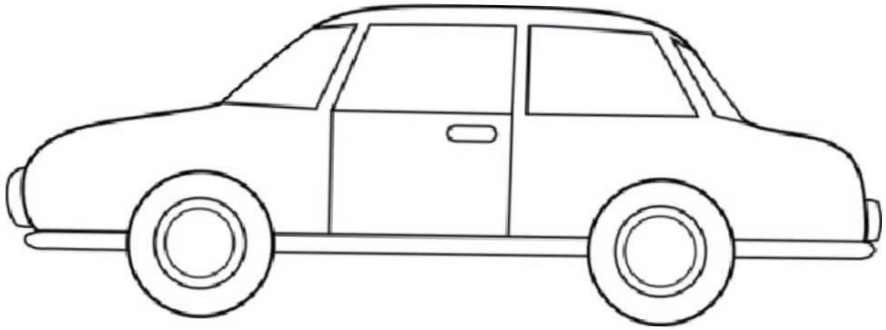
Correlation map

Hands-on: Template Matching

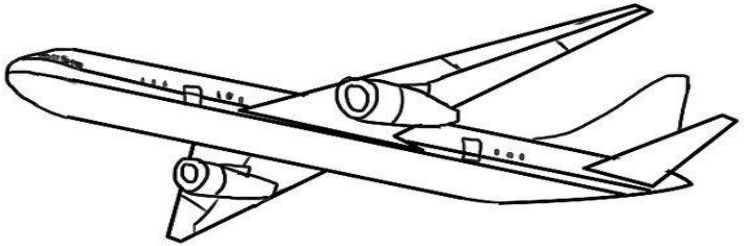
6_Template_Matching.ipynb

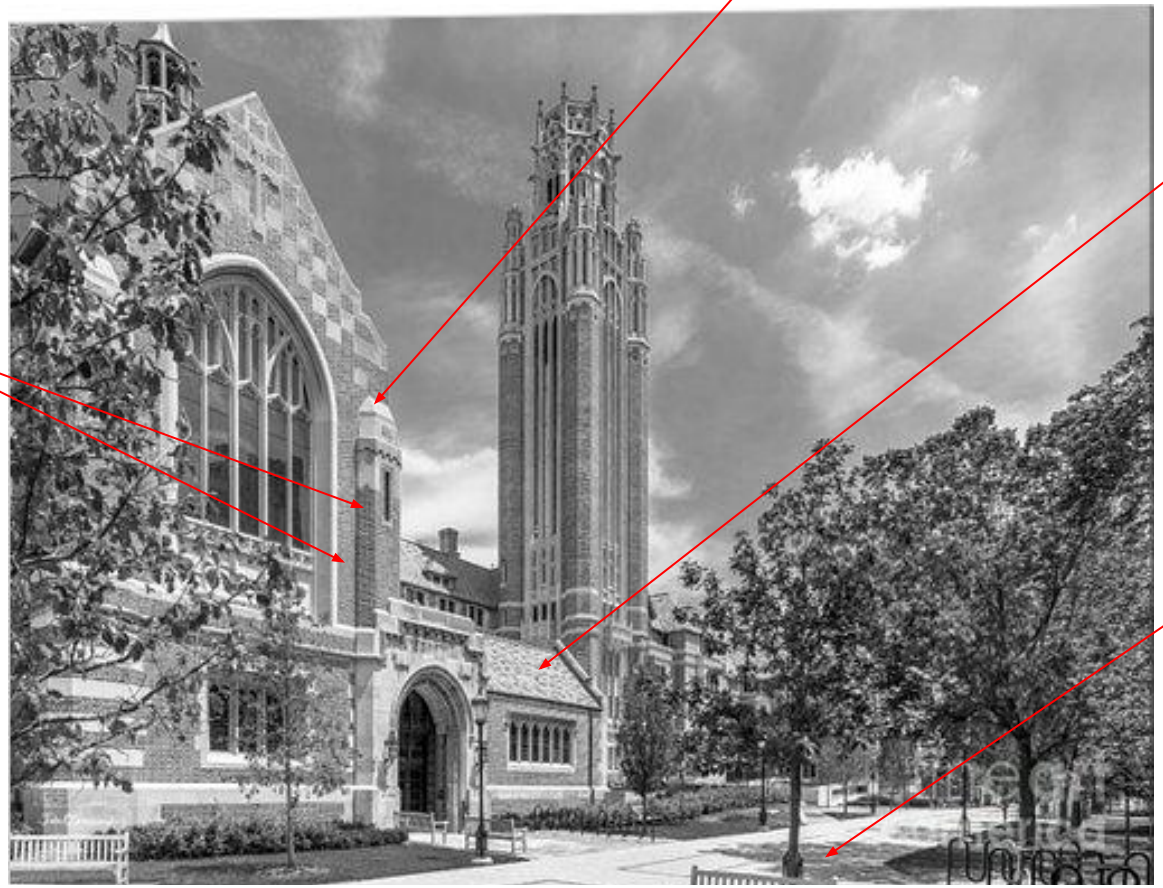




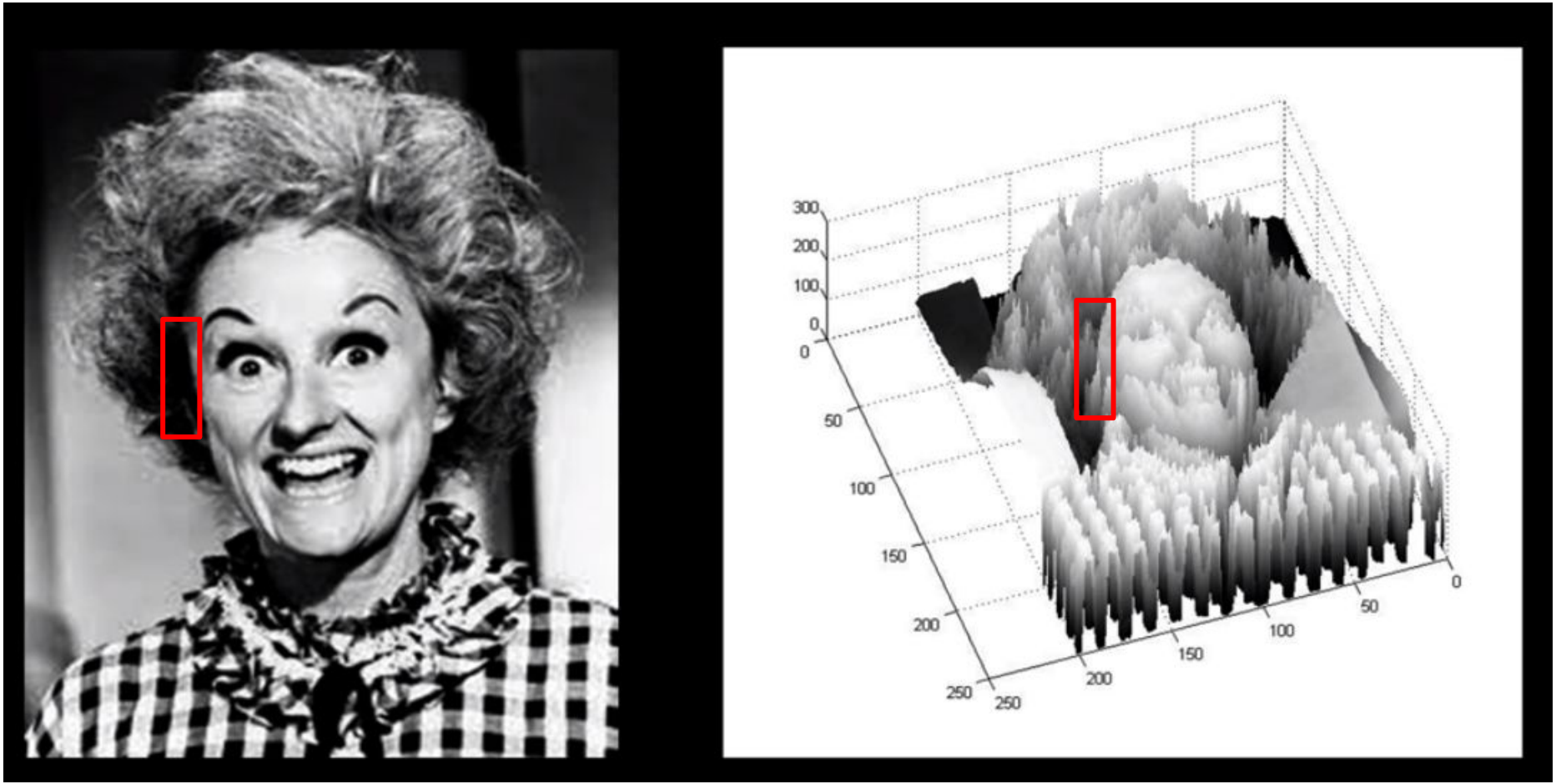


Edges seem to be important ...




Discontinuity in surface orientation
Reflectance change: appearance information, texture
Depth discontinuity: object boundary
Cast shadow

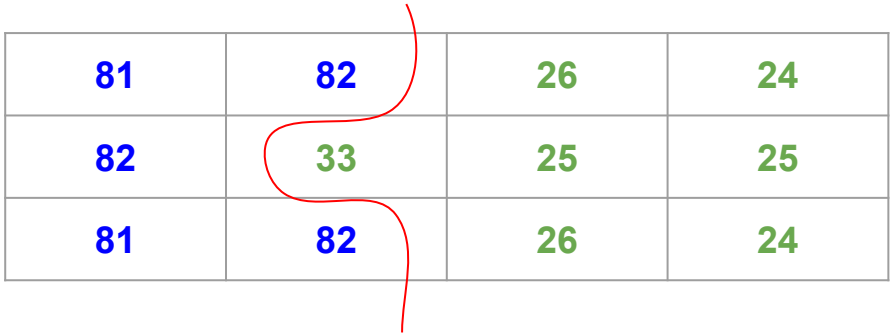
How we determine that a pixel at some location x,y is actually an edge pixel?


300
200
100
0
0
50
100
150
200
250
250
200
150
100
50
0

Edge detection

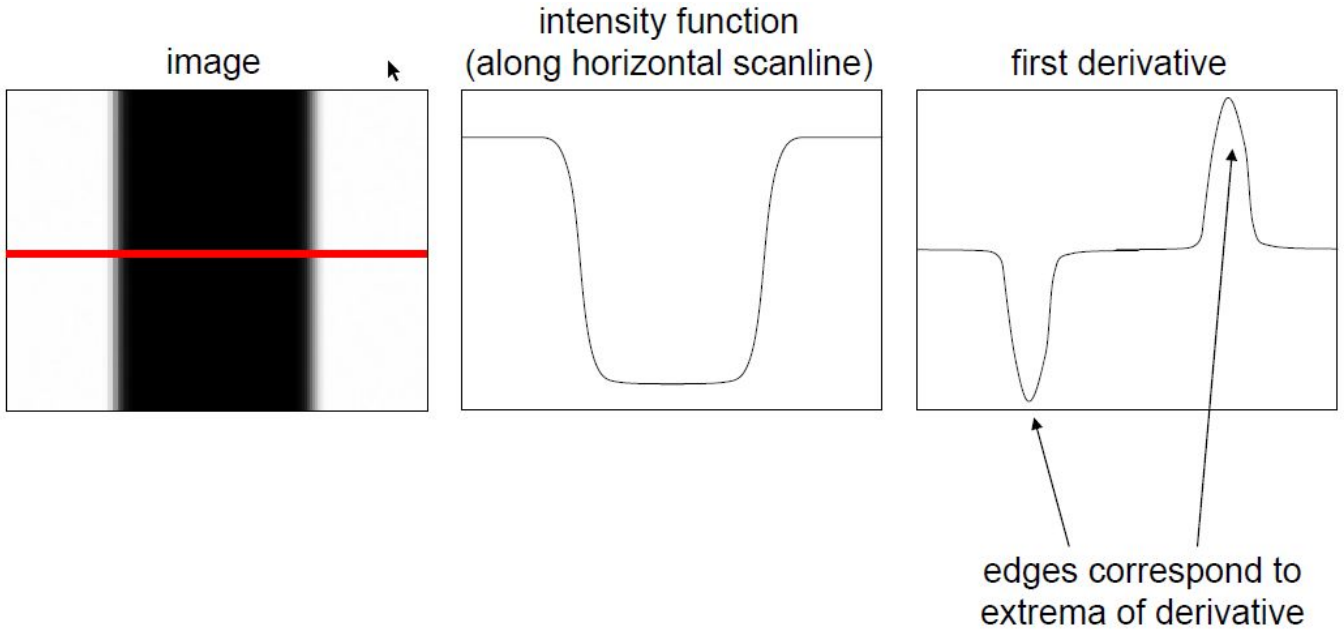
- **Idea: look for a neighborhood with strong signs of change.**

- Questions:
  - Neighborhood size
  - How to detect change



Changes in a function? => Derivatives

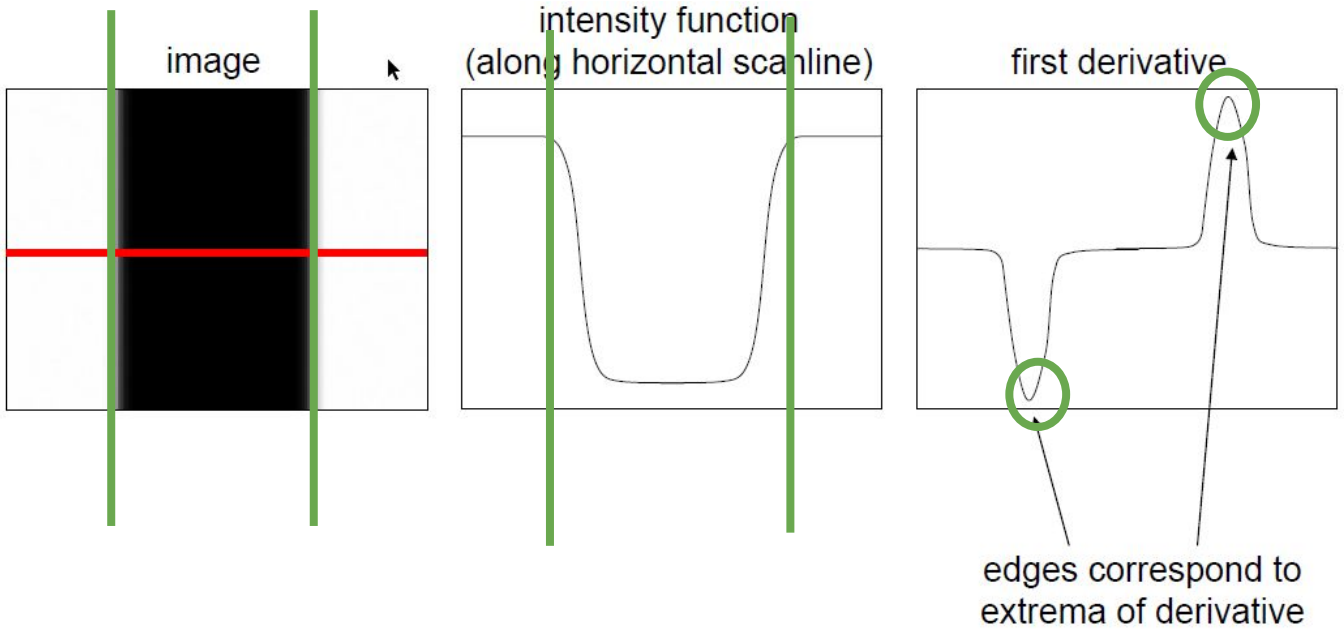
- An edge is a place of rapid change in the image intensity function



Source: S. Lazebnik

Changes in a function? => Derivatives

- An edge is a place of rapid change in the image intensity function



Source: S. Lazebnik

How we will find the peaks?

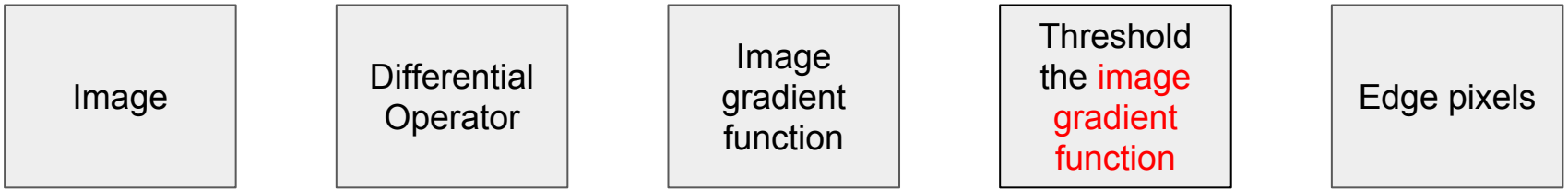
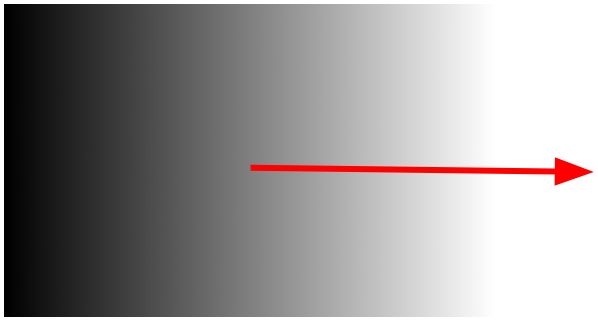


Image
Differential Operator
Image gradient function
Threshold the image gradient function
Edge pixels

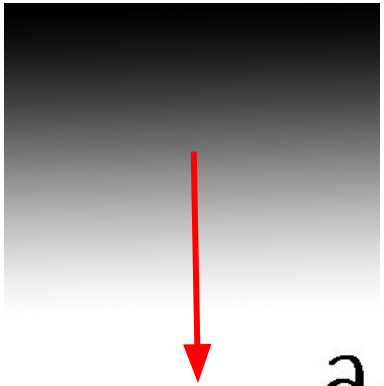
What is a gradient?

Image gradient

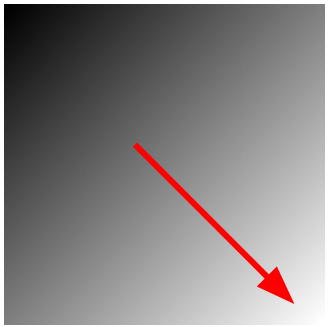
- The gradient of an image: $\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$



$\nabla f = [\frac{\partial f}{\partial x}, 0]$



$\nabla f = [0, \frac{\partial f}{\partial y}]$

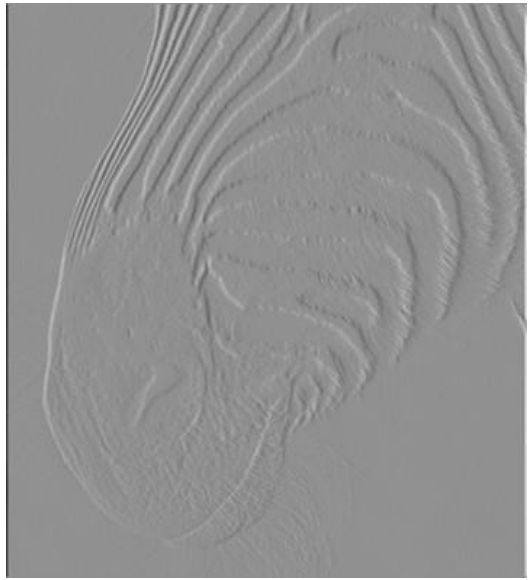


$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$

Finite differences - discrete gradient

$$\frac{\partial f(x,y)}{\partial x} \approx f(x+1,y) - f(x,y)$$

"right derivative"



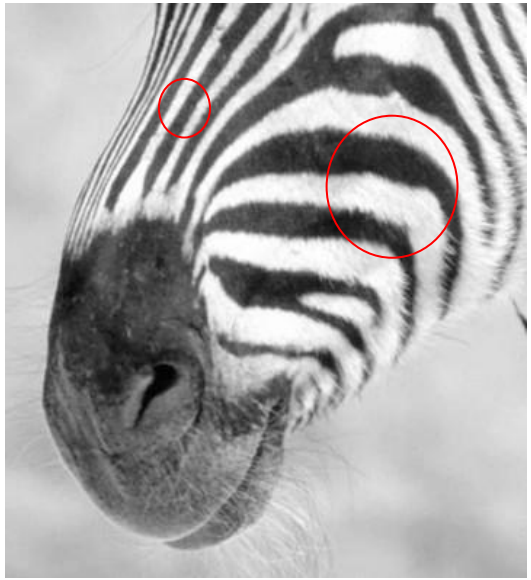
Finite difference in x or in y?

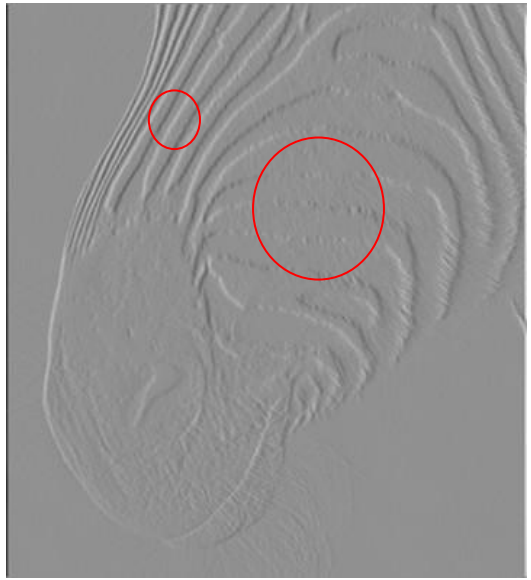
Source: D. Forsyth

Finite differences - discrete gradient

$$\frac{\partial f(x,y)}{\partial x} \approx f(x+1,y) - f(x,y)$$

"right derivative"

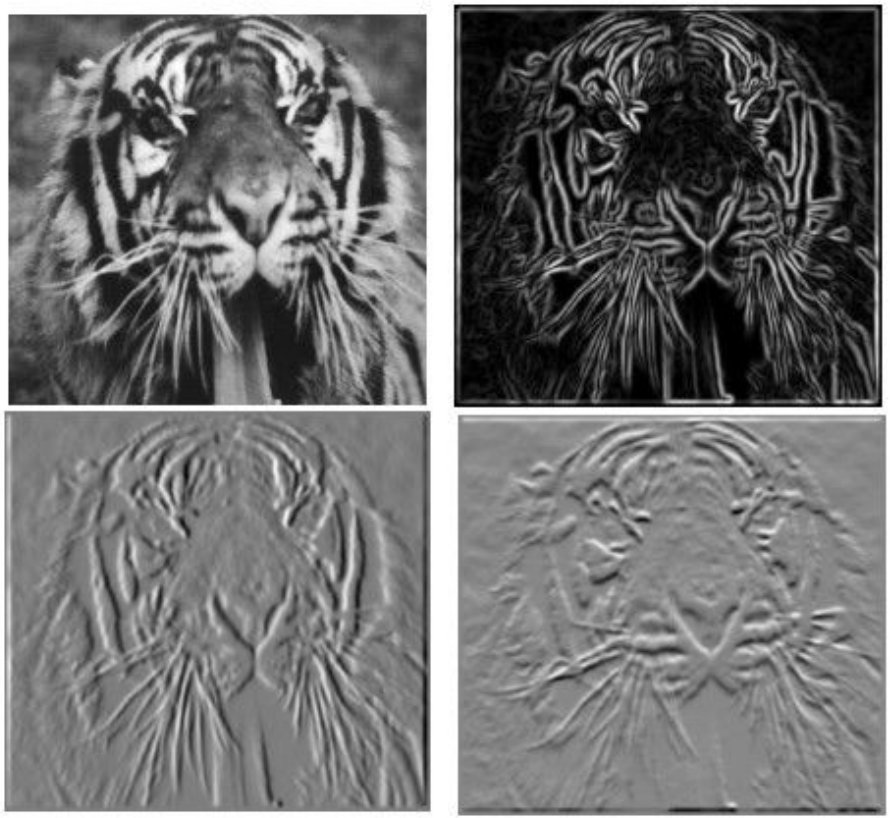




Finite difference in x or in y?

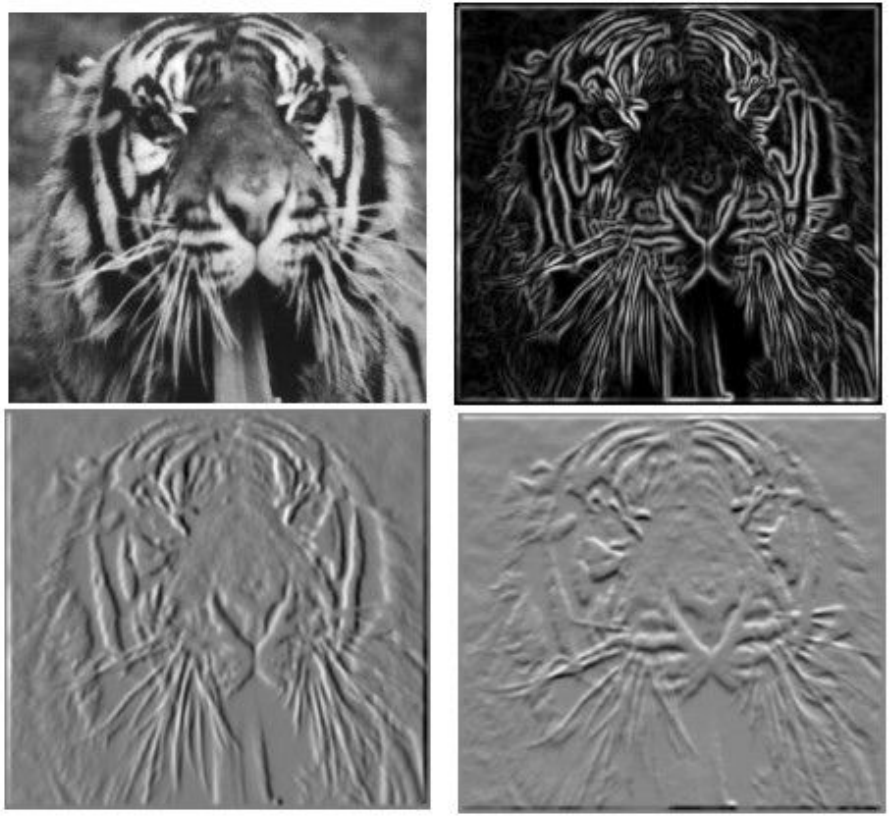
Source: D. Forsyth

Partial derivatives of an image



In the images at bottom, which one shows changes in x direction and which one in y direction?

Partial derivatives of an image



$$\frac{\partial f(x,y)}{\partial x}$$

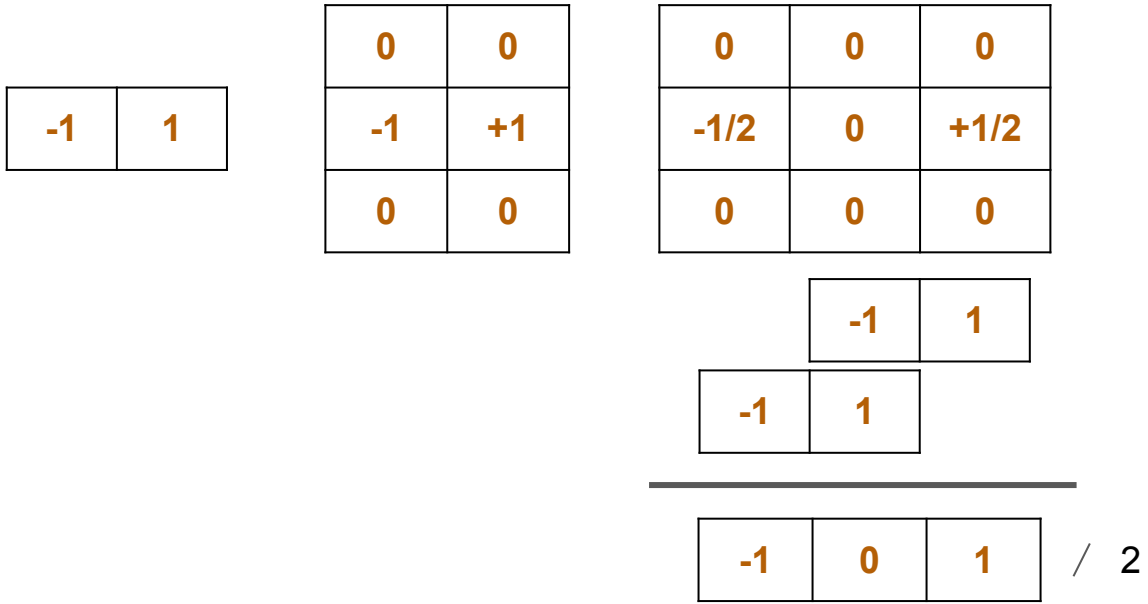
-1	1

(correlation filters)

$$\frac{\partial f(x,y)}{\partial y}$$

1
-1

Discrete gradient - operator



Hands-on: Edge detection

7_Edge_Detection.ipynb

- Sobel operator: $\frac{1}{8} *$

-1	0	1
-2	0	2
-1	0	1

S_x

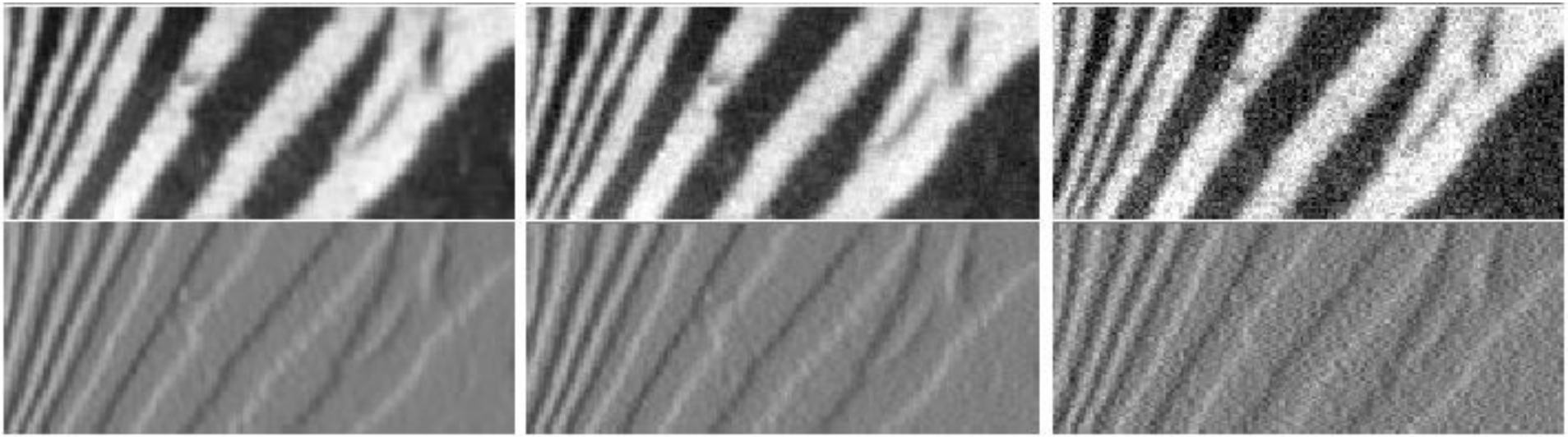
$\frac{1}{8} *$

1	2	1
0	0	0
-1	-2	-1

S_y

- Other gradient masks: Sobel, Prewitt, Roberts

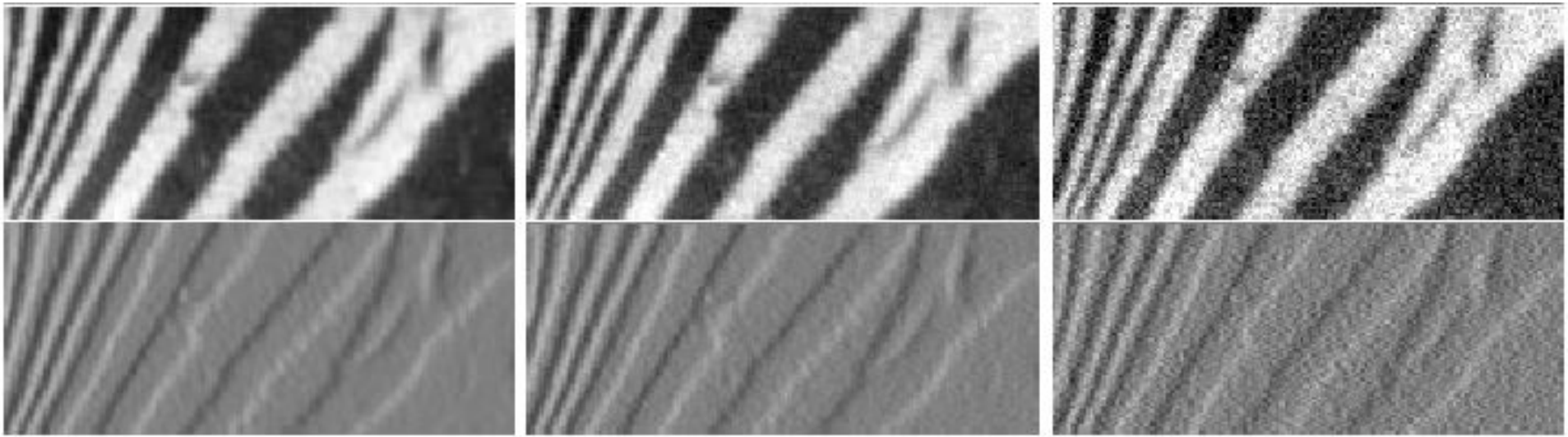
But, in the real world...



What we can apply to eliminate noise?

Source: S. Lazebnik

But, in the real world...



Apply Gaussian filtering => extract derivative of Gaussian filter

Source: S. Lazebnik

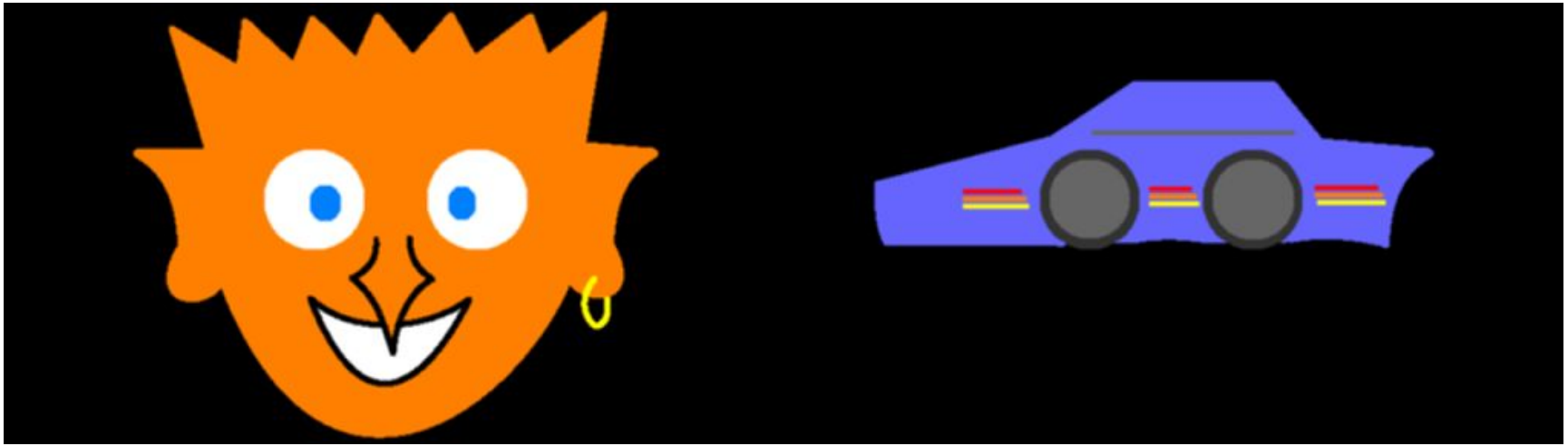
Hands-on: Canny edge operator

8_Canny_Edge_Operator.ipynb

1. **Filter image with the derivative of Gaussian**
2. **Find magnitude and orientation of the gradient**
3. **Non-maximum suppression:**
    a. **Thin multi-pixel wide “ridges” down to single pixel width**
4. **Linking and thresholding(hysteresis):**
    a. **Define two thresholds: low and high**
    b. **Use the high threshold to start edge curves and the low threshold to continue them**

Hands-on: Canny edge operator

8_Canny_Edge_Operator.ipynb



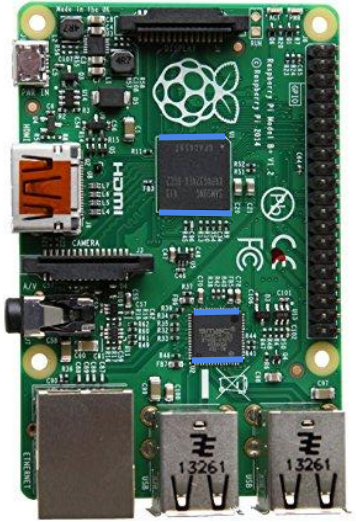
Find the common edges in the two pictures

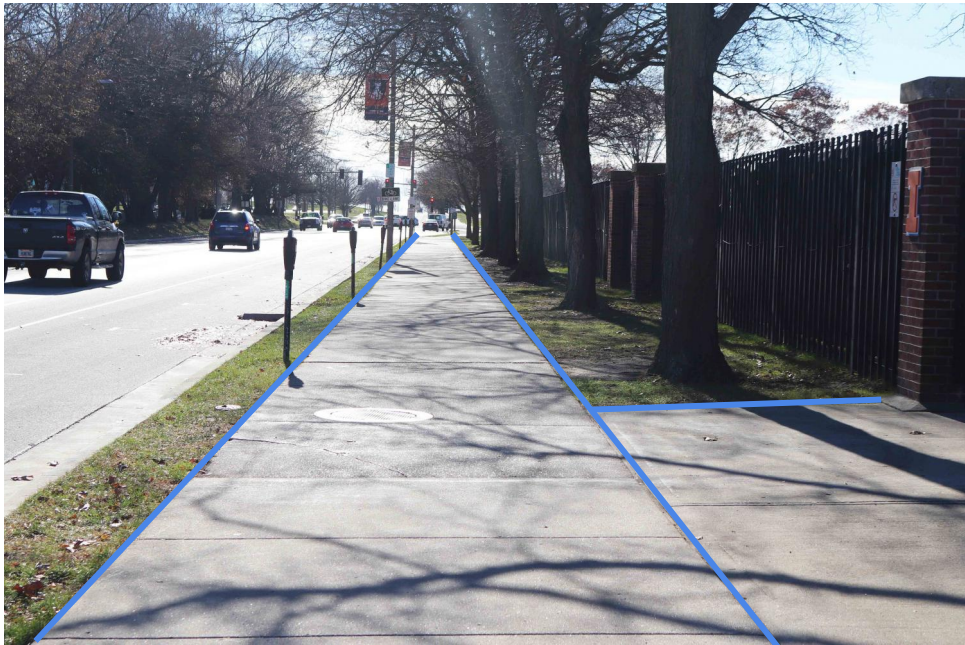
Image processing: $F : I(x,y) \rightarrow I'(x,y)$

Computer vision: $F : I(x,y) \rightarrow$ **good stuff**

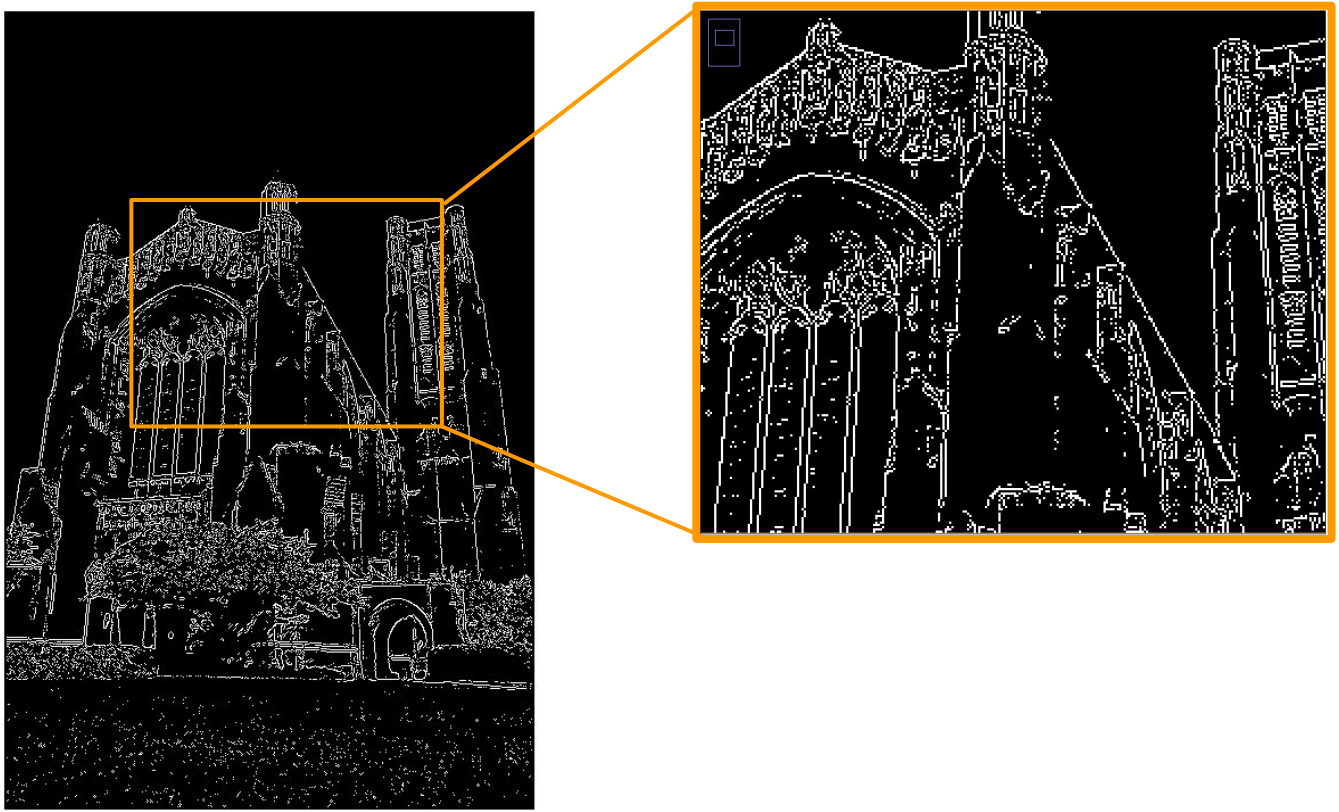
stuff e.g.: lines, circles, cars ===> called parametric models

Line fitting





Line fitting



Line fitting

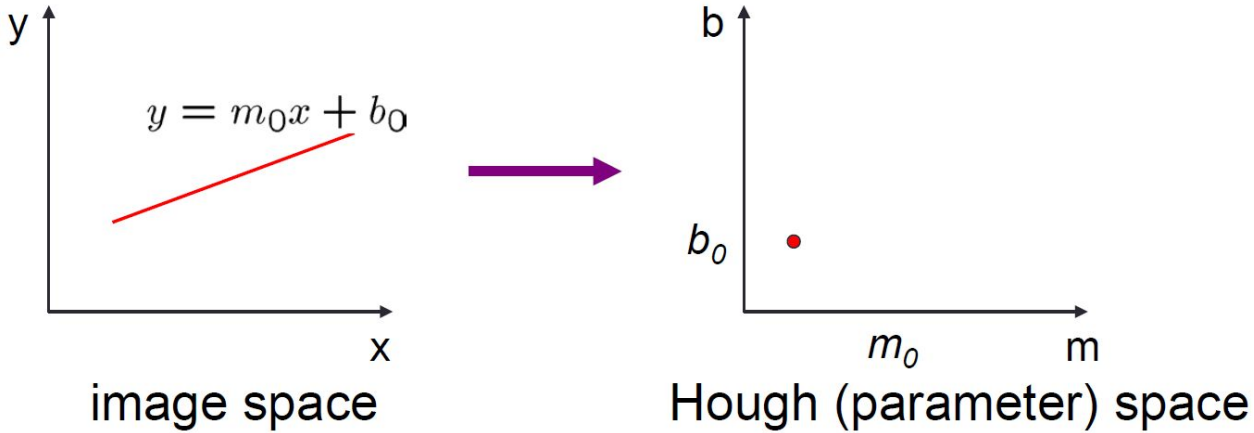
How do we go from edges to lines?

Voting

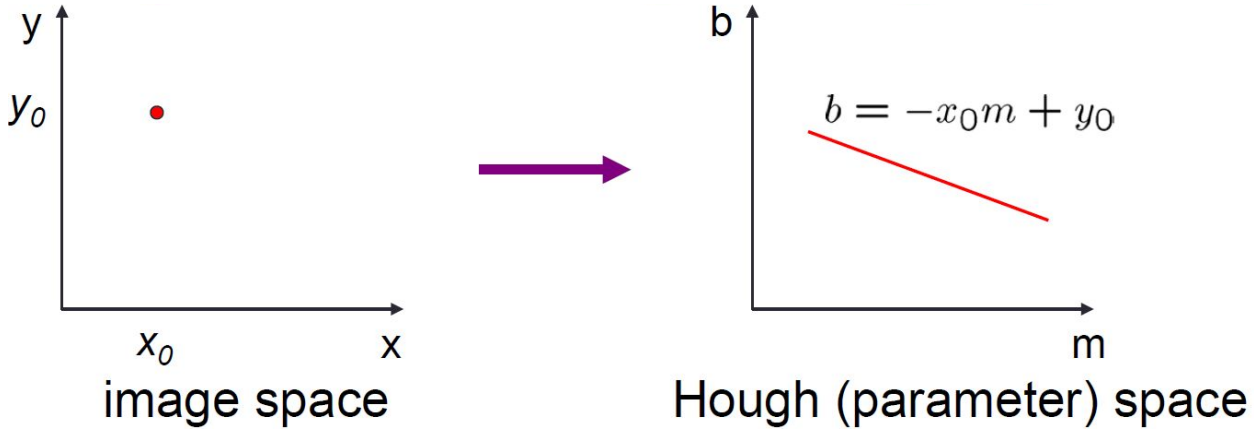
- General technique where we let the features vote for all models that are compatible with it:
    1. Cycle through features, each casting votes for model parameters.
    2. Look for model parameters that receive a lot of votes.

Fitting lines

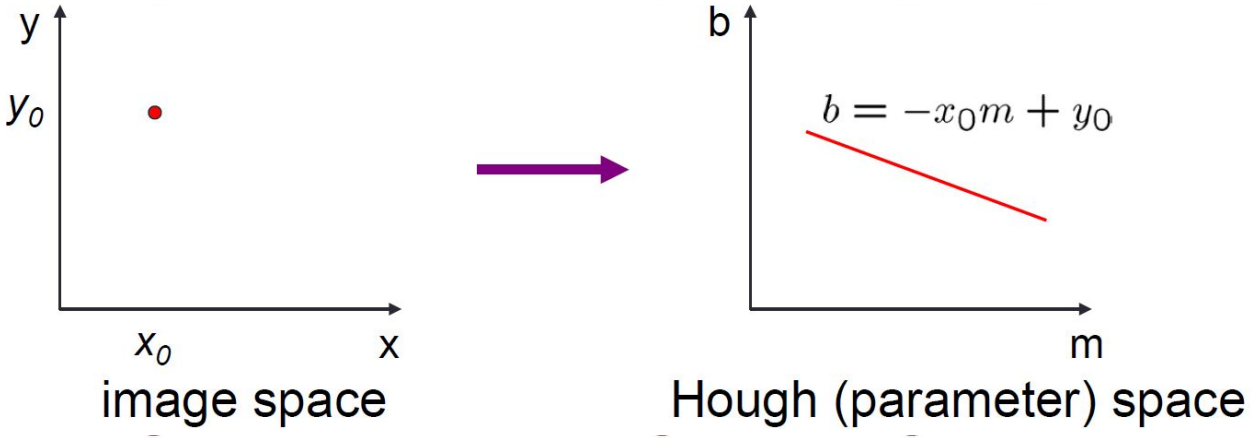
- **Hough Transform**
  - Each edge point votes for compatible lines
  - Look for lines that get many votes

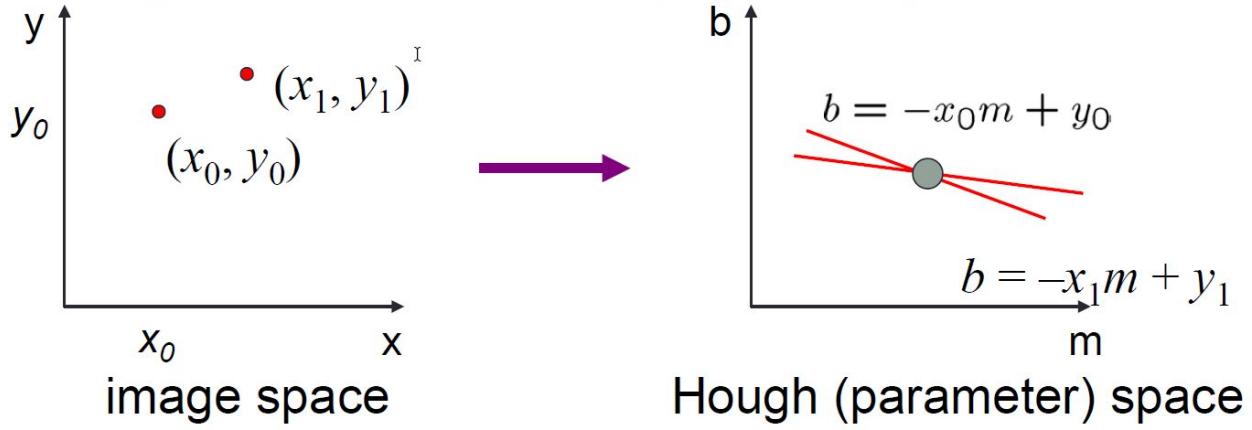


Hough Transform



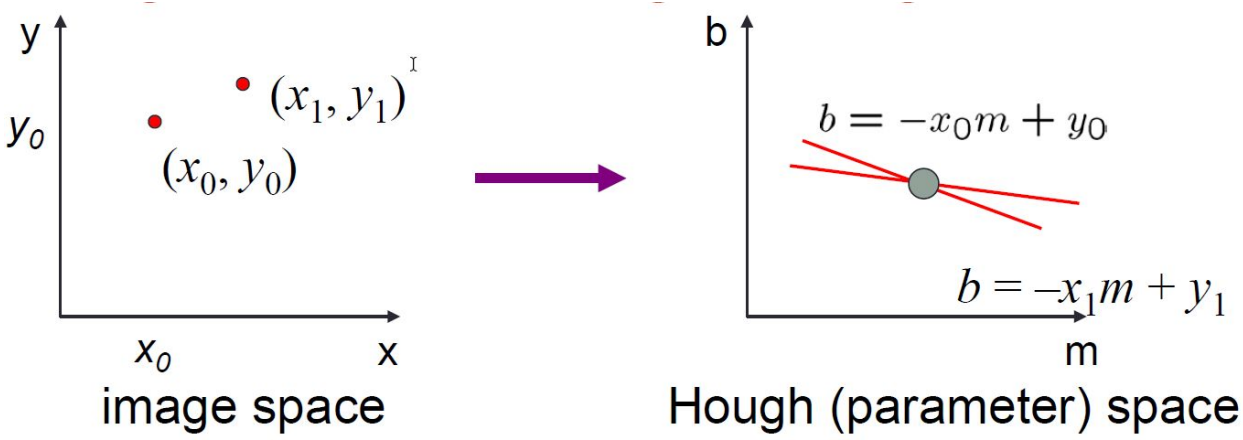
Hough Transform



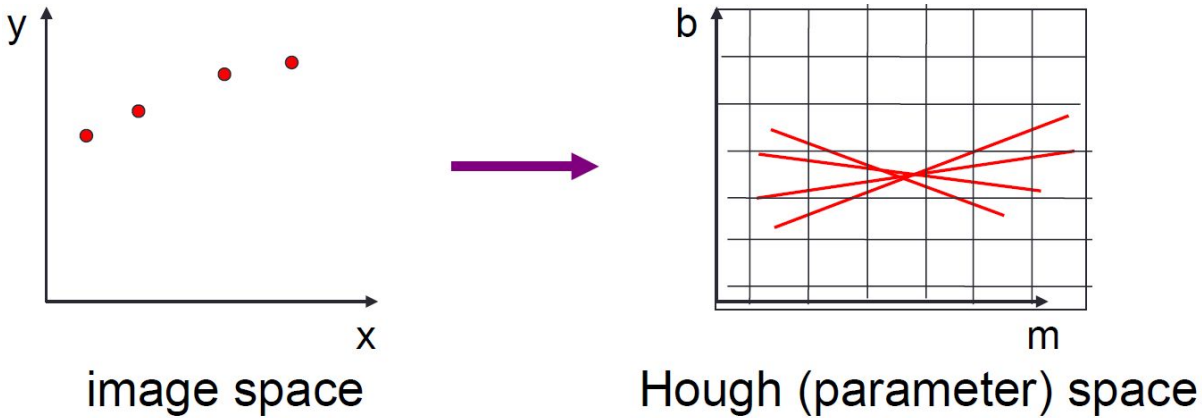


Hough Transform

What line will be consistent with both points?

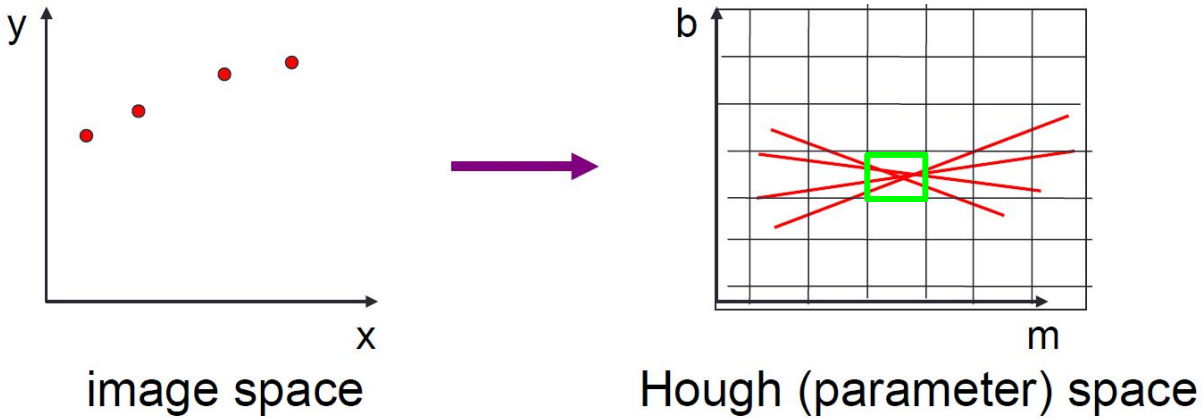


Hough Transform - the algorithm



- Each edge point in image space votes for a set of possible parameters in Hough space

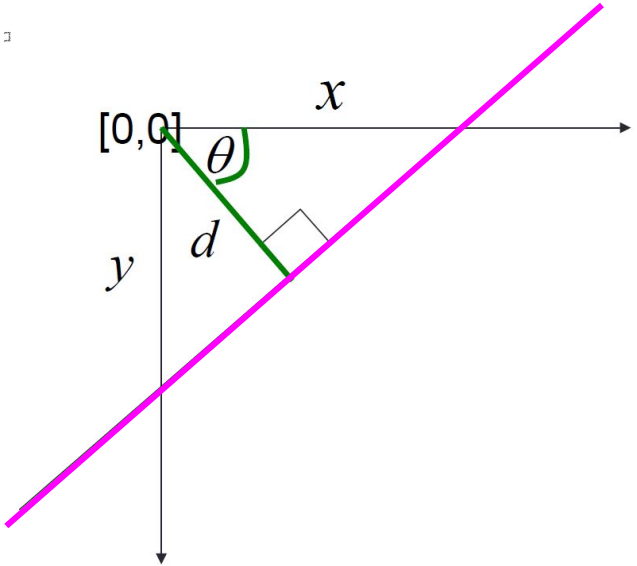
Hough Transform - the algorithm



- Each edge point in image space votes for a set of possible parameters in Hough space

Hough Transform - use a polar representation of lines

- Issues with usual (m,b) parameter space: can take on infinite values, undefined for vertical lines



d : perpendicular distance from line to origin

θ : angle the perpendicular makes with the x-axis

$$x\cos\theta - y\sin\theta = d$$

Hands-on: Hough transform

9_Hough_Transform.ipynb

Extensions:

- Using the gradient
  - θ = gradient at (x,y) $\rightarrow$ to reduce the voting time
- Can be used for detecting circles, squares, or any other shape

What is an image?

Until nou: a function - a 2D pattern of intensity values

Now: a 2D projection of 3D points

- **Features and Matching**
  - **Corner detection**
  - **Panorama reconstruction**

https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/

Join the RCC's image analysis and visualization Slack team