The First Intelligent Car?

http://automoblog.net

Intelligent Cars:

Lane Detection

In Driver Assistance Systems

Nikhil Churamani

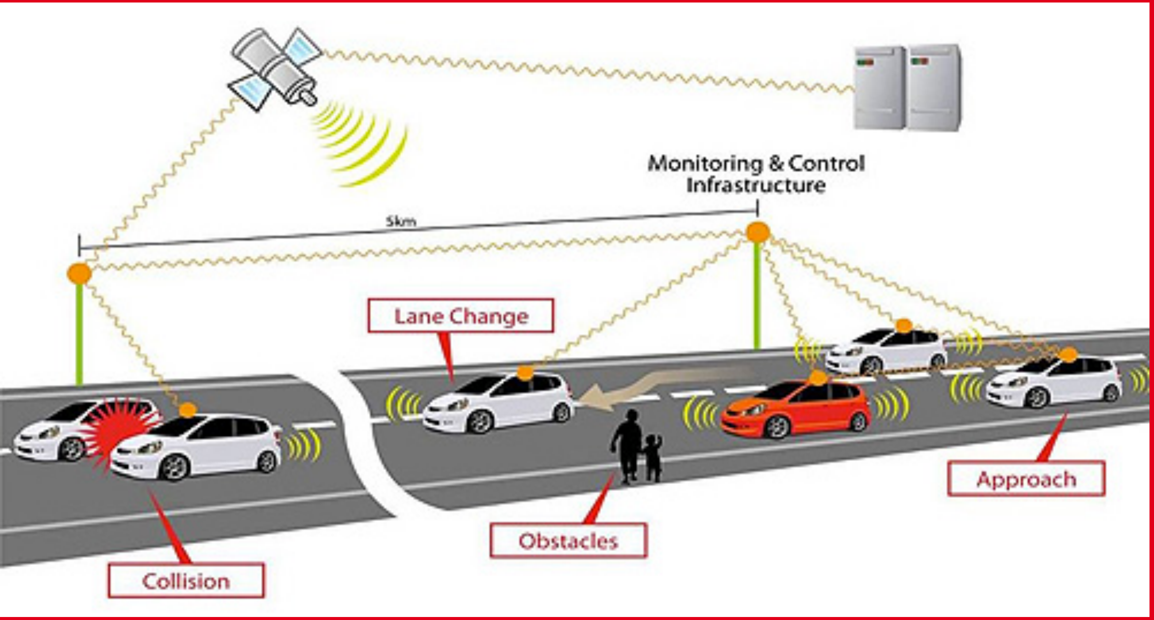
64.425 Integrated Seminar: Intelligent Robotics
WiSe – 2015

Outline

- Intelligent Cars : Motivation
- Role of Driver Assistance Systems
- Lane Detection System
  - Canny Edge Detection
  - Hough Transforms
  - Monte Carlo Localization
- Quality Evaluation of the system
- Bibliography

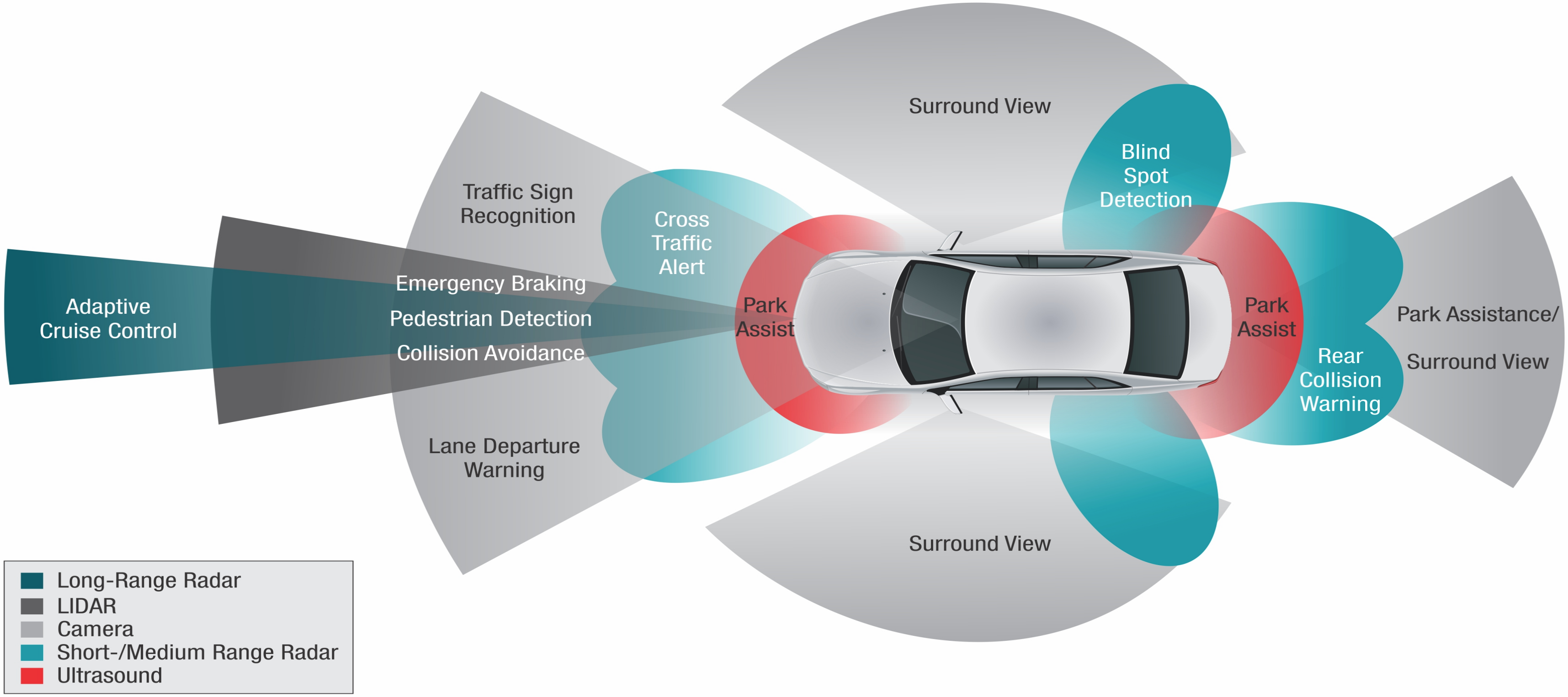
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Intelligent Cars



http://goauto.com

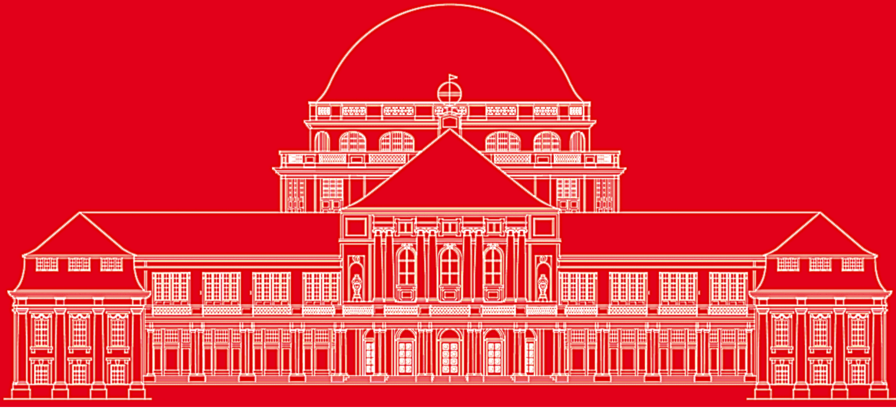
Why Intelligent Cars?



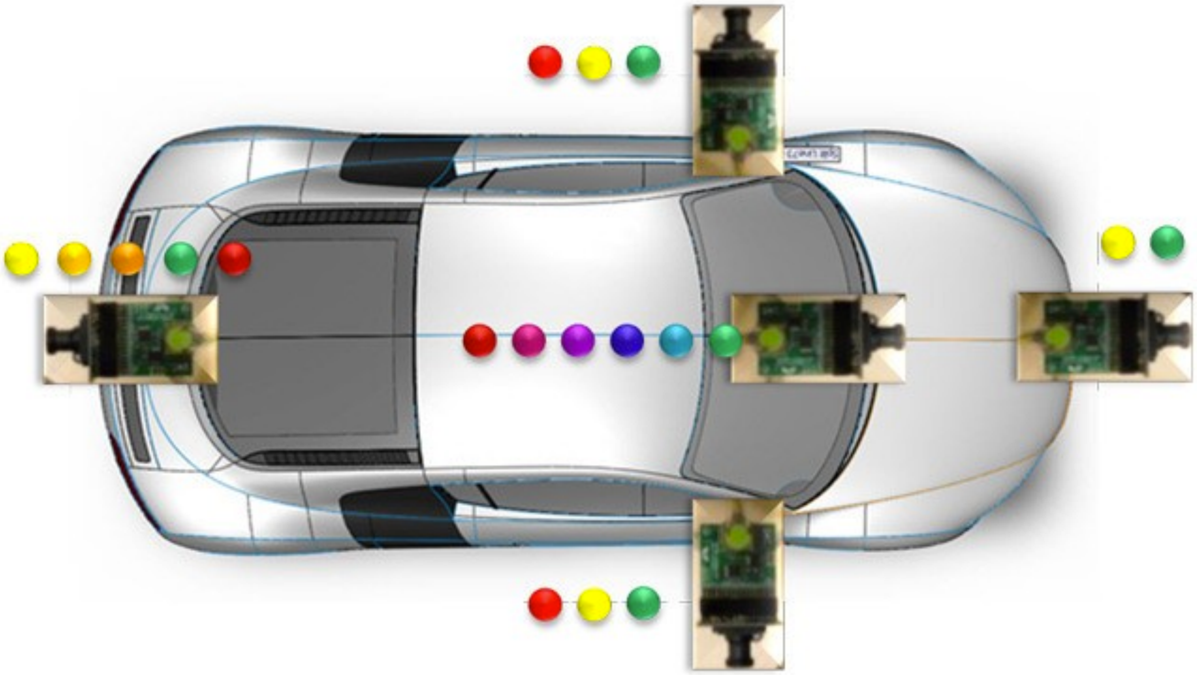
http://ti.com

http://dribble.com

Driver Assistance Systems



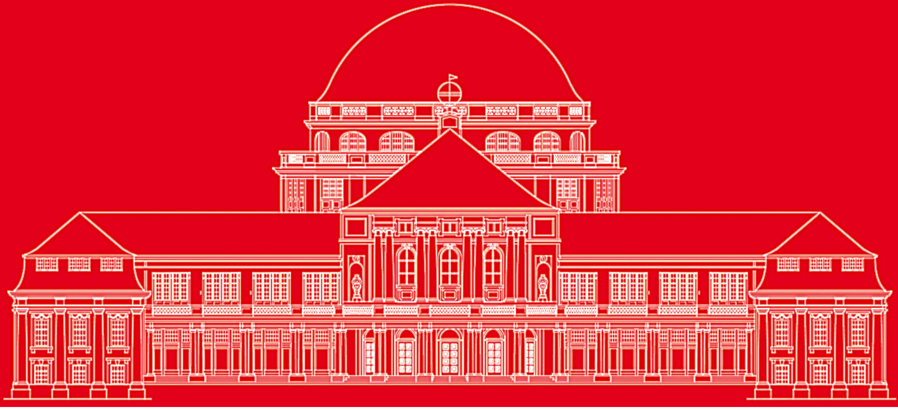
Driver Assistance System: Components



- ● 3D Surround View
- ● Rear View Camera
- ● Rear Cross Traffic
- ● Blind Spot Detection
- ● Lane Departure Warning
- ● Intelligent Headlamp Control
- ● Traffic Sign Recognition
- ● Forward Collision Warning
- ● Intelligent Speed Control
- ● Pedestrian Detection

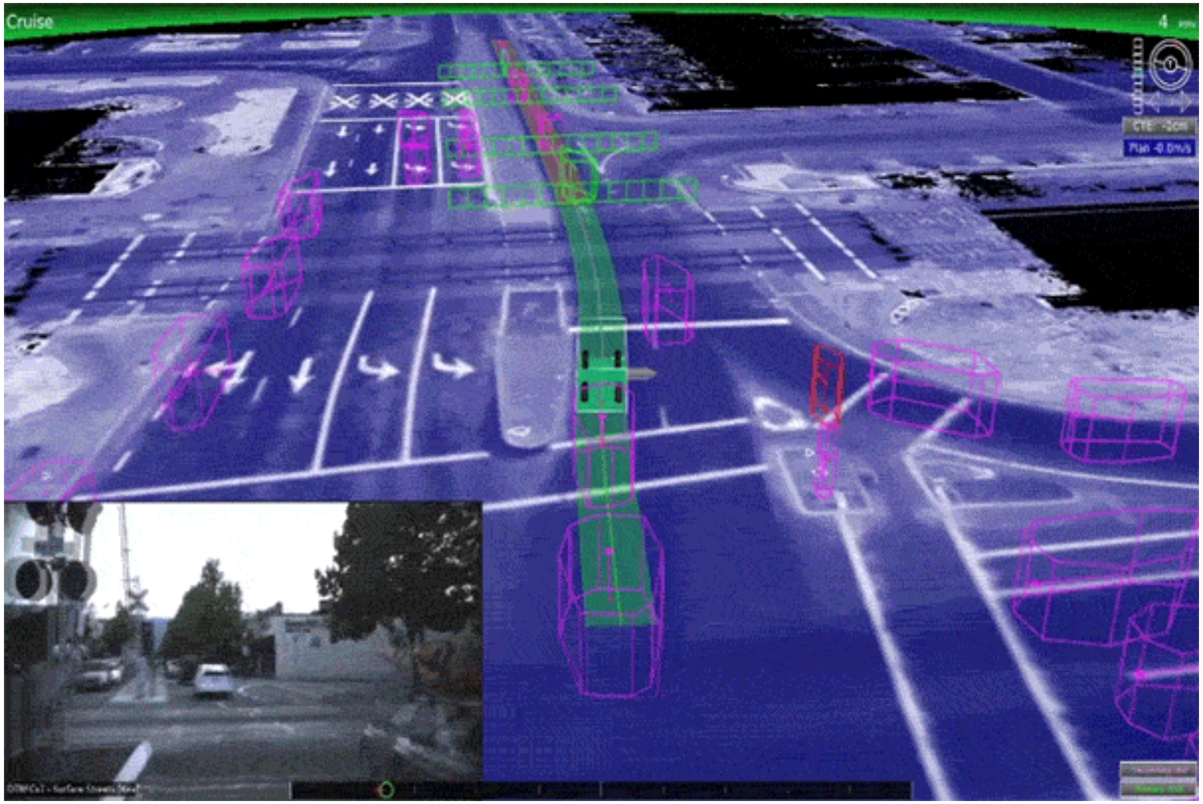
http://xilinx.xom

Lane Detection System



Mohamed Aly, Computational Vision Lab Electrical Engineering California Institute of Technology. ***Real time Detection of Lane Markers in Urban Streets.***

Lane Detection System



http://extremetech.com

Lane Detection System

Detection of Edges Points

- Canny Edge Detection

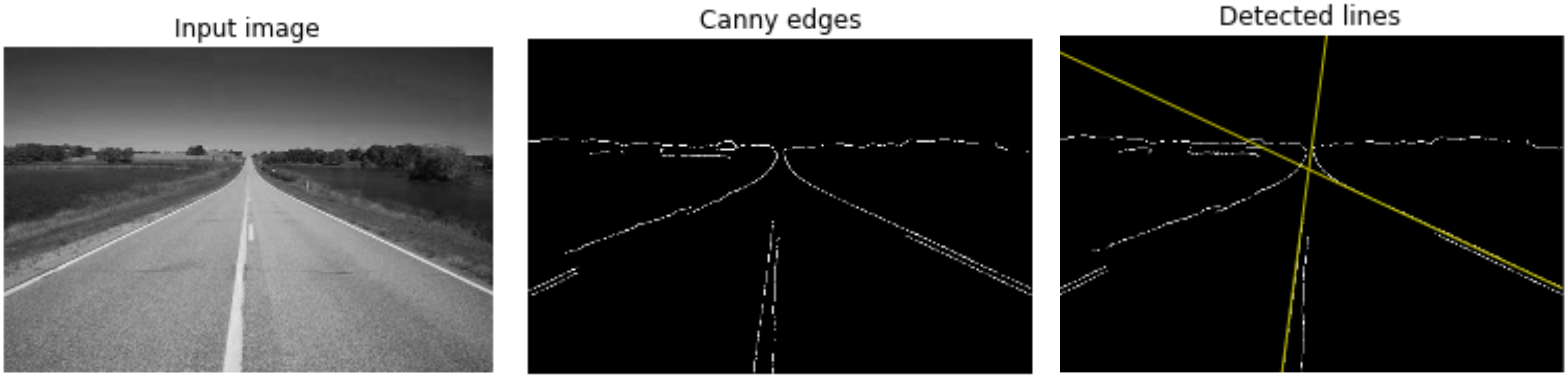
Detection of Edges / Lanes

- Hough Transform

State Estimation

- Monte Carlo Localization

Lane Detection Systems



Input image

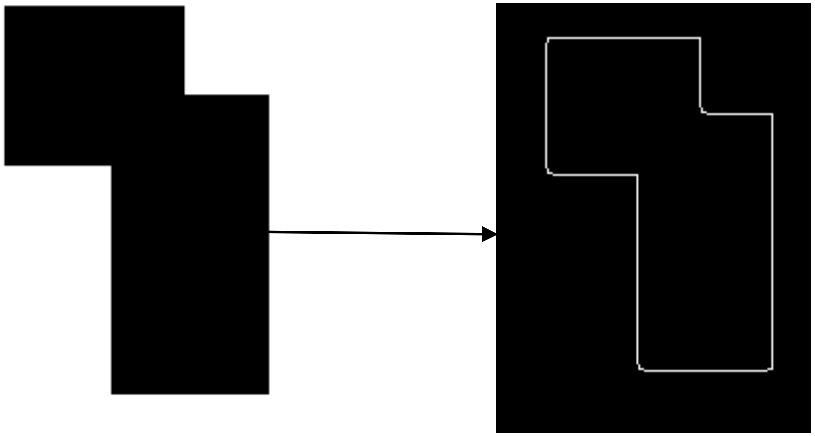
Canny edges

Detected lines

Canny Edge Detection

Canny Edge Detection

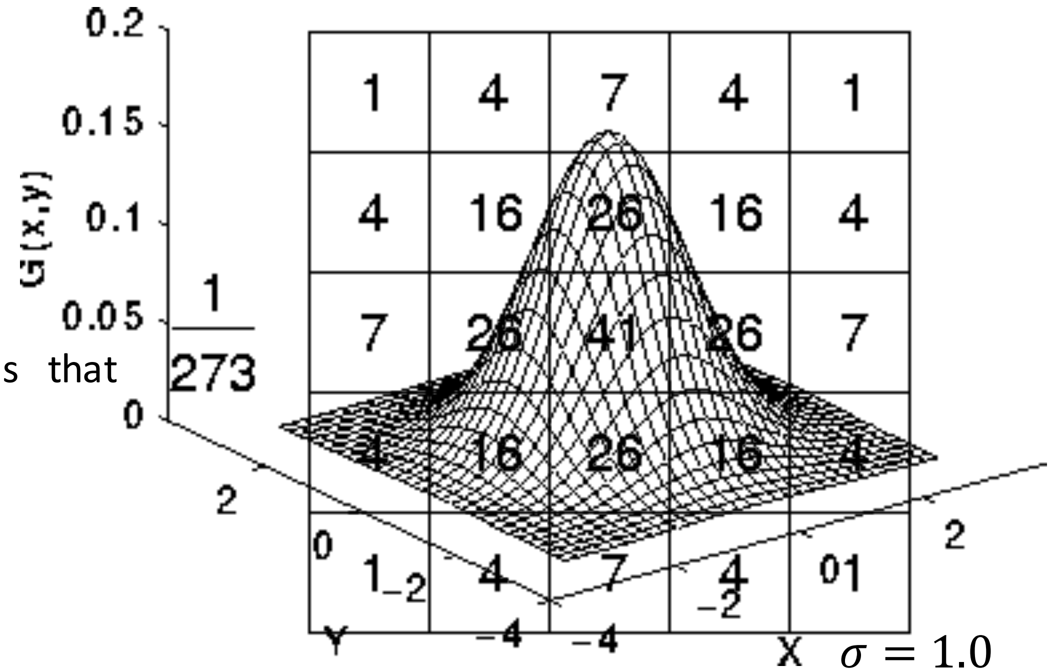
- Optimal Edge Detection algorithm
- Based on three criterion to improve edge detection:
  - Low error rate.
  - Localize edge points
  - Removes Ambiguity



http://homepages.inf.ed.ac.uk/rbf/HIPR2/images

Canny Edge Detection: Algorithm

- **Step 1:**
  - Lower effect of noise
  - Perform **Gaussian Smoothening**
  - Can Control the amount of details that appear in the edge image



$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$\sigma = 1.0$

http://homepages.inf.ed.ac.uk/rbf/HIPR2/images

Canny Edge Detection: Algorithm

- **Step 2:**
  - Finding the edge direction
  - Sobel Operator (or any other) to calculate the gradient magnitude and direction
    - Calculate Gradient along x axis using kernel (G_x)
    - Calculate Gradient along y axis using kernel (G_y)

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

Gx

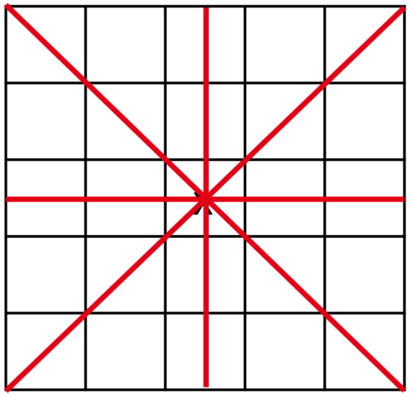
+1	+2	+1
0	0	0
-1	-2	-1

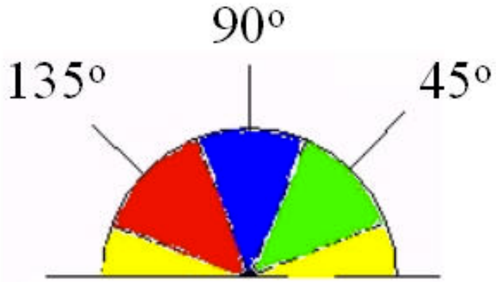
Gy

$$|G| = \sqrt{|G_x|^2 + |G_y|^2} \quad \theta = \arctan\left(\frac{G_y}{G_x}\right)$$

Canny Edge Detection: Algorithm

- **Step 2 (contd.):**
  - If the image is expressed as an array of pixel, there can be only 4 directions for edges
    - 0 degree – Horizontal
    - 90 degree – Vertical
    - 45 degree & 135 degree – Diagonals
  - Gradient angle in yellow = 0
  - Gradient angle in green = 45
  - Gradient angle in blue = 90
  - Gradient angle in red = 135

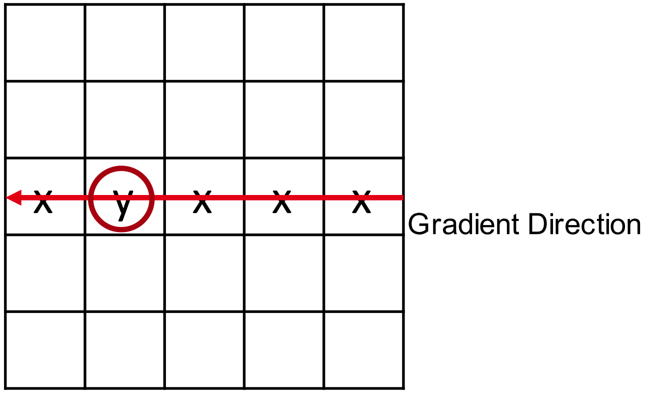




http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html

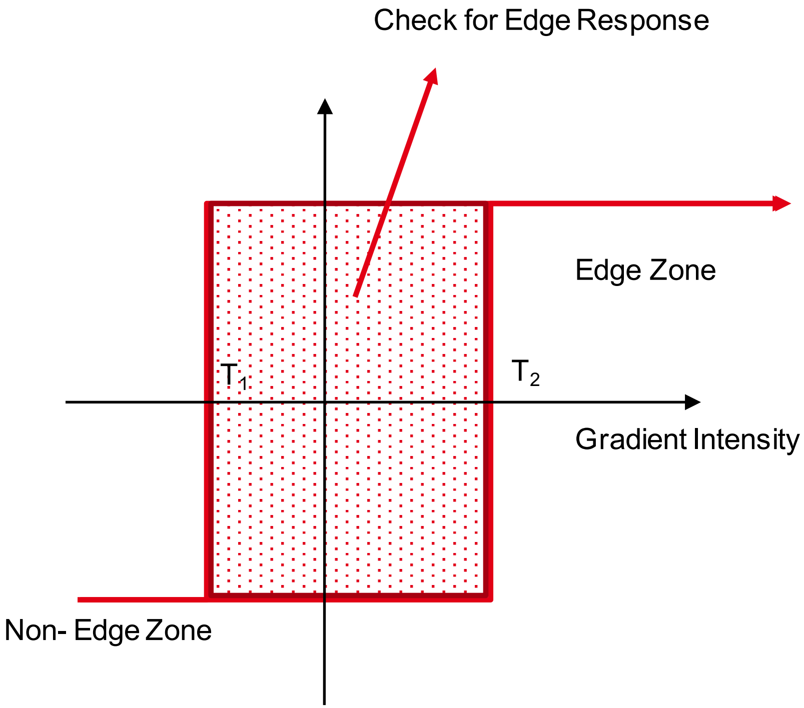
Canny Edge Detection: Algorithm

- **Step 3:**
  - Non-maximal suppression to thin the edge.
  - Trace all pixel values along the edge direction
  - Suppress any pixel which is not considered on the edge i.e. the gradient magnitude is not equal to the local maxima.
  - Leads to thinning of the edge.

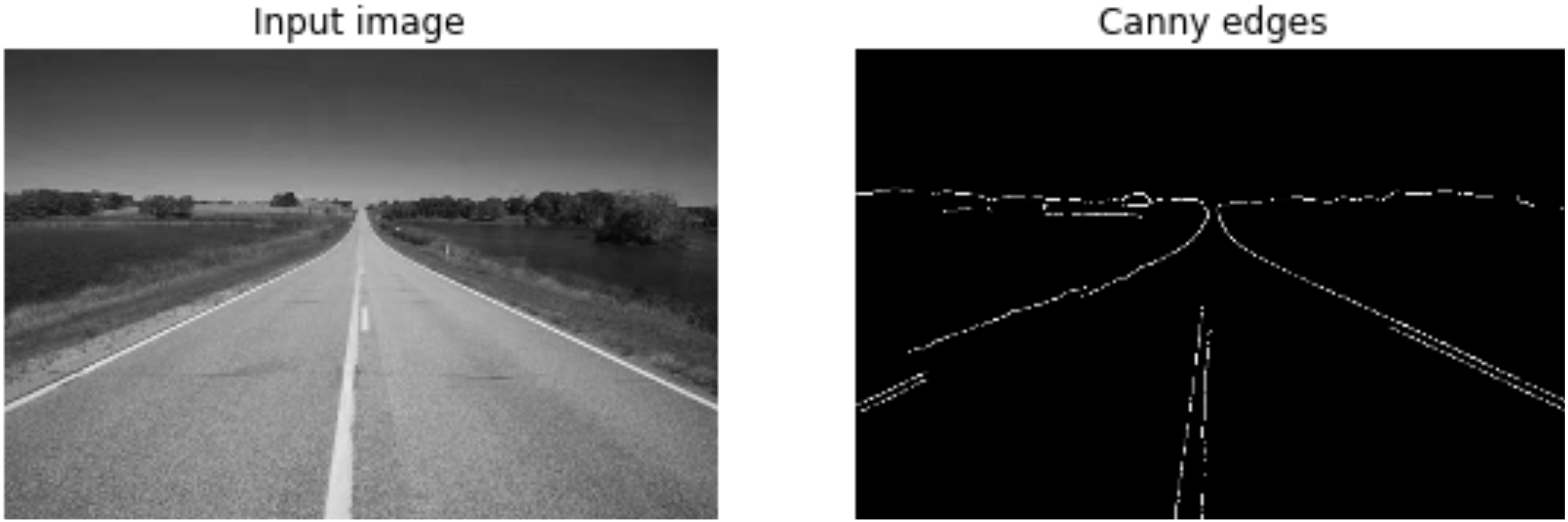


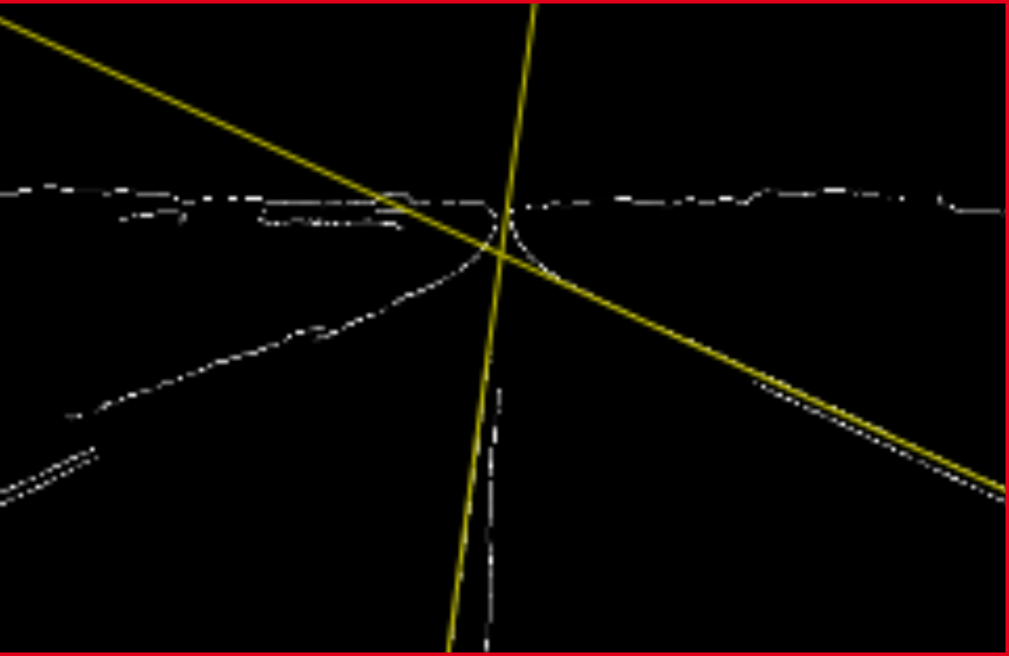
Canny Edge Detection: Algorithm

- **Step 4:**
  - Double Thresholding technique (hysteresis)
  - Reject value < T_1
  - Accept value > T_2
  - If $T_1 <$ value $< T_2$
    - Check for edge response



Canny Edge Detection: Resultant Image

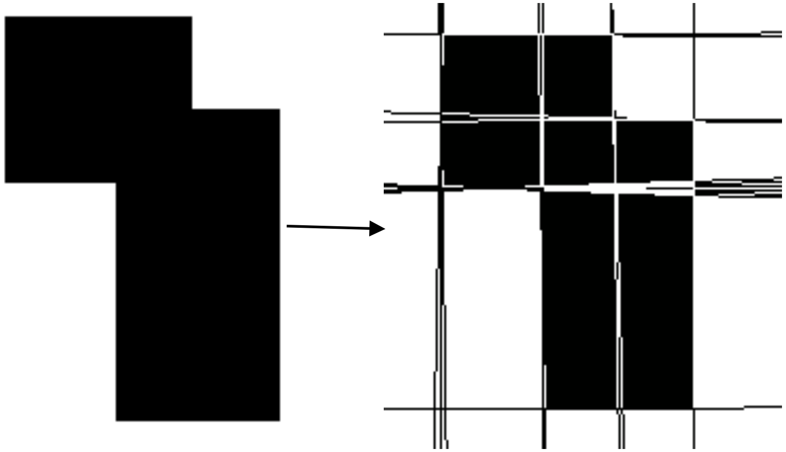




Hough Transforms

Hough Transforms

- Feature Extraction Technique - Used to extract regular (parametric) curves i.e. lines, circles, rectangles etc.
- Used with limited knowledge about the image.
- Given a set of edge points, finds the lines which best explain the data.



http://homepages.inf.ed.ac.uk/rbf/HIPR2/images

Hough Transforms: Theory behind it!

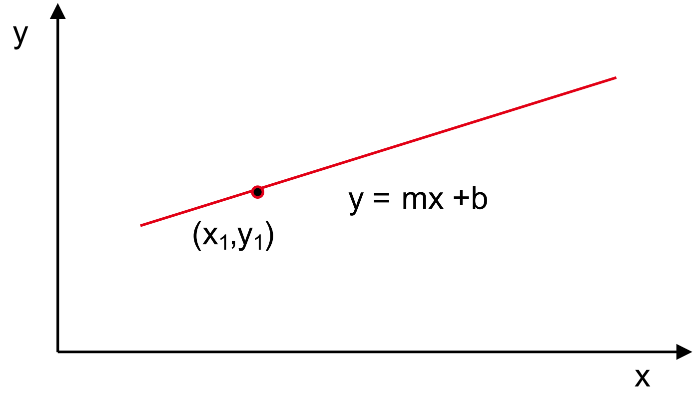
- Equation of a line:

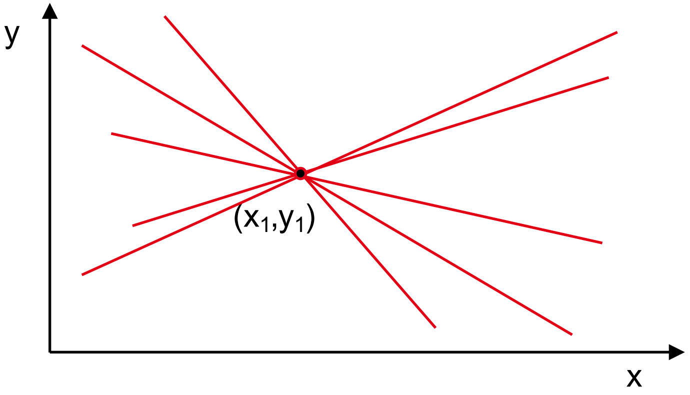
$$y = mx + b$$

- Equation a line passing through a point (x_1,y_1):

$$y_1 = mx_1 + b$$

- Different tuples of (m,b) give all the lines passing through (x_1,y_1)





Hough Transforms: Theory behind it!

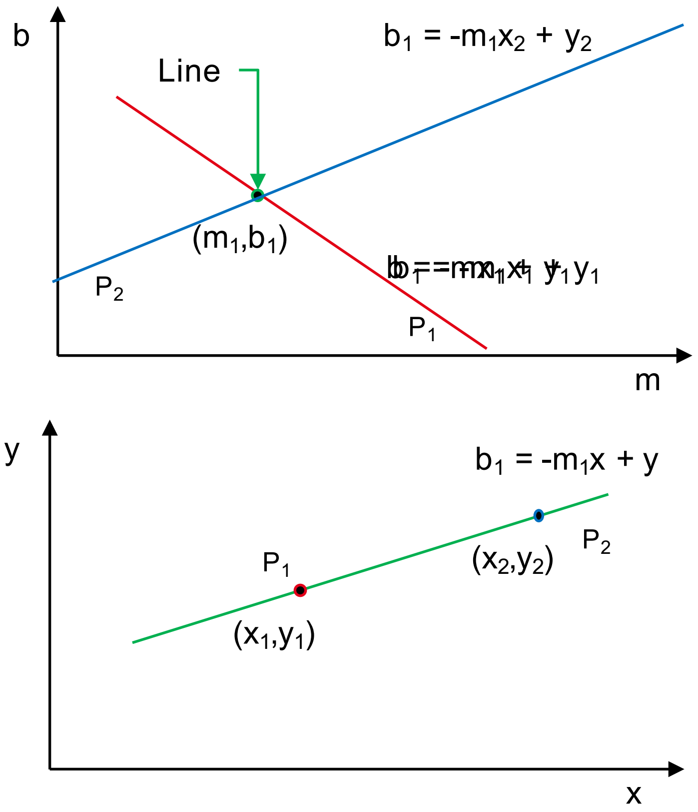
- Change the equation of a line to the (m,b) parameter space:

$$b = -mx_1 + y_1$$

- Consider another line in (m,b) space which passes through (m_1,b_1)

$$b_1 = -m_1x_2 + y_2$$

- Both these lines represent points (x_1,y_1) and (x_2,y_2) in Cartesian space and (m_1,b_1) represents the line joining the two.



Hough Transforms: Theory behind it!

Do you see a problem?

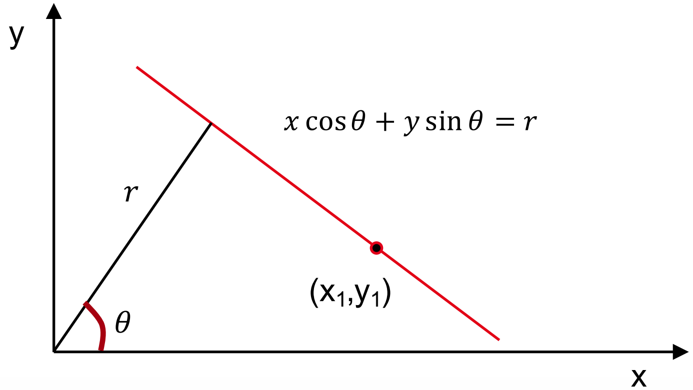
- Parametric equation of a line:

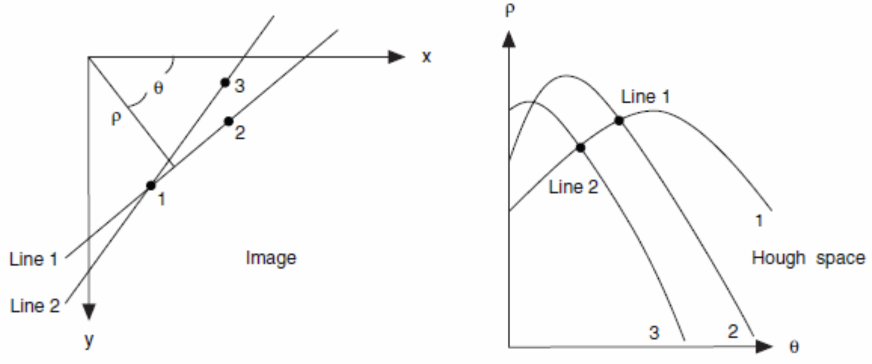
$$x\cos\theta + y\sin\theta = r$$

- Line passing through (x_1,y_1):

$$x_1\cos\theta + y_1\sin\theta = r$$

- Locus of each point is a sinusoid and intersection of sinusoids gives the line.

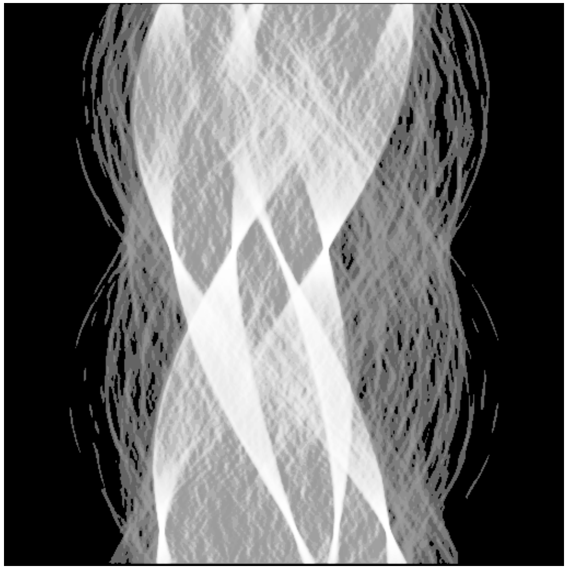




http://zone.ni.com

Hough Transforms: Hough Space Plot

- Sinusoids represent edge points in Cartesian Space.
- Intersection of sinusoids give edges.
- Most prominent edges can be separated from others.



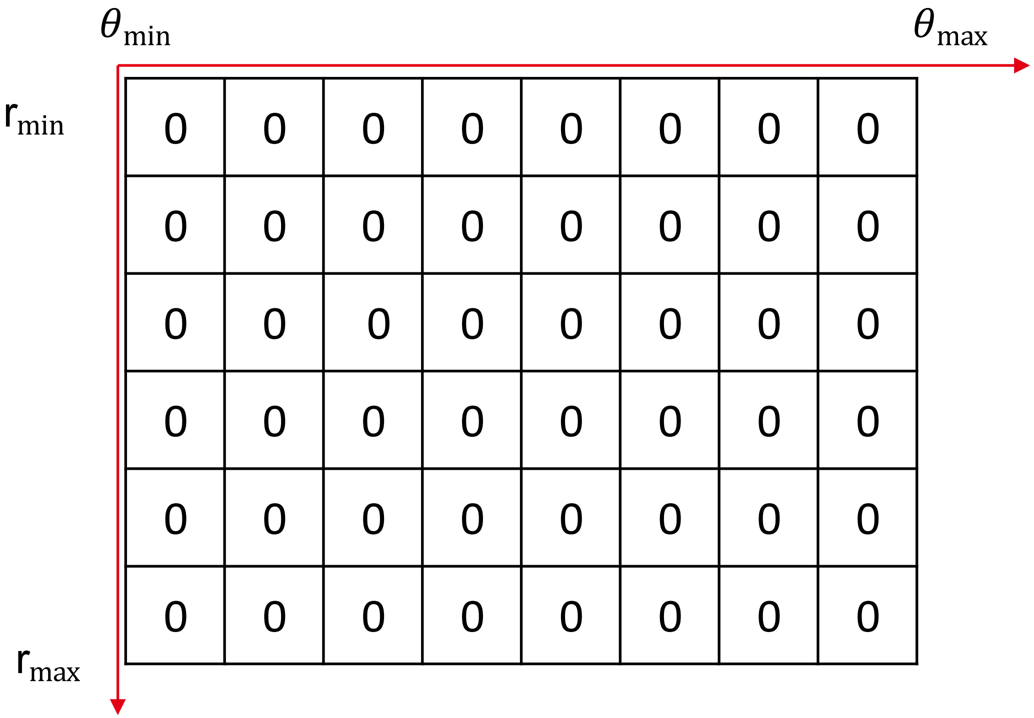
http://homepages.inf.ed.ac.uk/rbf/HIPR2/images

Hough Transform: Accumulator Matrix

- Define a 2D (r, θ) matrix to store 'votes' from each of the edges (in the parameter space) detected. Initialize it to 0.
- For each (x,y) satisfying the equation

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

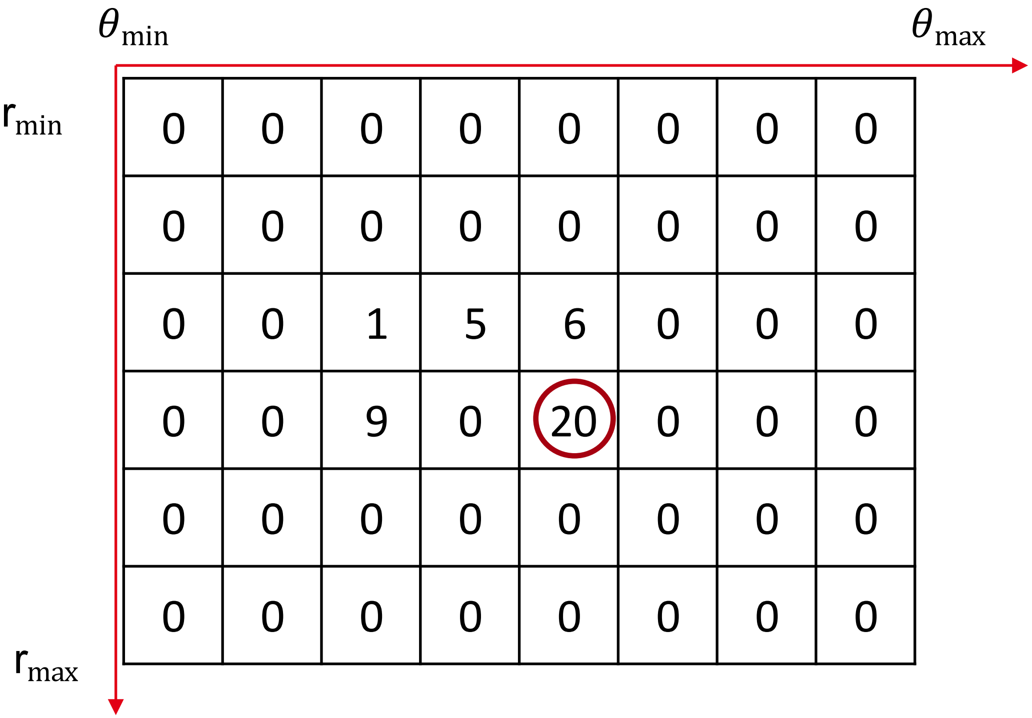
increment the matrix value by 1.



	θ_{min}							θ_{max}
r_{min}	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
r_{max}	0	0	0	0	0	0	0	0

Hough Transform: Accumulator Matrix

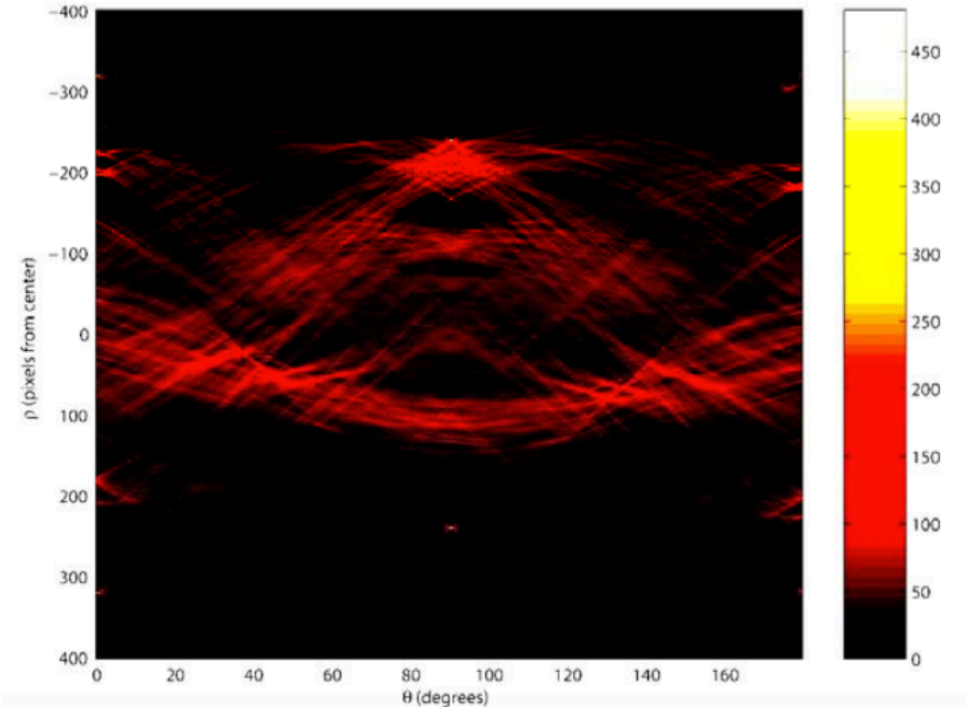
- Value at (r,θ) is the number of points forming a line.
- Maximum value depicts the most prominent edge.



$$\begin{array}{c|cccccccc}
 & \theta_{\text{min}} & & & & & & & \theta_{\text{max}} \\
\hline
r_{\text{min}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 1 & 5 & 6 & 0 & 0 & 0 \\
 & 0 & 0 & 9 & 0 & \bigcirc\!\!\!\!\!\!20 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
r_{\text{max}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

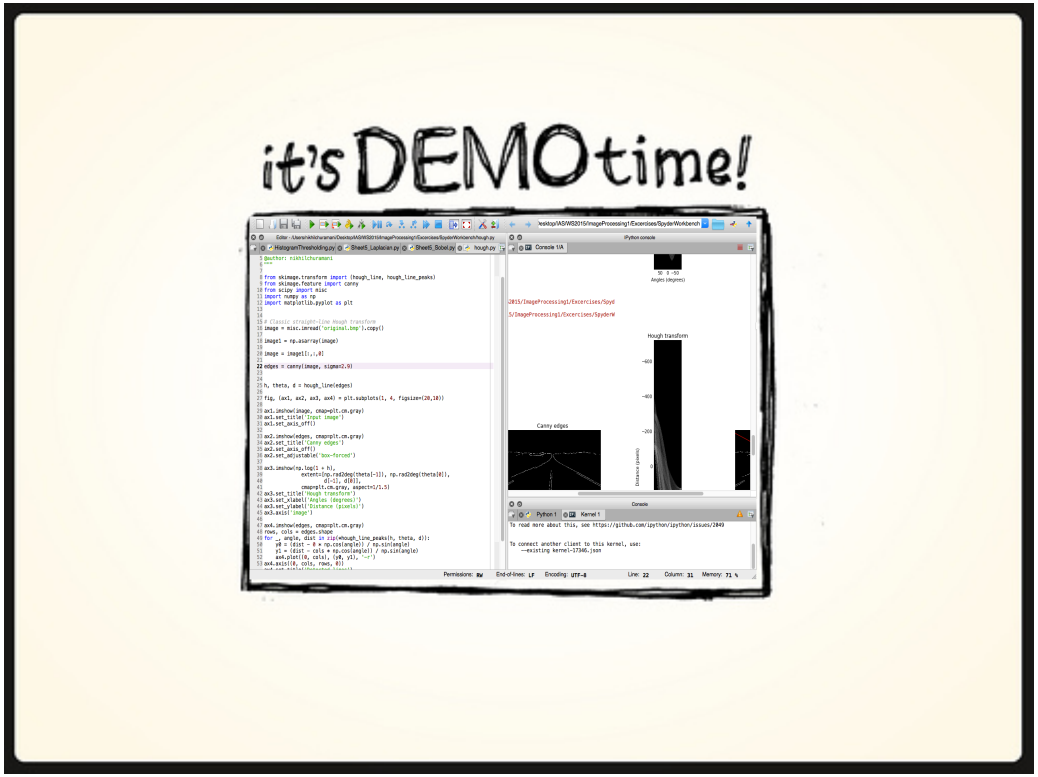
Hough Transform: Histogram and Thresholding

- Create Histogram from Accumulator Matrix
- Thresholding to select strongest edges!



University of Oslo – INF4300 (Anne Solberg)

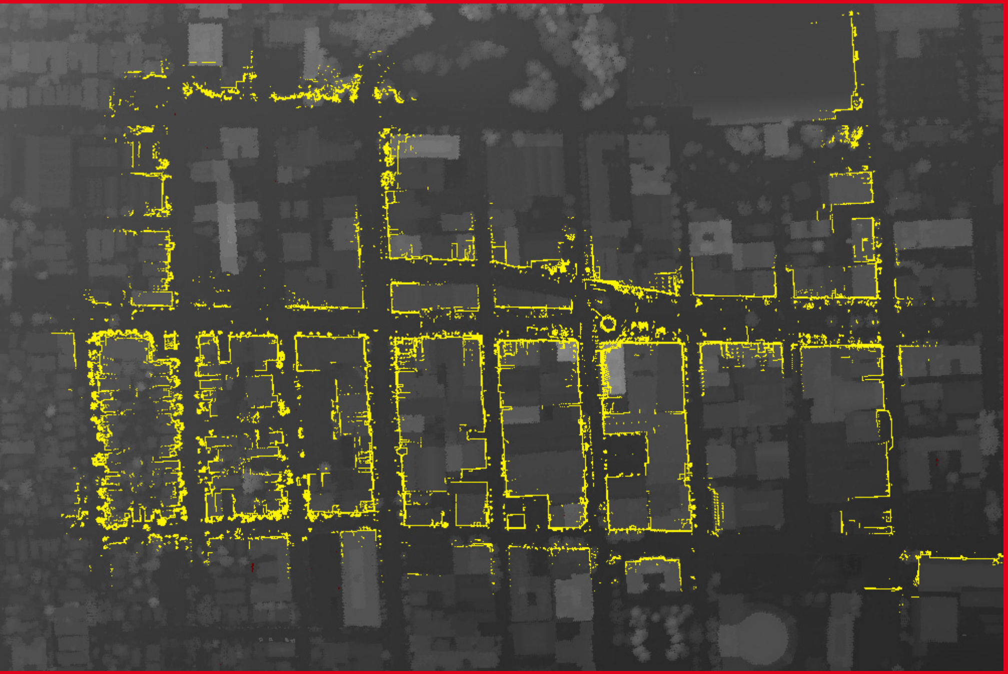
Hough Transform



```python
@author: nikhilchuramani
"""

from skimage.transform import (hough_line, hough_line_peaks)
from skimage.feature import canny
from scipy import misc
import numpy as np
import matplotlib.pyplot as plt


# Classic straight-line Hough transform
image = misc.imread('original.bmp').copy()

image1 = np.asarray(image)

image = image1[:,:,0]

edges = canny(image, sigma=2.9)


h, theta, d = hough_line(edges)

fig, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(20,10))

ax1.imshow(image, cmap=plt.cm.gray)
ax1.set_title('Input image')
ax1.set_axis_off()

ax2.imshow(edges, cmap=plt.cm.gray)
ax2.set_title('Canny edges')
ax2.set_axis_off()
ax2.set_adjustable('box-forced')

ax3.imshow(np.log(1 + h),
             extent=[np.rad2deg(theta[-1]), np.rad2deg(theta[0]),
                     d[-1], d[0]],
             cmap=plt.cm.gray, aspect=1/1.5)
ax3.set_title('Hough transform')
ax3.set_xlabel('Angles (degrees)')
ax3.set_ylabel('Distance (pixels)')
ax3.axis('image')

ax4.imshow(edges, cmap=plt.cm.gray)
rows, cols = edges.shape
for _, angle, dist in zip(*hough_line_peaks(h, theta, d)):
    y0 = (dist - 0 * np.cos(angle)) / np.sin(angle)
    y1 = (dist - cols * np.cos(angle)) / np.sin(angle)
    ax4.plot((0, cols), (y0, y1), '-r')
ax4.axis((0, cols, rows, 0))
```

```
2015/ImageProcessing1/Excercises/Spyd

5/ImageProcessing1/Excercises/SpyderW
```

```
To read more about this, see https://github.com/ipython/ipython/issues/2049


To connect another client to this kernel, use:
    --existing kernel-17346.json
```



Monte Carlo Localization

http://www-video.eecs.berkeley.edu/~frueh/3d/posest/montecarlo.html

Monte Carlo Localization for Automated Driving

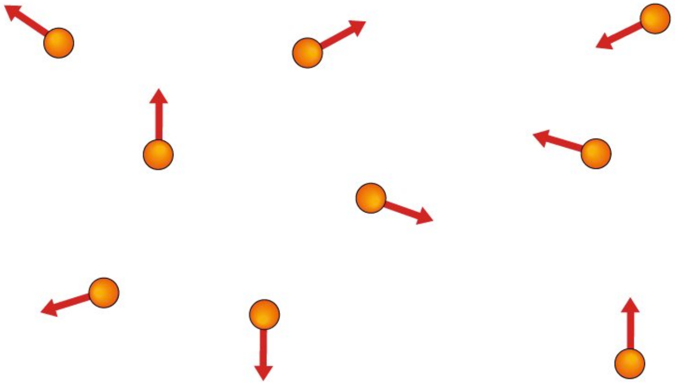
http://www.cs.cmu.edu/~youngwoo/img/omni-view.jpg

Monte Carlo Localization: Theory

- Set of particles / samples:

$$\chi_t = \{x^i | \, x^i \in \chi_t\}$$

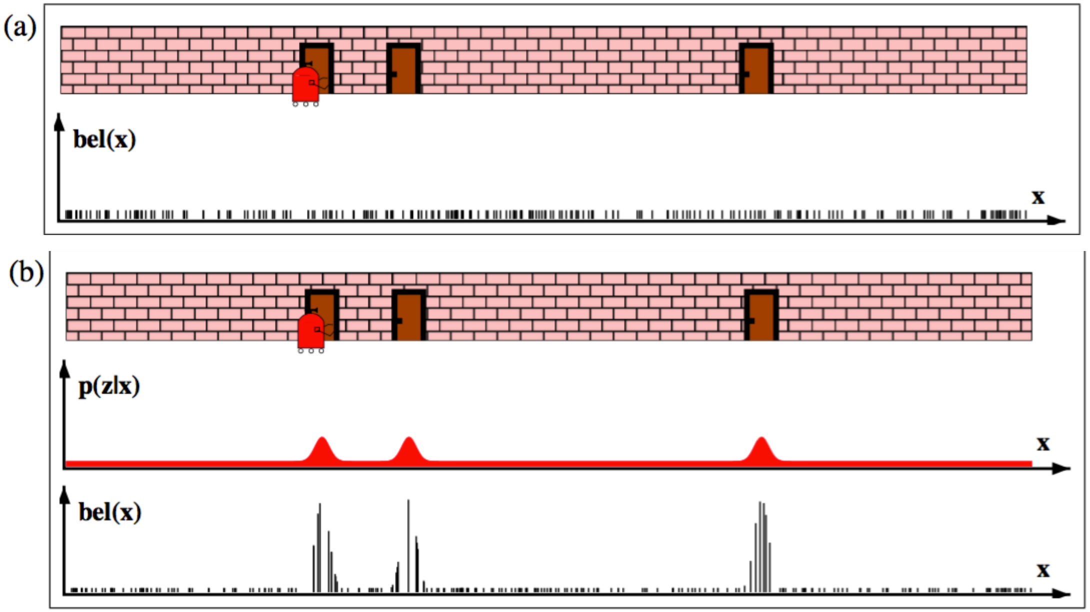
- Higher the number of particles, more accurate the approximation
- *bel(*χ_t*)* formed based on the particle set χ_t
- ω_t is the weight corresponding to each particle.



http://surfguppy.com

Monte Carlo Localization: Understanding

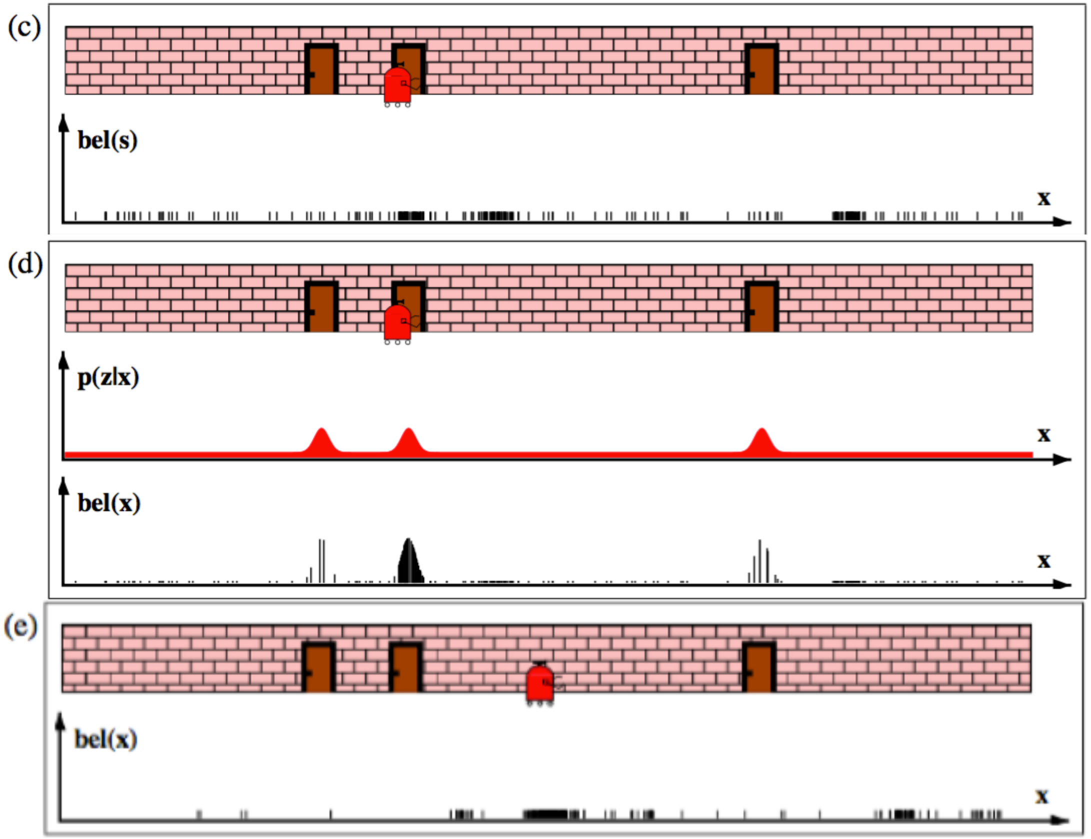
1. Initialize set of particles (or beliefs about the robot's location.)
2. Gather data from sensors
3. Evaluate observed data against the belief and update weight of particles accordingly.



[9] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

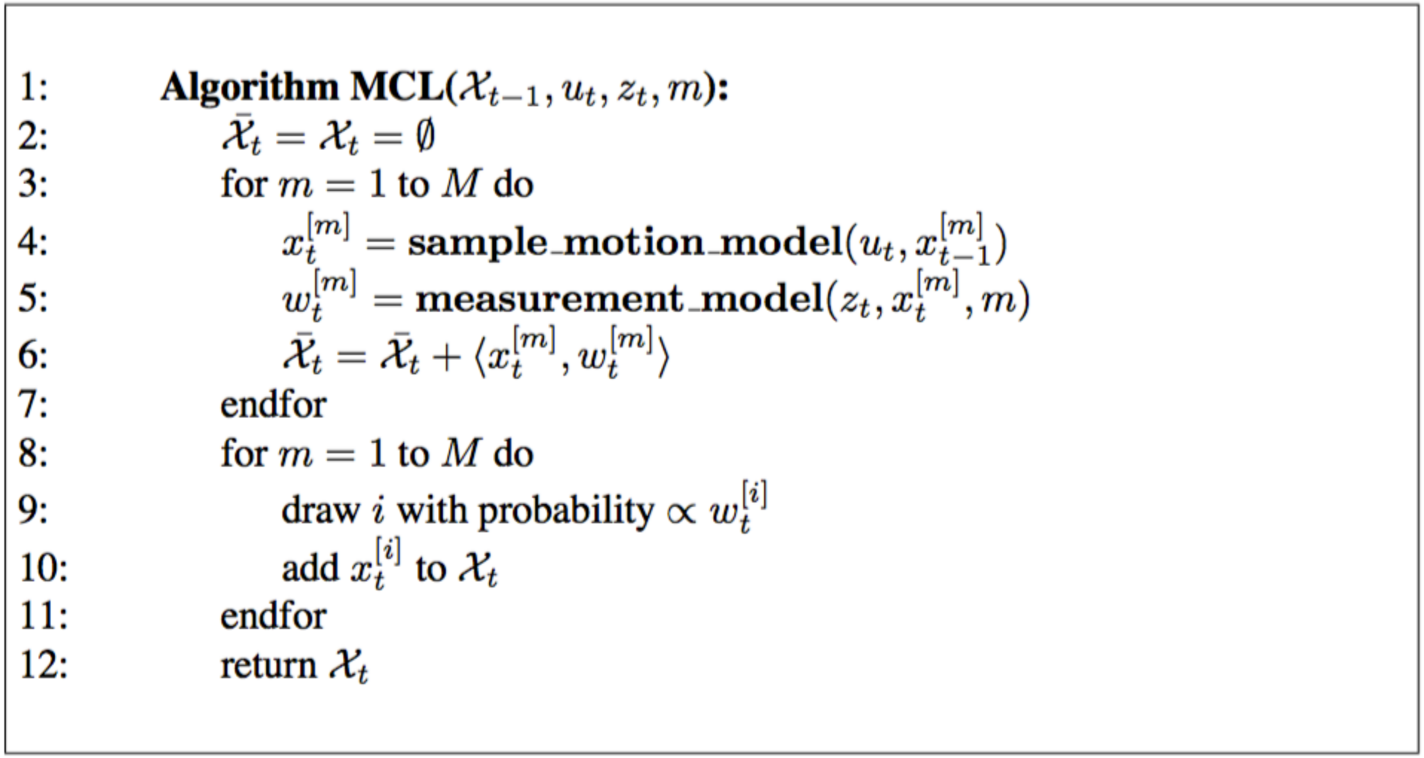
Monte Carlo Localization: Understanding

1. Initialize set of particles (or beliefs about the robot's location.)
2. Gather data from sensors
3. Evaluate observed data against the belief and update weight of particles accordingly.
4. Sample new particles based on the updated weight values
5. Repeat step 2-5



[9] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

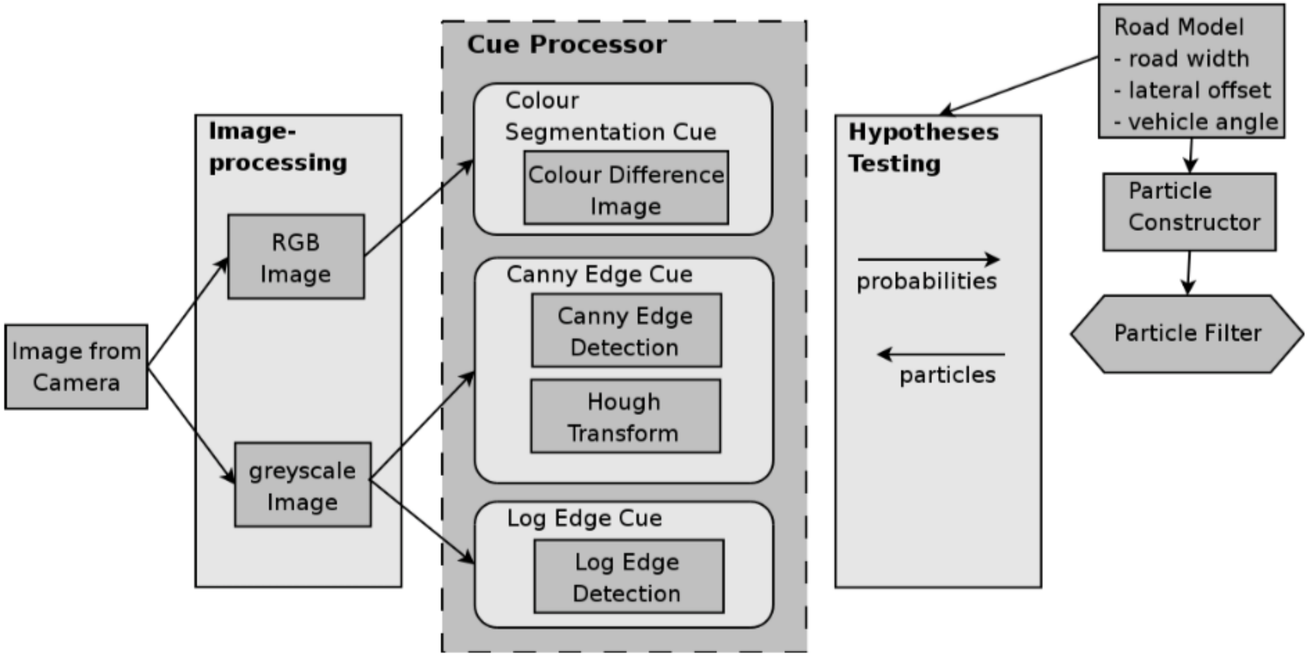
Monte Carlo Localization: Algorithm



```
Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
    for $m = 1$ to $M$ do
        $x_t^{[m]} =$ sample_motion_model($u_t, x_{t-1}^{[m]}$)
        $w_t^{[m]} =$ measurement_model($z_t, x_t^{[m]}, m$)
        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    endfor
    for $m = 1$ to $M$ do
        draw $i$ with probability $\propto w_t^{[i]}$
        add $x_t^{[i]}$ to $\mathcal{X}_t$
    endfor
    return $\mathcal{X}_t$
```

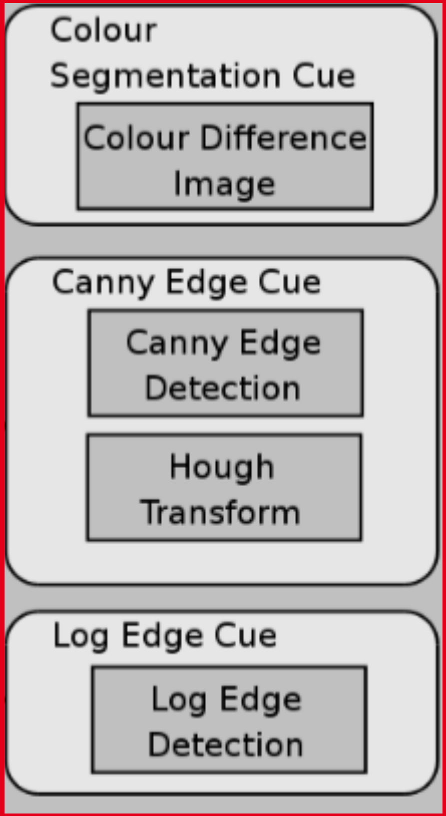
[9] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

Monte Carlo Localization: Lane Detection



[3] Macek, Kristijan, et al. "A lane detection vision module for driver assistance."*None*. No. LSA-CONF-2004-006. 2004.

Quality Evaluation



Colour Segmentation Cue

Colour Difference Image

Canny Edge Cue

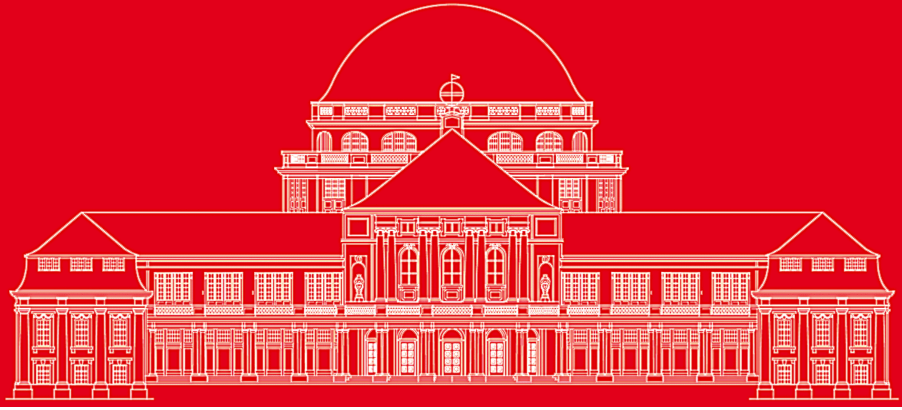
Canny Edge Detection

Hough Transform

Log Edge Cue

Log Edge Detection

[3] Macek, Kristijan, et al. "A lane detection vision module for driver assistance." *None*. No. LSA-CONF-2004-006. 2004.



Quality Evaluation

- **Multiple Cues improve** the **performance** as more information is available for the Particle Filter.
- **Canny Edge - Hough Transform** proves to be most **robust** with **change in scene and to noise**. Is pixel position insensitive.
- **Canny Edge - Hough Transform** Cue works in edge space (parameter) thus **less computation complexity.**
- Color Cue **sensitive** to **brightness change** and shadows.
- LoG filter does **not contain edge direction** information.
- **LoG** becomes **insensitive** to **objects occluding** lane boundaries.
- **Color Cue** and **LoG** work in the image space thus **computationally expensive**.

Bibliography

[1] Hamarneh, Ghassan, Karin Althoff, and Rafeef Abu-Gharbieh. "Automatic line detection." *Project Report for the Computer Vision Course Lund, Simon Fraser University* (1999).

[2] Liu, Guoliang, Florentin Wörgötter, and Irene Markelić. "Combining statistical Hough transform and particle filter for robust lane detection and tracking."*Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010.

[3] Macek, Kristijan, et al. "A lane detection vision module for driver assistance."*None*. No. LSA-CONF-2004-006. 2004.

[4] Yu, Bin, and Anil K. Jain. "Lane boundary detection using a multiresolution hough transform." *Image Processing, 1997. Proceedings., International Conference on*. Vol. 2. IEEE, 1997.

[5] Saudi, Azali, et al. "Fast lane detection with randomized hough transform."*Information Technology, 2008. ITSim 2008. International Symposium on*. Vol. 4. IEEE, 2008.

[6] Satzoda, Ravi Kumar, et al. "Hierarchical additive Hough transform for lane detection." *Embedded Systems Letters, IEEE* 2.2 (2010): 23-26.

[7] Bengler, Klaus, et al. "Three Decades of Driver Assistance Systems."

[8] Sharma, Sachin, and D. J. Shah. "A much advanced and efficient lane detection algorithm for intelligent highway safety." *Computer Science & Information Technology* (2003): 51.

[9] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

Q&A

http://inquisitr.com

Towards Smarter Cars…

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Intelligent Cars

Driver Assistance Systems

Lane Detection and Changing System

Nikhil Churamani

64.425 Integrated Seminar: Intelligent Robotics
WiSe – 2015