CMPE 185 Autonomous Mobile Robots

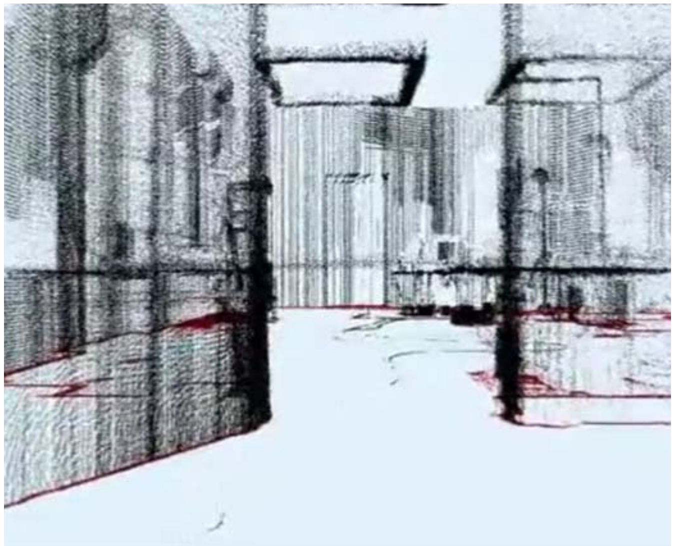
Perception: Feature Extraction II & Computer Vision and Image Processing

Dr. Wencen Wu
Computer Engineering Department
San Jose State University

Line Extraction from a Point Cloud

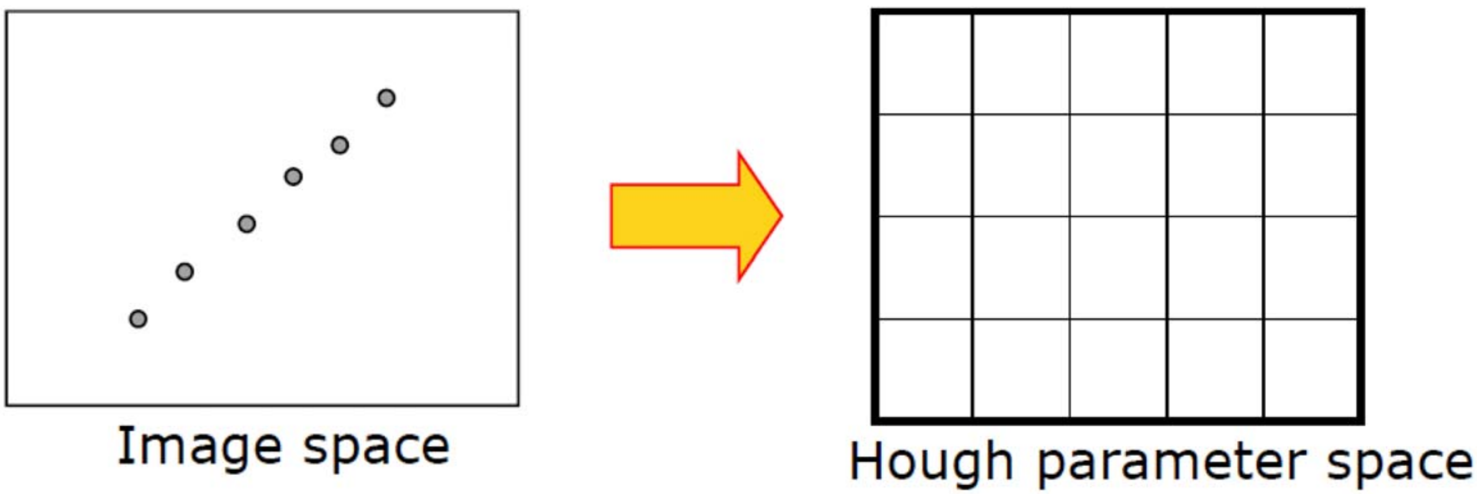
- Extract lines from a point cloud (e.g. range scan)
- Three main problems:
  - How many lines are there?
  - **Segmentation**: Which points belong to which line?
  - **Line Fitting/Extraction**: Given points that belong to a line, how to estimate the line parameters?

- Algorithms we will see:
  - Split-and-merge
  - Linear regression
  - RANSAC
  - Hough-Transform



Line Extraction – Hough-Transform

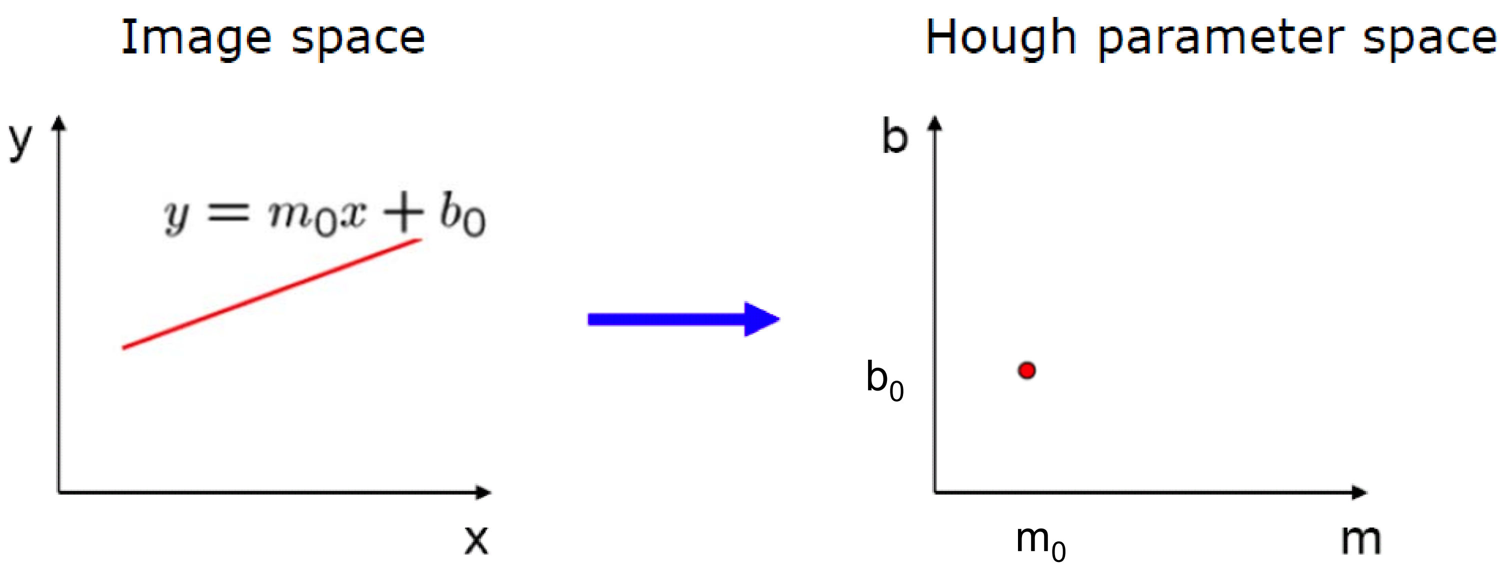
- Points vote for plausible line parameters
- Hough-Transform: maps image-space into Hough-space
- **Hough-space**: voting accummulator, parametrized w.r.t. line characteristics



1. P. Hough, Machine Analysis of Bubble Chamber Pictures, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959
2. J. Richard, O. Duda, P.E. Hart (April 1971). "Use of the Hough Transformation to Detect Lines and Curves in Pictures". Artificial Intelligence Center (SRI International)

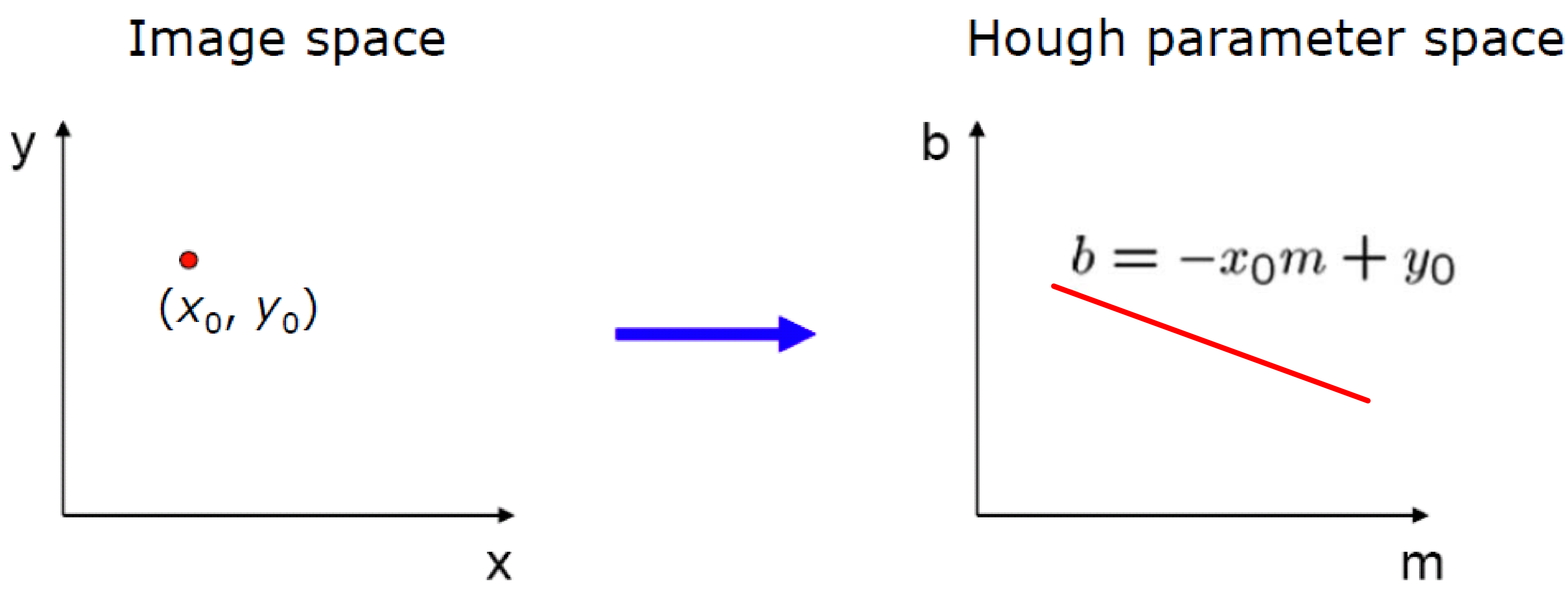
Line Extraction – Hough-Transform

- A line in the image corresponds to a point in Hough space



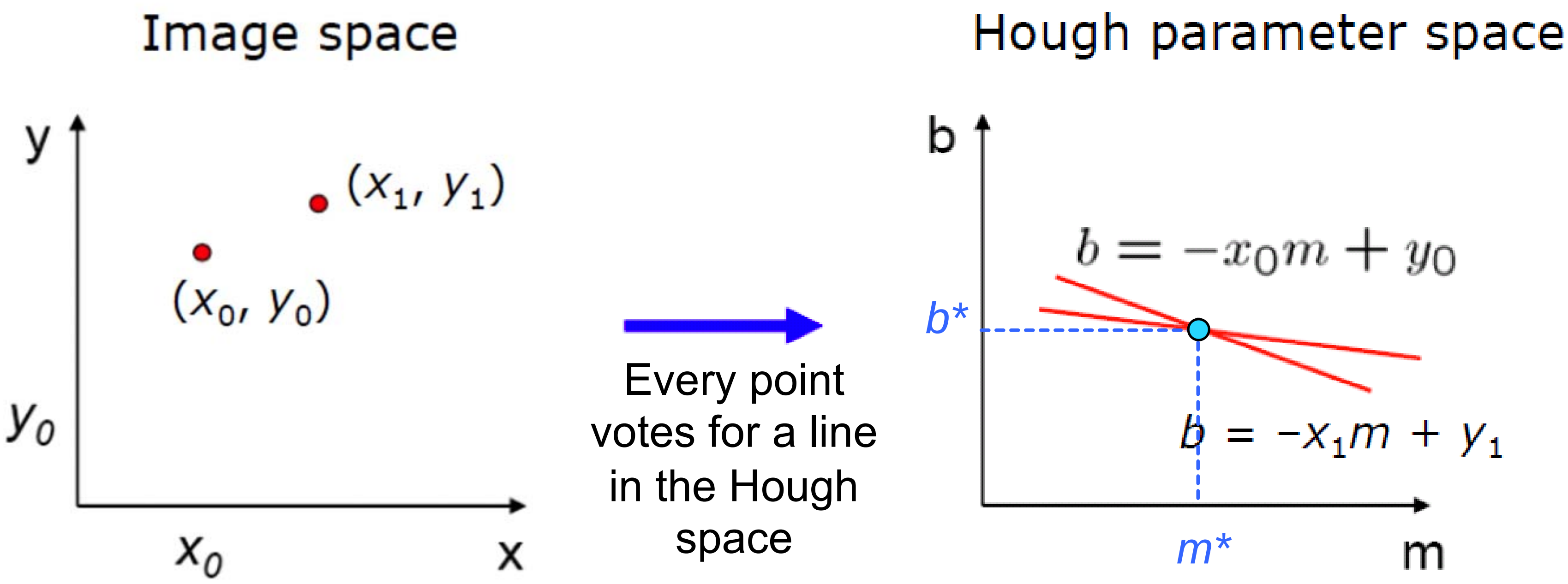
Line Extraction – Hough-Transform

- What does a point (x_0, y_0) in the image space map to in the Hough space?

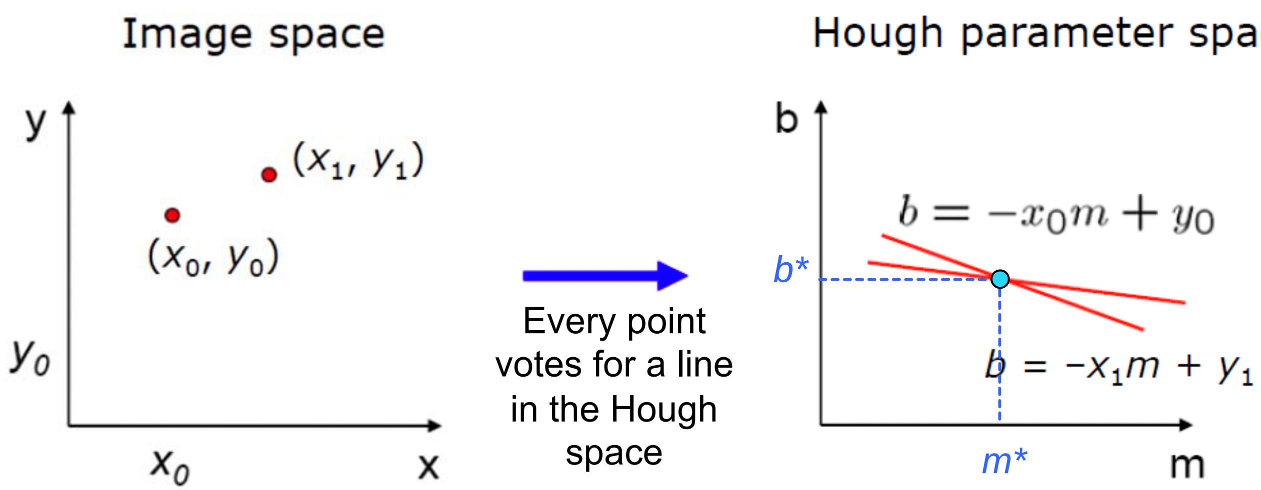


Line Extraction – Hough-Transform

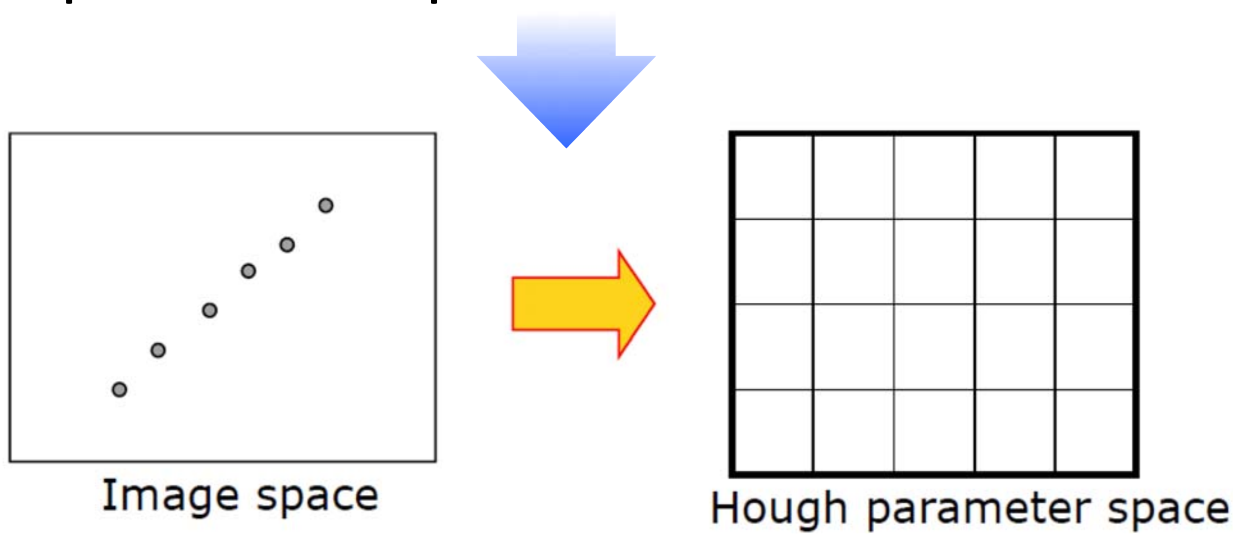
- Where is the line that contains both (x_0, y_0) and (x_1, y_1)
  - It is the intersection of the lines
    $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$



Line Extraction – Hough-Transform



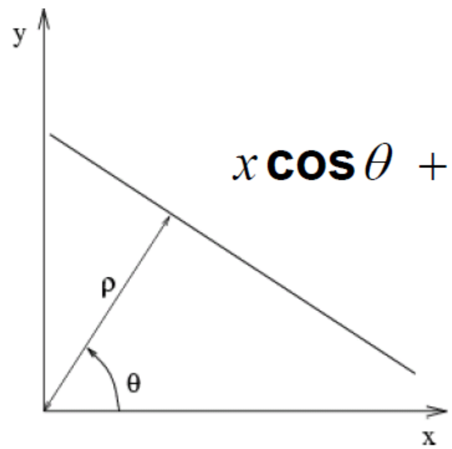
- Each point in image space votes for line-parameters in Hough parameter space



Line Extraction – Hough-Transform

- Problems with the *(m,b)* space:
  - Unbounded parameter domain
  - How to represent vertical lines?

- Alternative: polar representation



$$x\cos\theta + y\sin\theta = \rho$$

Each point in image space will map to a **sinusoid** in the (ρ, θ) parameter space

Line Extraction – Hough-Transform

1. Initialize accumulator H to all zeros
2. for each edge point (x,y) in the image

for all θ in [0,180]

Compute $\rho = x\ cos\theta + y\ sin\theta$

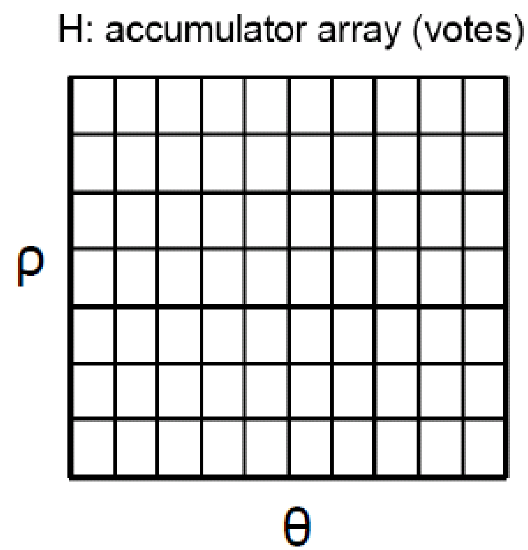
$H(\theta, \rho) = H(\theta, \rho) + 1$

end

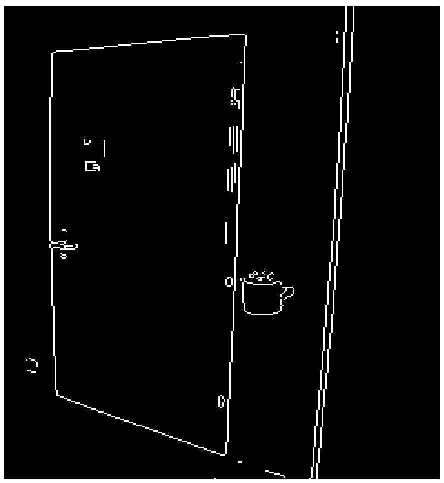
end

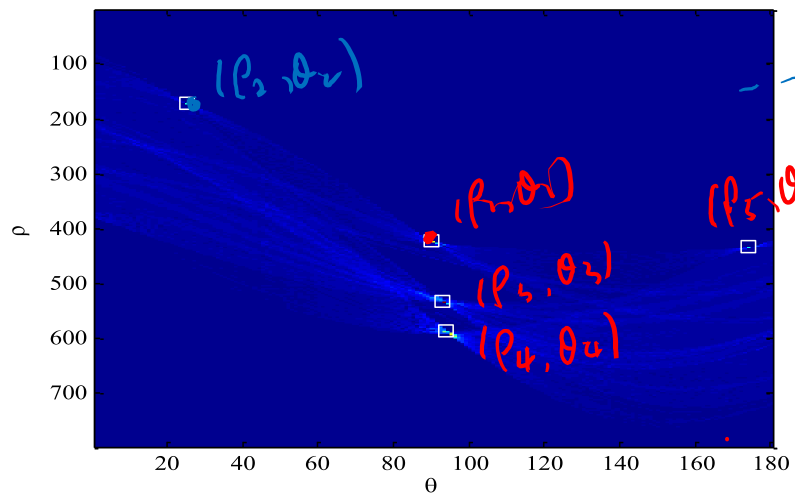
3. Find the values of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
4. The detected line in the image is given by:

$$\rho = x\ cos\theta + y\ sin\theta$$

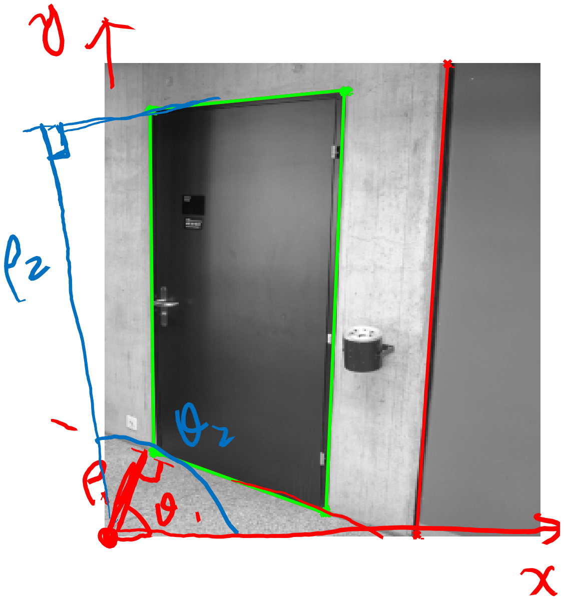


Line Extraction – Hough-Transform: Examples

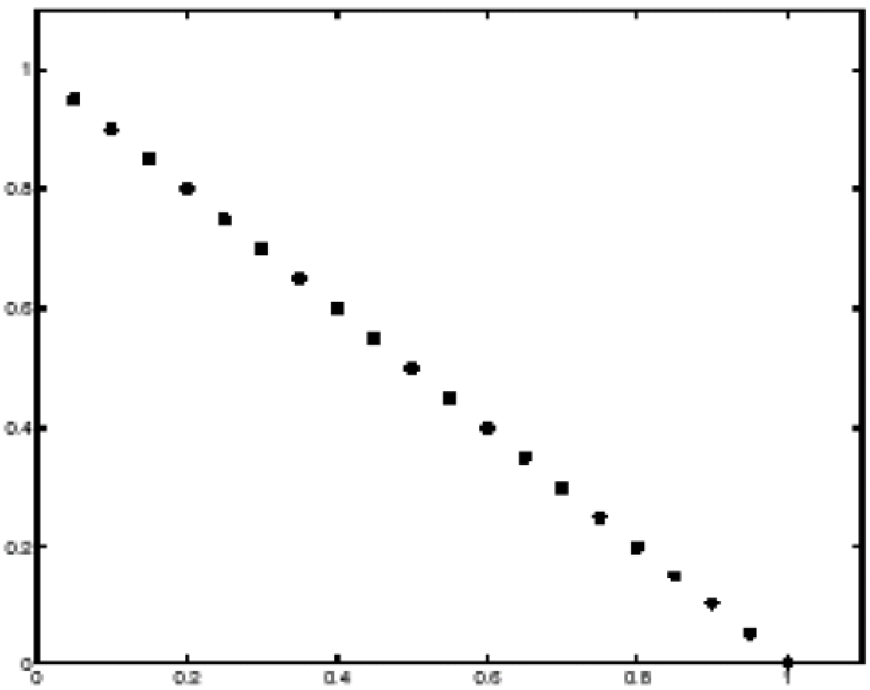




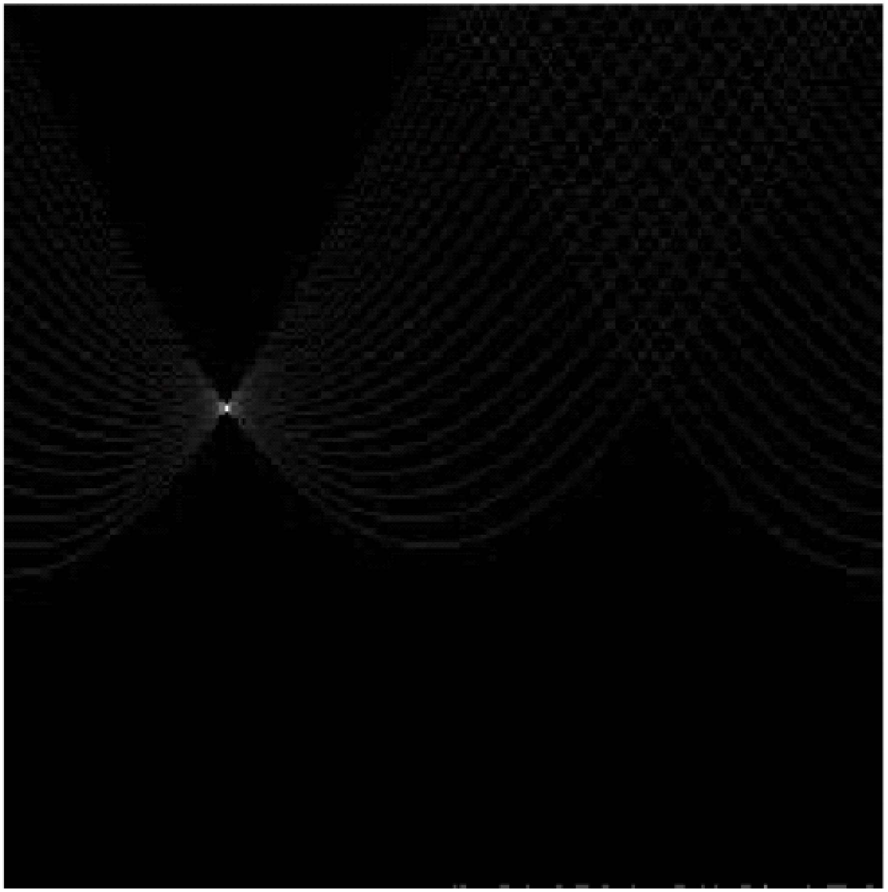
Hough Transform



Line Extraction – Hough-Transform: Examples



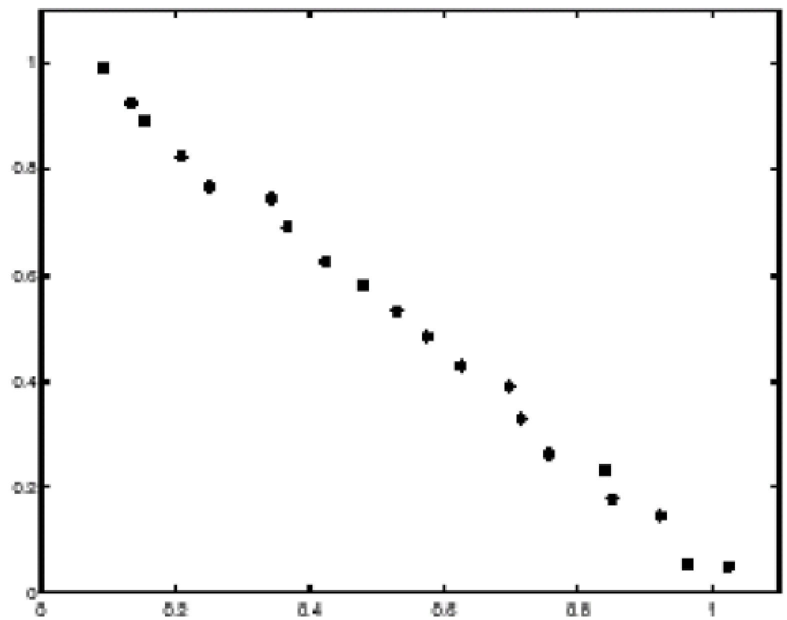
features



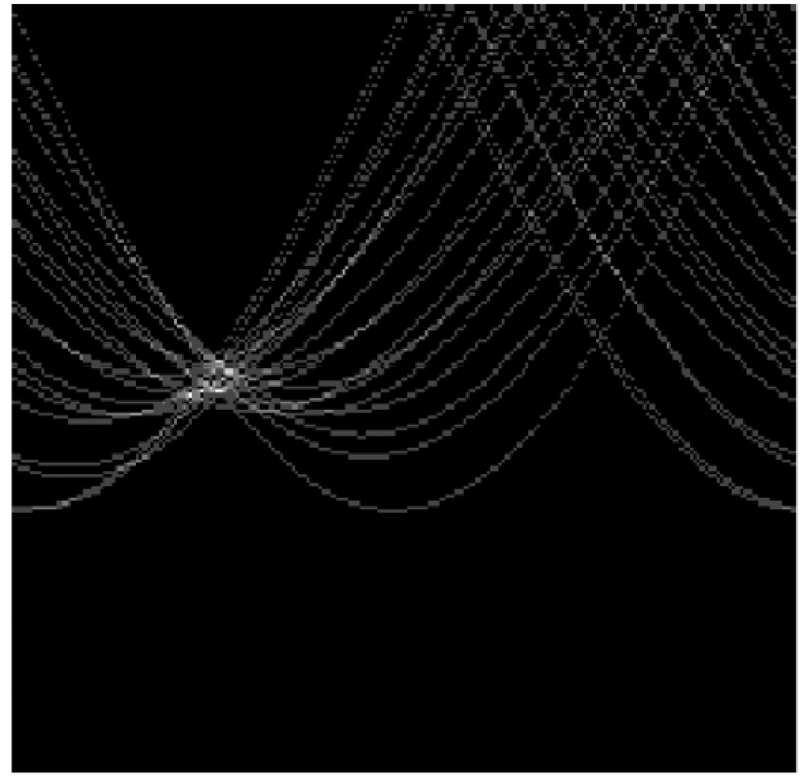
votes

Line Extraction – Hough-Transform: Examples

Effect of Noise:



features



votes

- Peak gets fuzzy and hard to locate

Line Extraction – Comparison

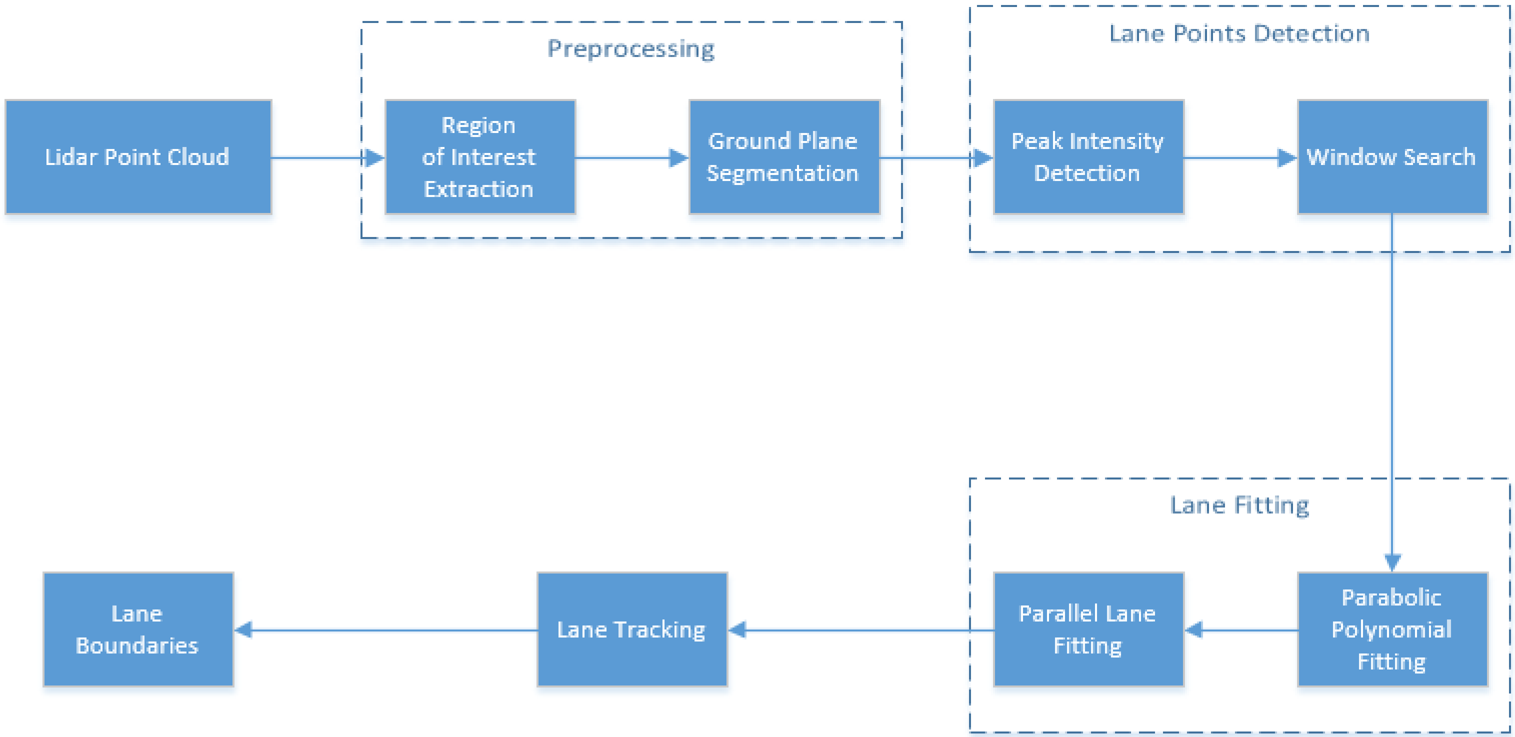
- Split-and-merge, Incremental and Line-Regression: **fastest** – best applied on **laser scans**
- Deterministic & make use of the **sequential ordering** of raw scan points (: points captured according to the rotation direction of the laser beam)
- If applied **on randomly captured points** only last 3 algorithms would segment all lines.
- RANSAC, Hough-Transform and EM produce greater precision --> more robust to outliers

	Complexity	Speed (Hz)	False positives	Precision
Split-and-Merge	$N \log N$	1500	10%	+++
Incremental	$S\,N$	600	6%	+++
Line-Regression	$N\,N_f$	400	10%	+++
RANSAC	$S\,N\,k$	30	30%	++++
Hough-Transform	$S\,N\,N_C + S\,N_R\,N_C$	10	30%	++++
Expectation Maximization	$S\,N_1\,N_2\,N$	1	50%	++++

N : no. points considered
N_f : no. points in window
S : no. line-segments to be found
k : no. iterations
N_C, N_R : no columns, rows of the accumulator array

Comparison by [Nguyen et al. IROS 2005]

Example:Lane Detection in 3D Lidar Point Cloud



Lane Detection in 3D Lidar Point Cloud

- The advantages of using Lidar data for lane detection are:
  - Lidar point clouds give a better 3D representation of the road surface than image data, thus reducing the required calibration parameters to find the bird's-eye view
  - Lidar is more robust against adverse climatic conditions than image-based detection
  - Lidar data has a centimeter level of accuracy, leading to accurate lane localization

Lane Detection in 3D Lidar Point Cloud

- Lane detection in Lidar involves detection of the immediate left and right lanes, also known as ego vehicle lanes, w.r.t. the Lidar sensor. It involves the following steps:
  - Region of interest extraction
  - Ground plane segmentation
  - Peak intensity detection
  - Lane detection using window search
  - Parabolic polynomial fitting
  - Parallel lane fitting
  - Lane tracking

Examples:

Lidar Toolbox:

- https://www.mathworks.com/help/lidar/
- What is Lidar:
  - https://www.mathworks.com/discovery/lidar.html

Examples:

- https://www.mathworks.com/help/lidar/examples.html
- Line extraction:
  - https://www.mathworks.com/help/images/ref/houghlines.html
- Lane detection in 3D Lidar point cloud
  - https://www.mathworks.com/help/lidar/ug/lane-detection-in-3d-lidar-point-cloud.html
- Curb detection and tracking in 3D Lidar point cloud
  - https://www.mathworks.com/help/lidar/ug/curb-detection-in-lidar-point-cloud.html

More Examples

- Detect and track vehicles using Lidar data
  - https://www.mathworks.com/help/vision/ug/track-vehicles-using-lidar.html
- Ground plane and obstacle detection using Lidar
  - https://www.mathworks.com/help/driving/ug/ground-plane-and-obstacle-detection-using-lidar.html
- Build a map from Lidar data
  - https://www.mathworks.com/help/driving/ug/build-a-map-from-lidar-data.html

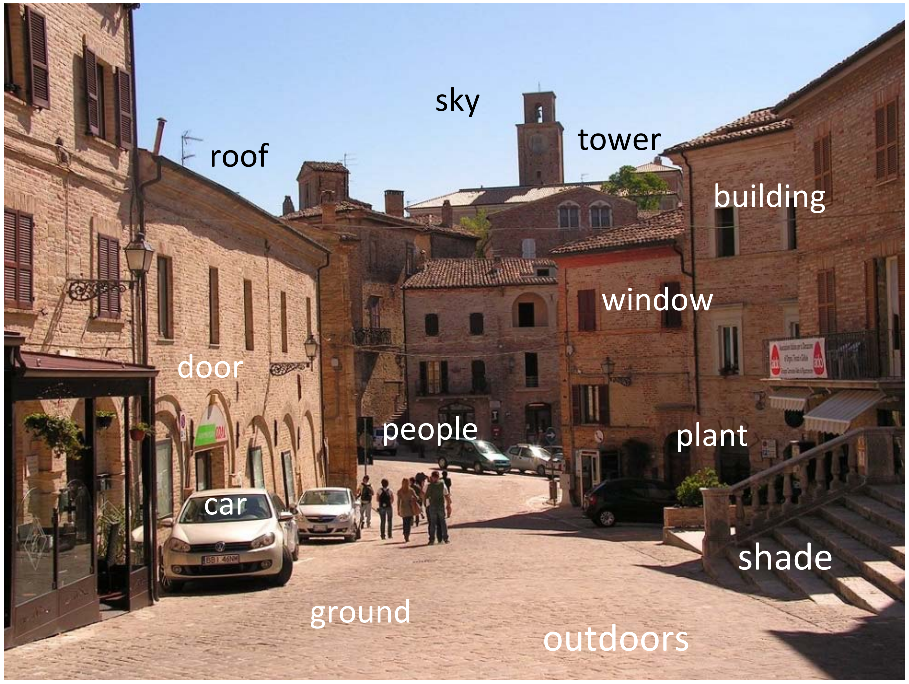
Computer Vision and Image Processing

Human Visual Capabilities

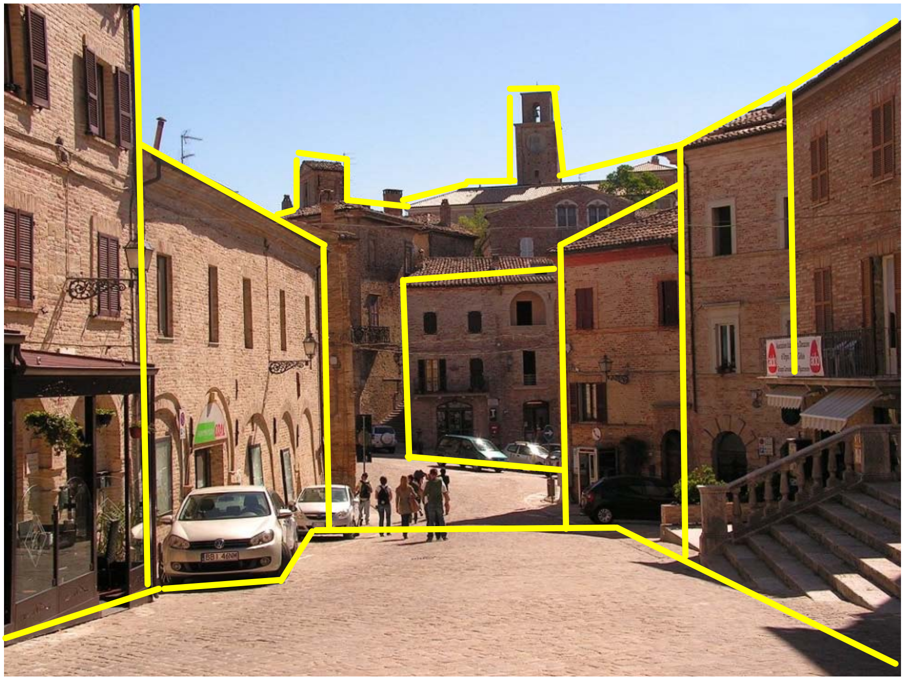
- Our visual system is very sophisticated
- Humans can interpret images successfully under a wide range of conditions – even in the presence of very limited cues

Computer Vision – What is it?

- Automatic extraction of "meaningful" information from images and videos
  - varies depending on the application



Semantic information


Geometric information

Computer Vision for Robotics

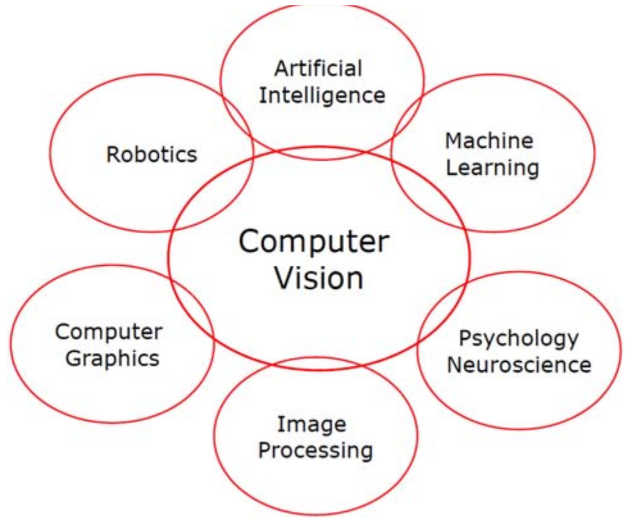
- Enormous descriptability of images → a lot of data to process (human vision involves 60 billion neurons!)
- Vision provides humans with a great deal of useful cues to explore the power of vision towards intelligent robots

Cameras:

- Vision is increasingly popular as a sensing modality:
  - descriptive
  - compactness, compatibility, ...
  - low cost
  - HW advances necessary to support the processing of images

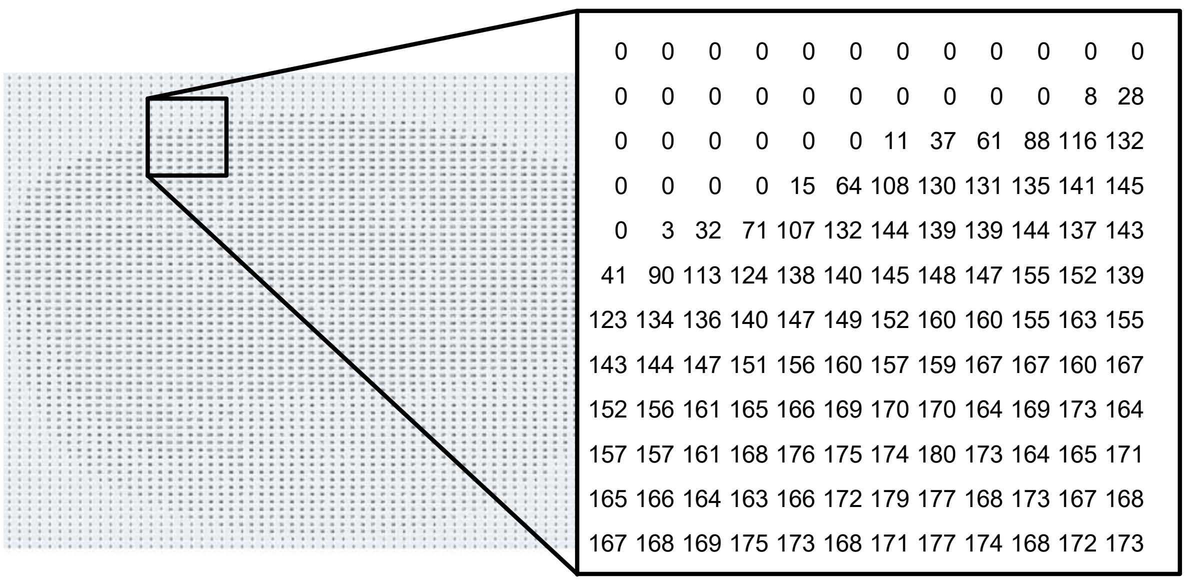
Computer Vision – Applications

- 3D reconstruction and modeling
- Recognition
- Motion capture
- Augmented reality:
- Video games and tele-operation
- Robot navigation and automotive
- Medical imaging



Computer Vision – Why is it hard?

- Achieving human-level visual perception is probably “AI-complete”



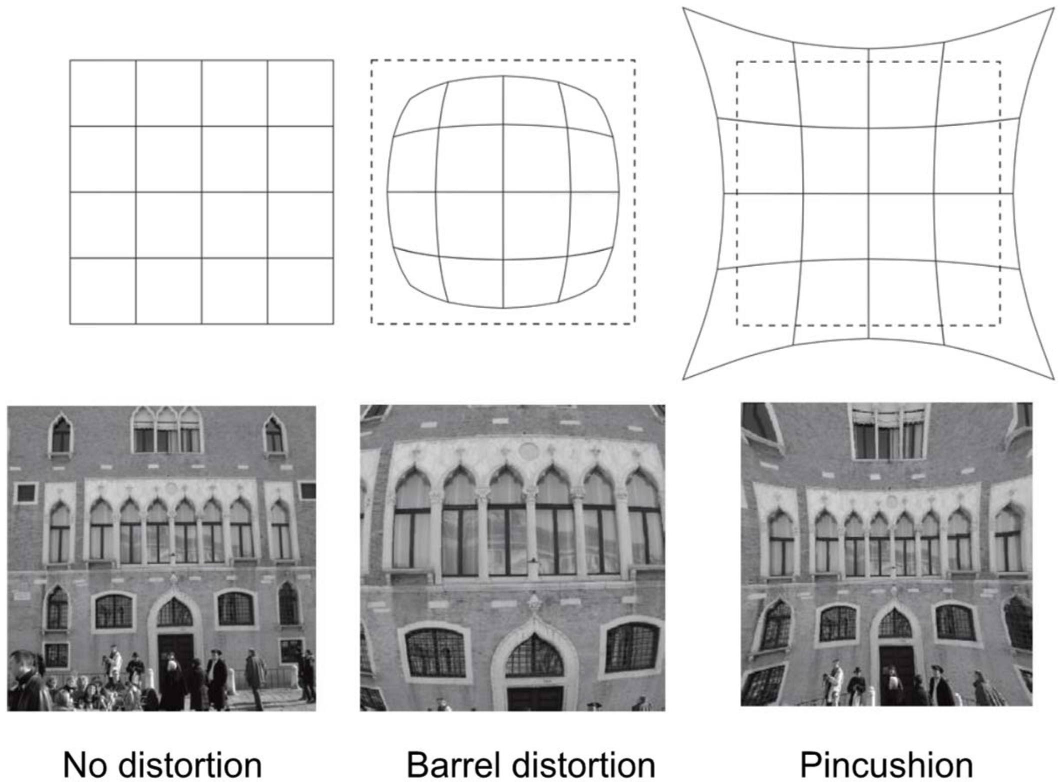
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	8	28
0	0	0	0	0	0	11	37	61	88	116	132
0	0	0	0	15	64	108	130	131	135	141	145
0	3	32	71	107	132	144	139	139	144	137	143
41	90	113	124	138	140	145	148	147	155	152	139
123	134	136	140	147	149	152	160	160	155	163	155
143	144	147	151	156	160	157	159	167	167	160	167
152	156	161	165	166	169	170	170	164	169	173	164
157	157	161	168	176	175	174	180	173	164	165	171
165	166	164	163	166	172	179	177	168	173	167	168
167	168	169	175	173	168	171	177	174	168	172	173

What a computer sees

What we see

Image Distortion

- The camera image is a projection of the three dimensional space into a two dimensional space,
- The projection process is affected by the characteristics of each camera



Camera Calibration

- Camera calibration: calculating the camera's unique parameters
- Camera calibration is necessary if you are measuring distance from images acquired with a stereo camera or processing images for object detection
  - Need to know the information of the camera: lens characteristics, the gap between the lens and the image sensor, and the twisted angle of the image sensor, etc.

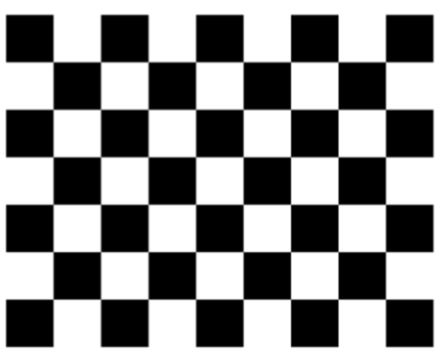


FIGURE 8-8 Chessboard for calibration (8 x 6)

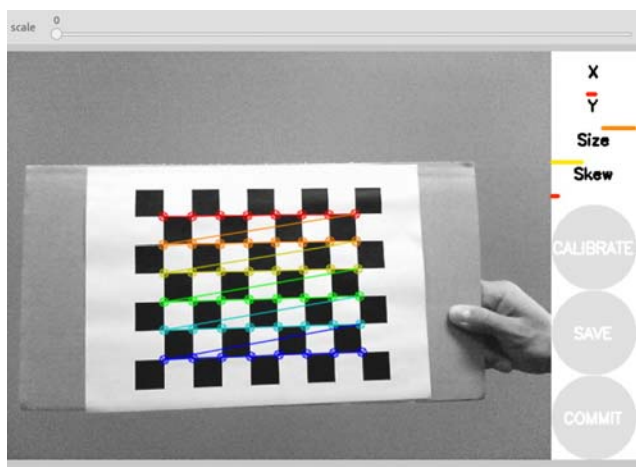


FIGURE 8- Calibration GUI initial state

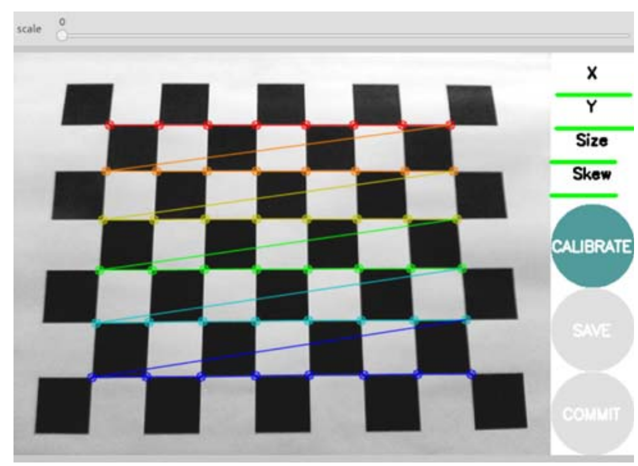
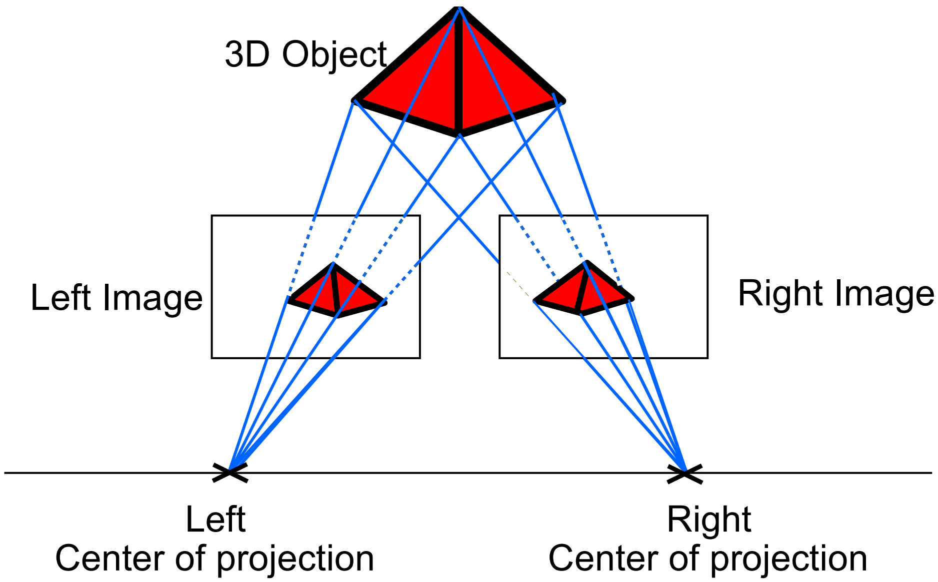


FIGURE 8-10 Calibration process using the calibration GUI

How do we measure distances with cameras?

- From a single image: we can only deduct the **ray** along which each image-point lies
- Stereo vision
  - using 2 cameras with known relative position T and orientation R, recover the 3D scene information



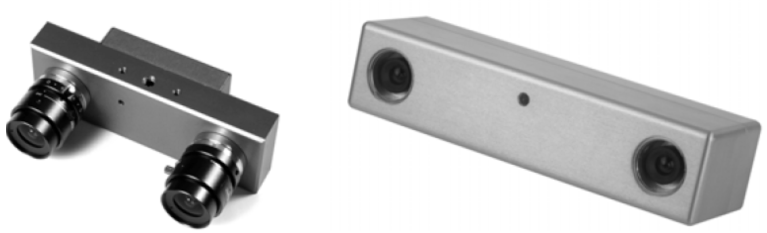
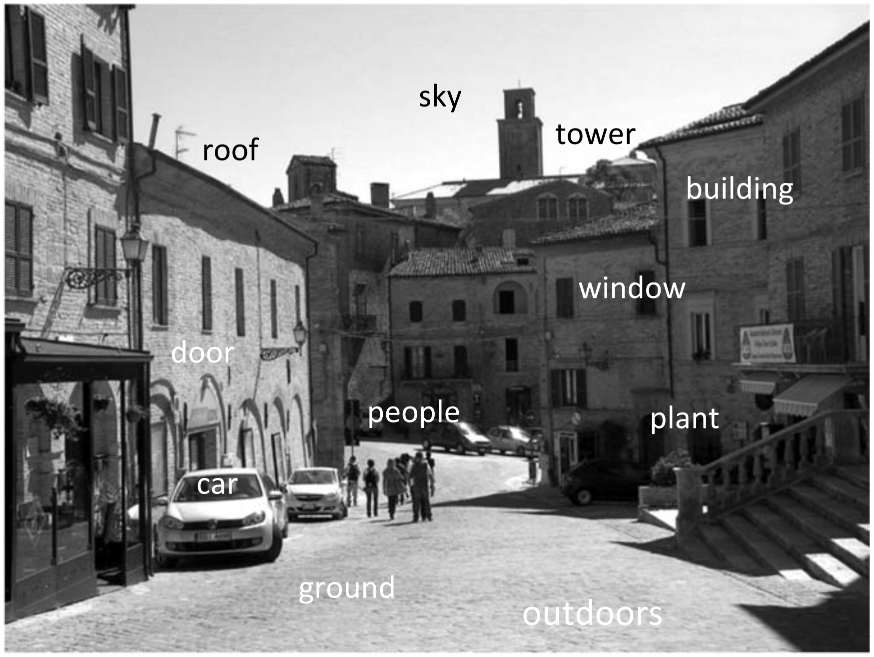


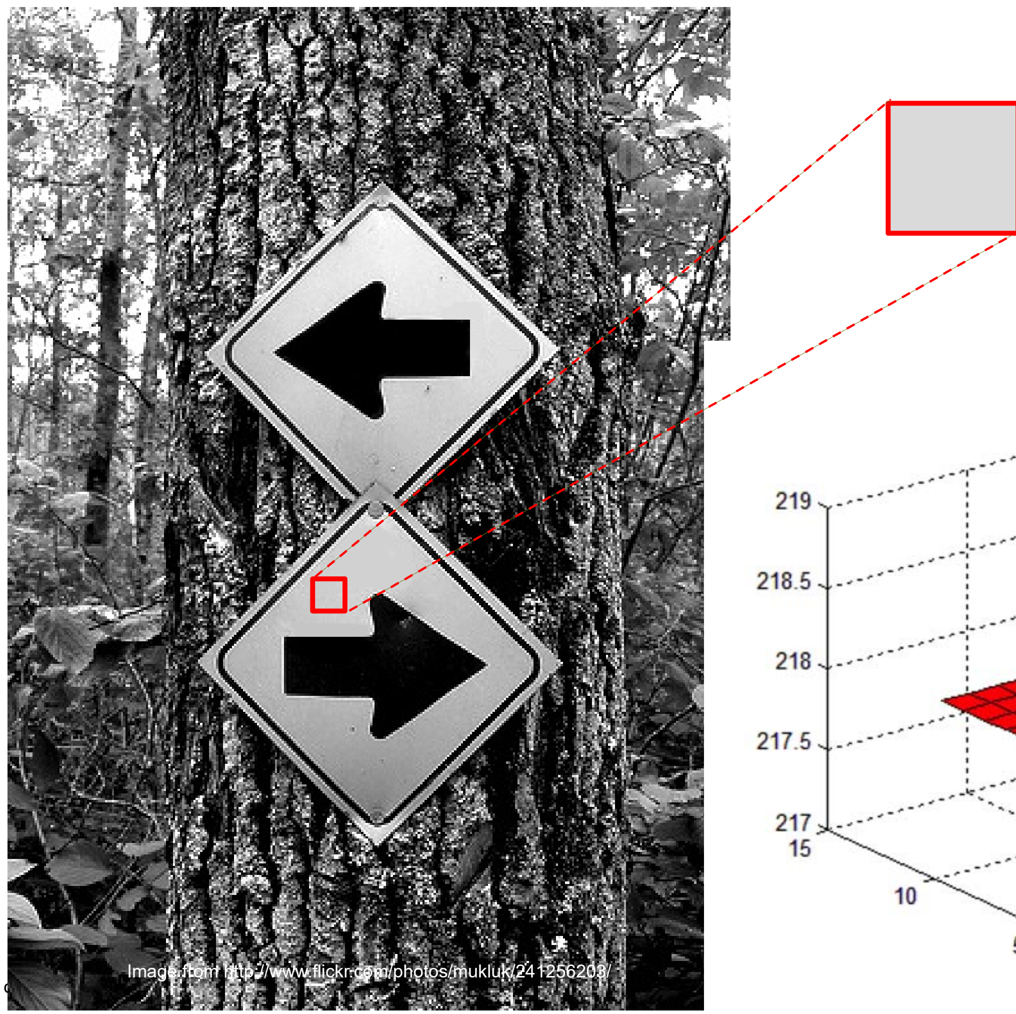
Image Processing

Image Intensities & Data Reduction

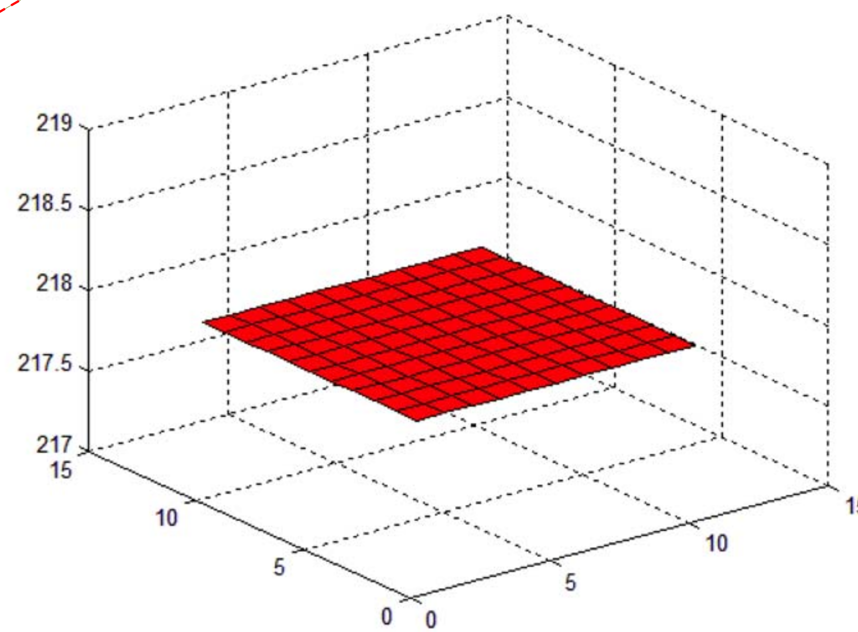
- Image capture a lot of information
- Monochrome image → matrix of intensity values
- Typical sizes:
  - 320 * 240 (QVGA)
  - 640 * 480 (VGA)
  - 1280 * 720 (HD)
- Intensity sampled to 256 grey levels – 8bits



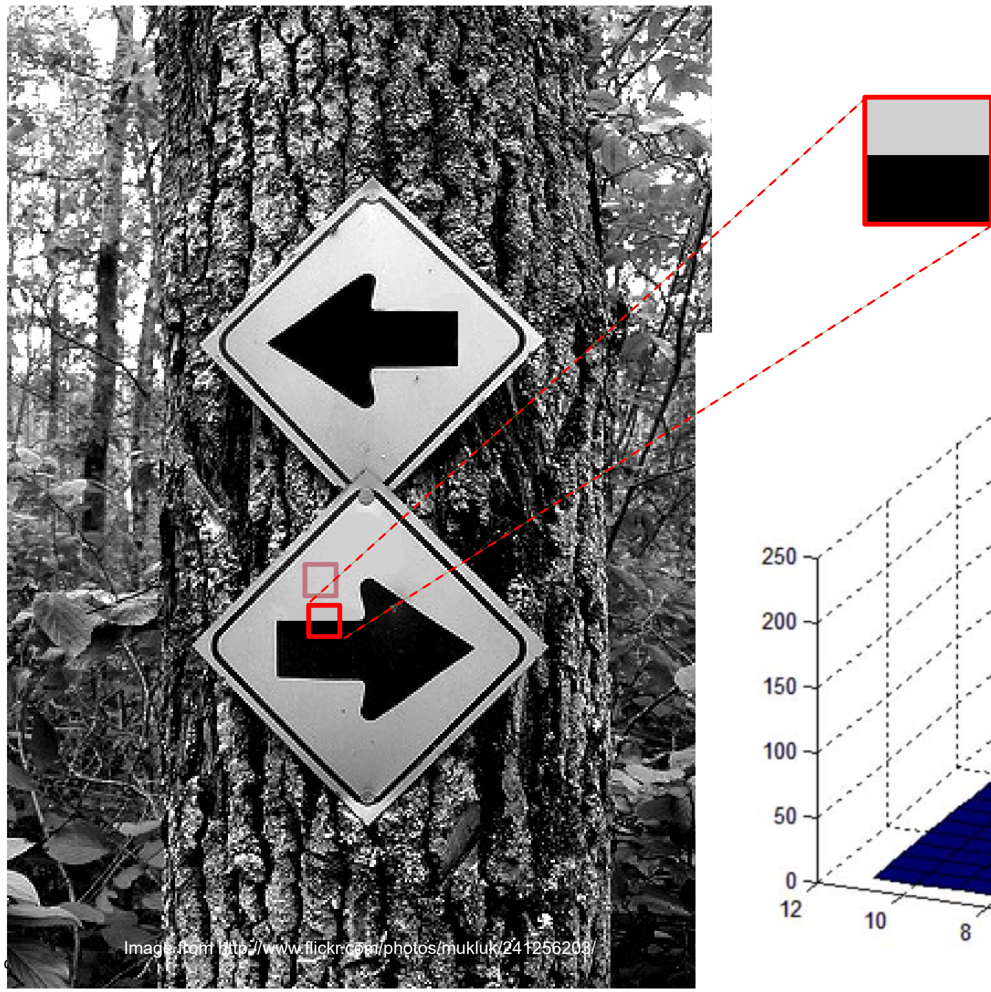
What is Useful, What is Redundant?



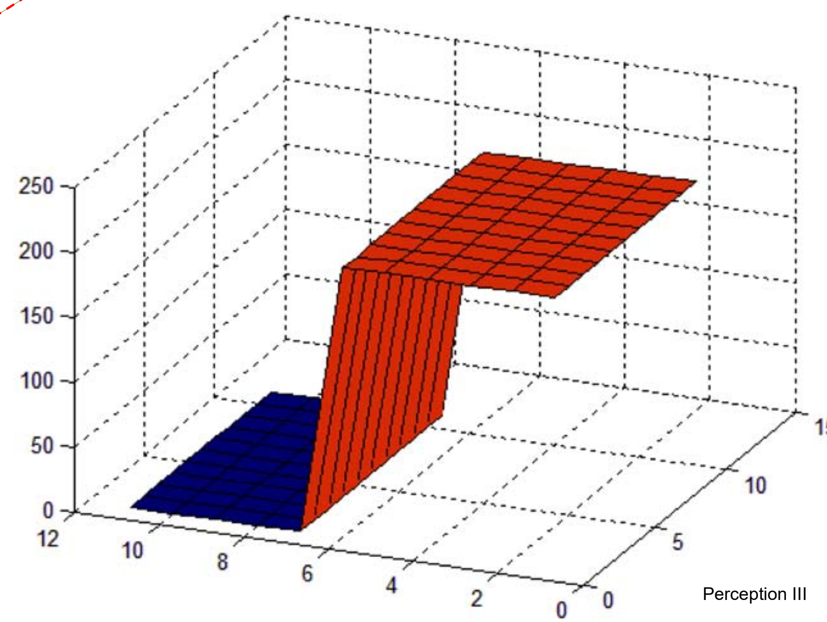
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218



What is Useful, What is Redundant?



208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	207	208	208	208	208	208	208	208	208	208
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	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	0	0	0	0	0



What is Useful, What is Redundant?

229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	230	229	229	229	229	229
5	17	31	7	1	0	229	229	229	229	229
0	0	1	0	0	0	229	229	229	229	229
0	0	0	0	0	0	229	229	229	229	229
0	0	0	0	1	4	229	229	229	229	229
0	0	0	0	0	11	229	229	229	229	229
0	0	0	0	0	5	229	229	229	229	229

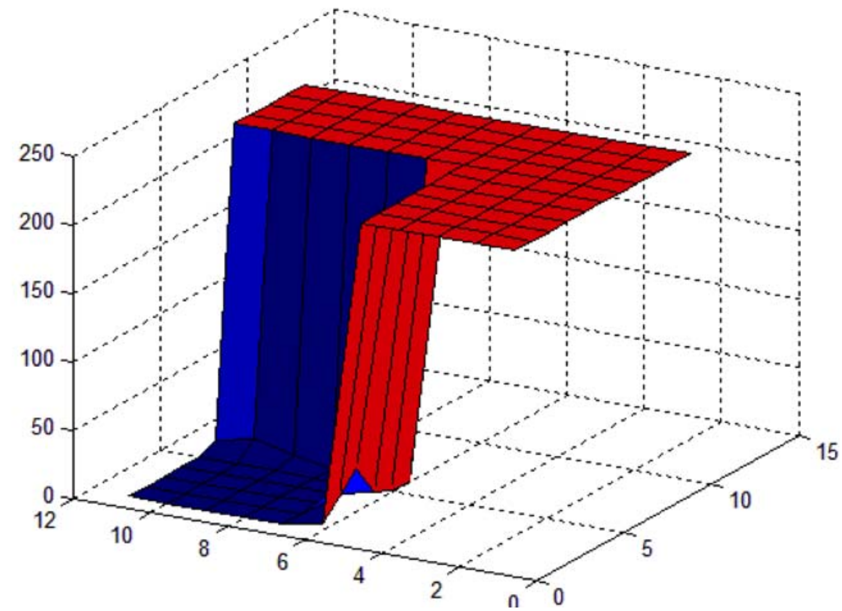


Image Filtering

- **filtering:** accept / reject certain components
- example: a low-pass filter allows low frequencies a blurring (smoothing) effect on an image – used to reduce image noise
- Smoothing can be achieved not only with **frequency filters**, but also with **spatial filters**.

Low-pass filtering:
retains low-frequency components (smoothing)

High-pass filtering:
retains high-frequency components (edge detection)

Image Filtering – Spatial Filters

- S_{xy}: neighborhood of pixels around the point *(x,y)* in an image *I*
- Spatial filtering operates on S_{xy} to generate a new value for the corresponding pixel at output image *J*

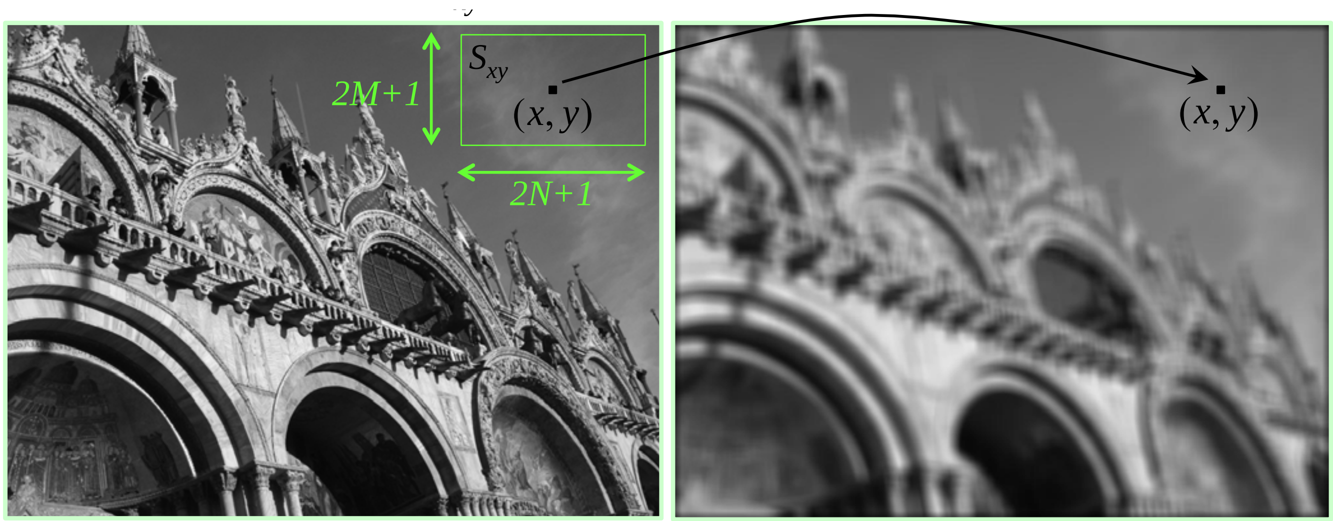


Image I Filtered Image $J = F(I)$

For example, an averaging filter is: $J(x, y) = \dfrac{\sum_{u,v \in S_{xy}} I(u,v)}{(2M+1)(2N+1)}$

Image Filtering – Linear, shift-invariant filters

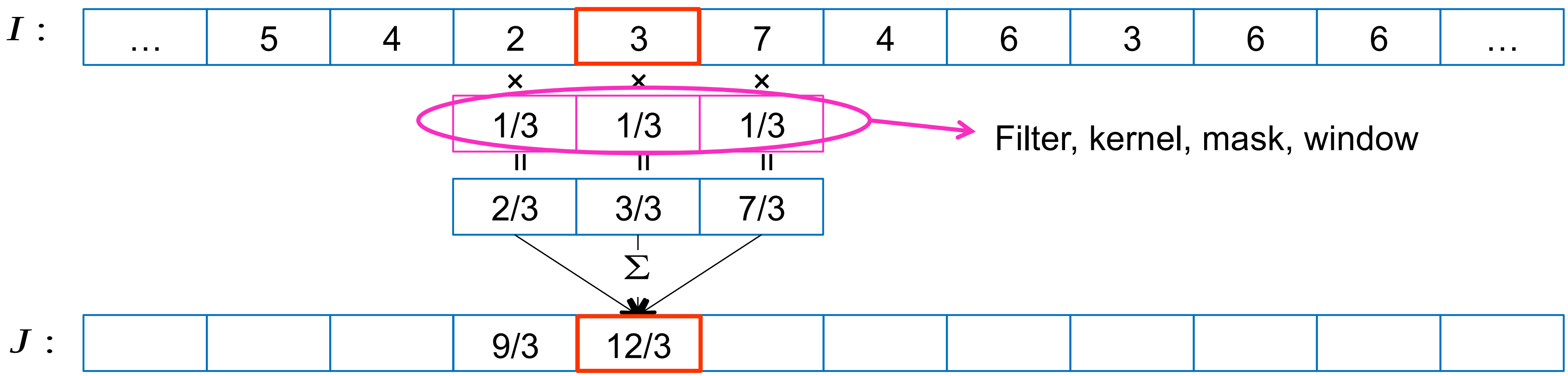
- Linear: every pixel is replaced by a linear combination of its neighbors

- Shift-invariant: the same operation is performed on every point on the image

- Why filter?
  - noise reduction, image enhancement, feature extraction, …

Image Filtering – Correlation

- An averaging filter in 1D



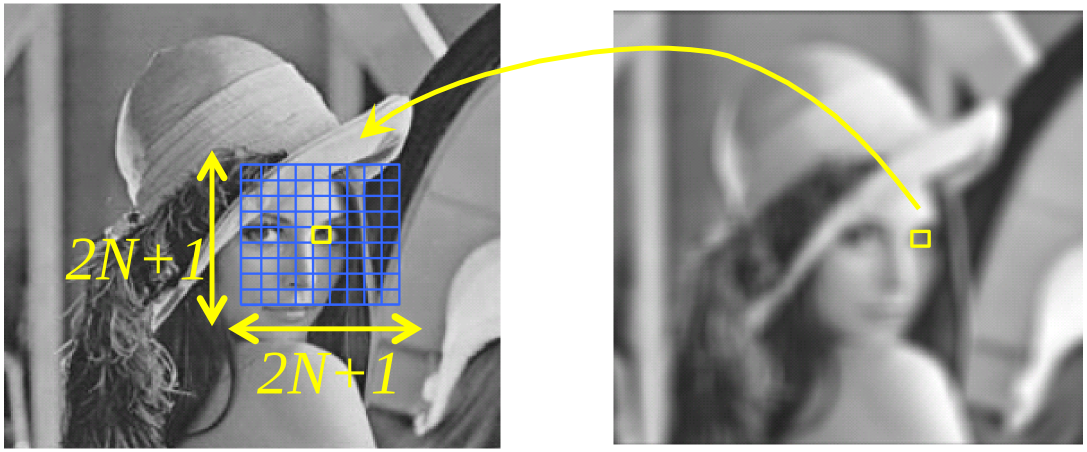
- Formally, Correlation is $J(x) = F \cdot I(x) = \sum_{i \in [-N,N]} F(i) I(x+i)$

- In this smoothing example $F(i) = \begin{cases} 1/3, i \in [-1,1] \\ 0, i \notin [-1,1] \end{cases}$

Image Filtering – Correlation in 2D

- Example: constant averaging filter

$$F = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$



- If $size(F) = (2N+1)^2$, i.e., a square filter
  - # of multiplications per pixel = $(2N+1)^2$
  - # of additions per pixel = $(2N+1)^2 - 1$

Image Filtering – Matching Using Correlation

- Find locations in an image that are similar to a template
- Filter = template

3	8	3

→ test it against all image locations

I :

3	2	4	1	3	8	4	0	3	8	7	7

J :

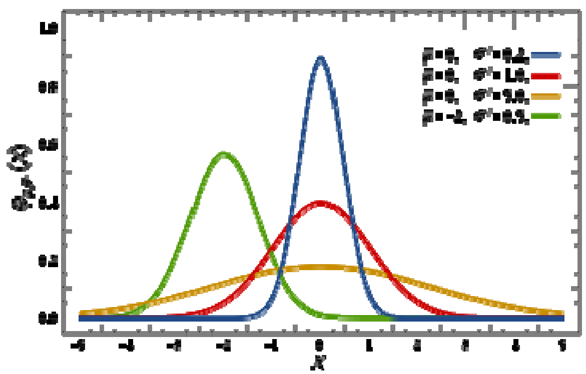
26	37	21	50	54	1	50	65	59	16	42	17

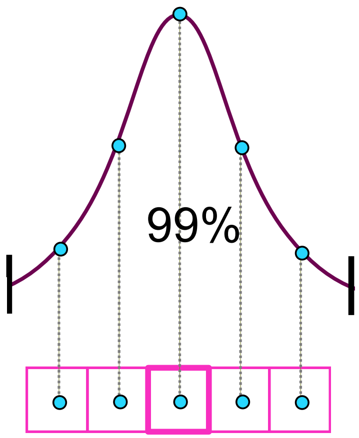
- Similarity measure: Sum of Squared Differences (SSD) minimizes

$$\sum_{i=-N}^{N} \left(F(i) - I(x+i)\right)^2$$

Image Filtering – Gaussian Filter

- Common practice for image smoothing: use a Gaussian





$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-u)^2}{2\sigma^2}}$$

$\mu = 0$

σ: controls the amount of smoothing

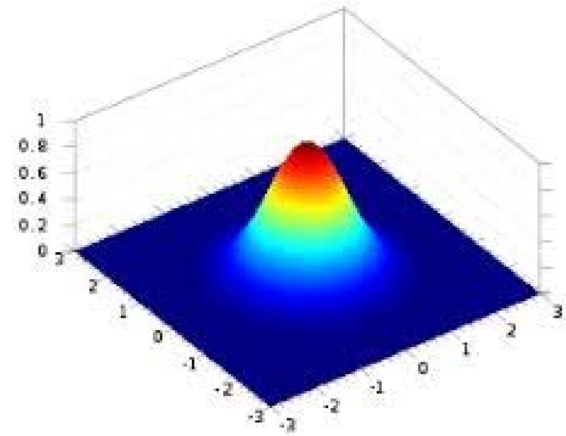
Normalize filter so that values always add up to 1

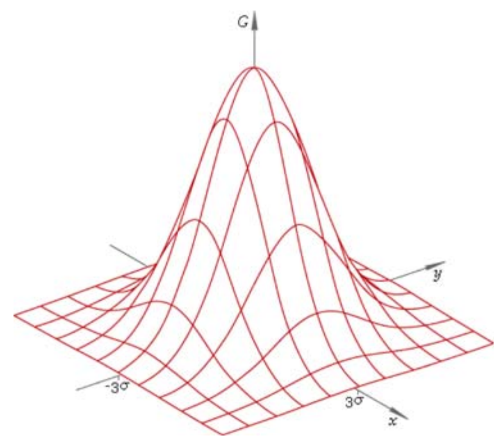
- Near-by pixels have a bigger influence on the averaged value rather than more distant ones

Image Filtering – 2D Gaussian Smoothing

- A general, 2D Gaussian

$$G(x,y) = \frac{1}{2\pi|S|^{1/2}} e^{-\frac{1}{2}\binom{x}{y} S^{-1}(x,y)}$$

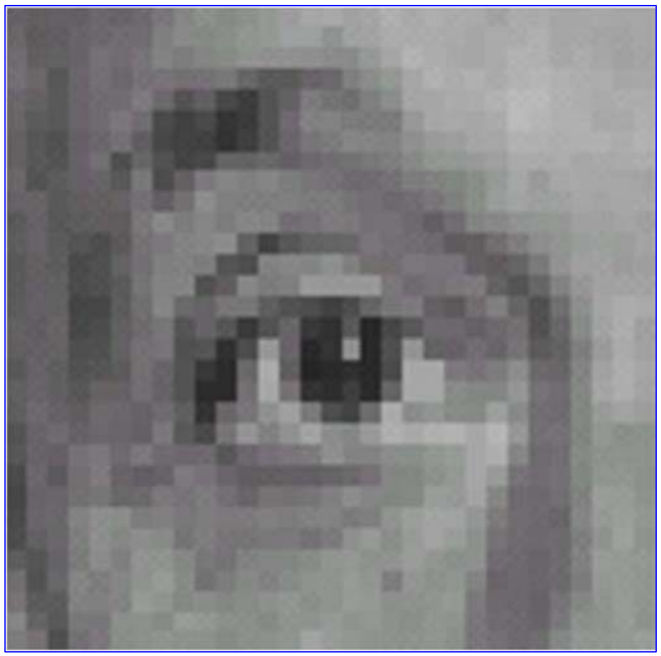




- We usually want to smooth by the same amount in both *x* and *y* directions

$$S = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Image Filtering – Examples


original image

*

0	0	0
0	1	0
0	0	0

=

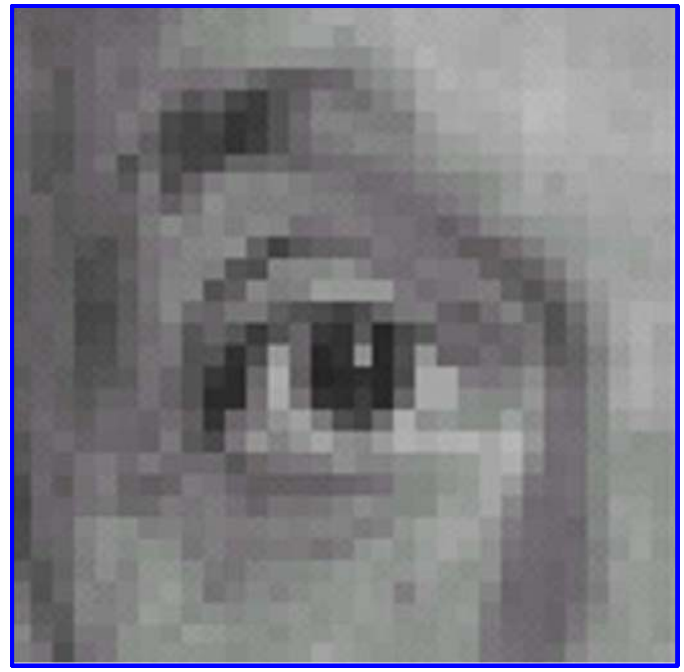
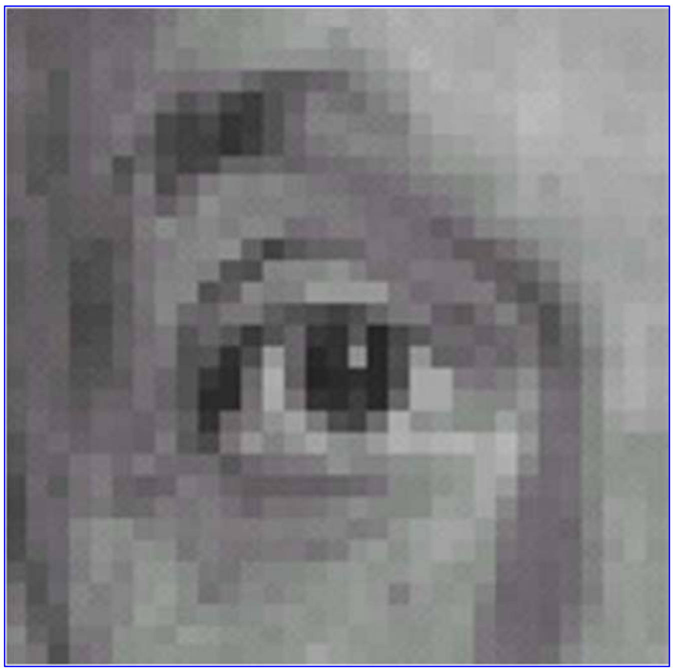
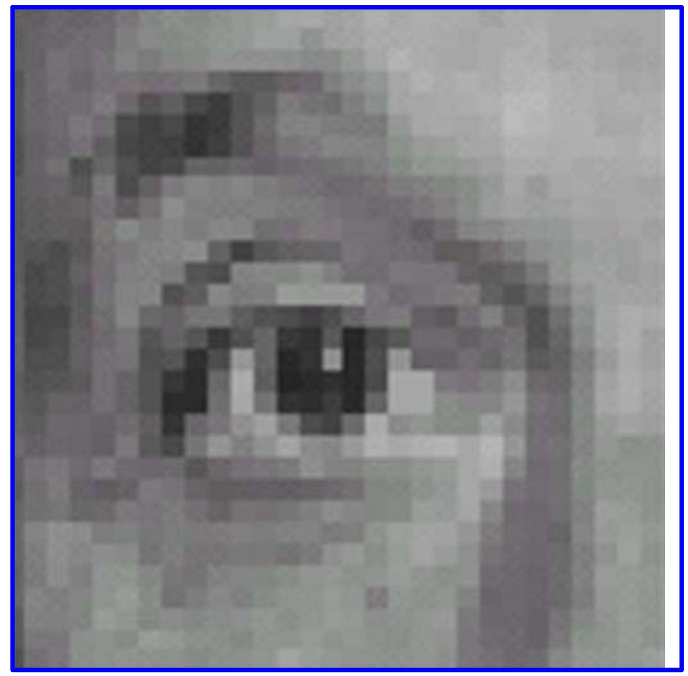


Image Filtering – Examples



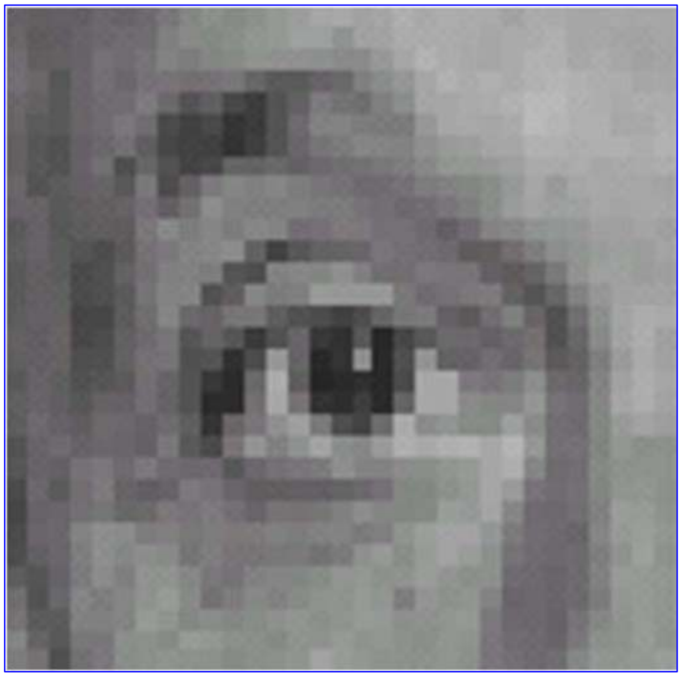
original image

$$
* \quad \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad =
$$



filtered (shifted left by 1 pixel)

Image Filtering – Examples

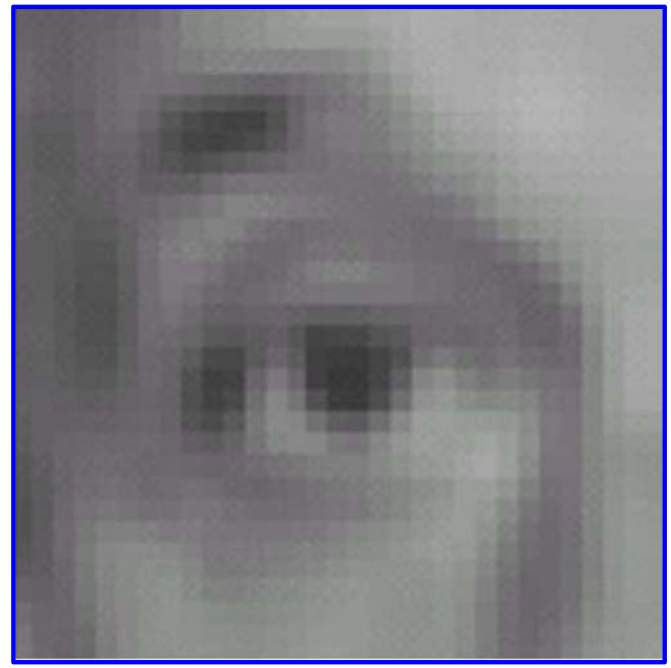


original image

*

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

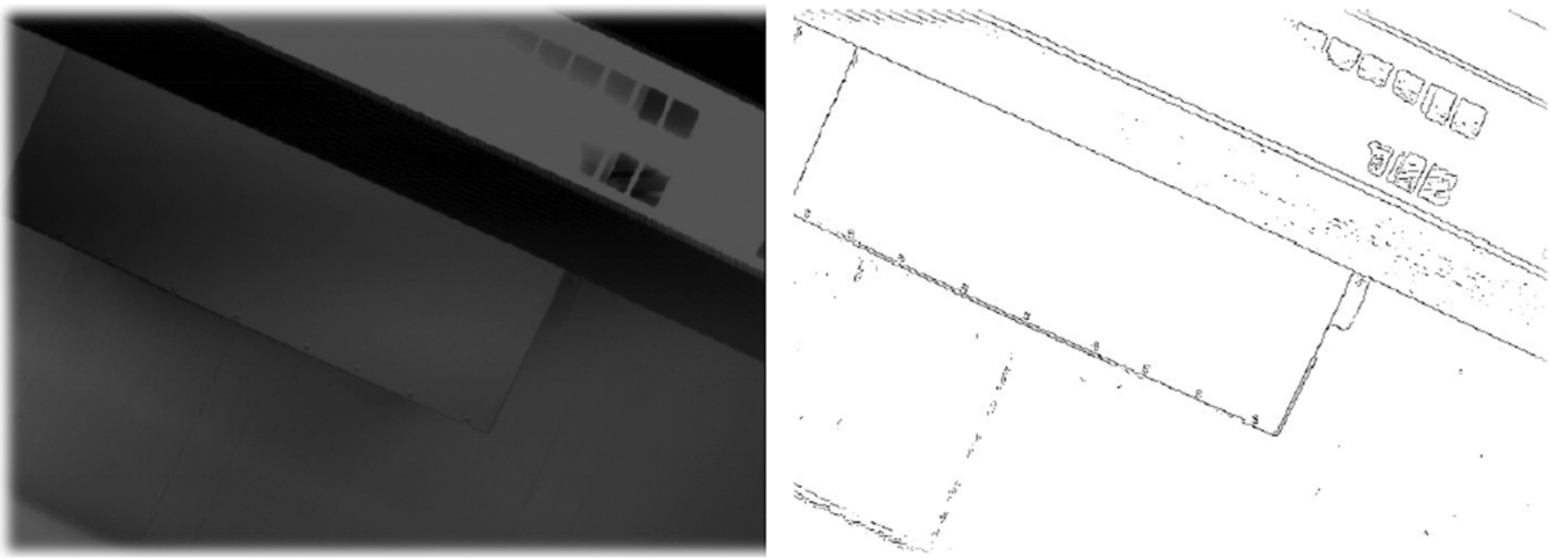
=



filtered (blurred with a box filter)

Image Filtering – Edge Detection

- Ultimate goal of edge detection: an idealized line drawing
- Edge contours in the image correspond to important scene contours



- Edges correspond to sharp changes of intensity
- How to detect an edge?
  - Big intensity change → magnitude of derivative is large

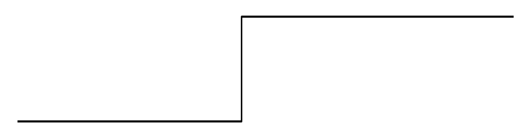


Image Filtering – Edge Detection

- Examples of edge detection filters
  - Prewitt: $F_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $F_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
  - Sobel: $F_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $F_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
  - Roberts: $F_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ $F_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Image Filtering – Edge Detection

I : original image (Lena)

- Lidar-camera filtering:
  - https://www.mathworks.com/help/lidar/ug/lidar-camera-calibration.html
- Camera calibration using AprilTag markers
  - https://www.mathworks.com/help/vision/ug/camera-calibration-using-apriltag-markers.html

- Thank you!