5LIA0 – Embedded Visual Control

Computer Vision

Egor
Bondarev

Personal Introduction

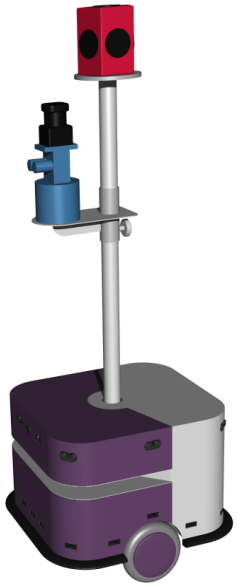
- MSc in Robotics, Belarus
- Invention Machine Corp – Artificial Intellect
- OOTI, Software Technology, Post-master, TUE
- PhD on Performance of Real-Time Systems, TUE
- Currently:
  - Assistant Professor at TUE: 3D Sensing and Reconstruction
  - CEO of IVRA Holding

VCA Group Experience: Sensing Platforms

Set of 2D/3D Sensing Platforms

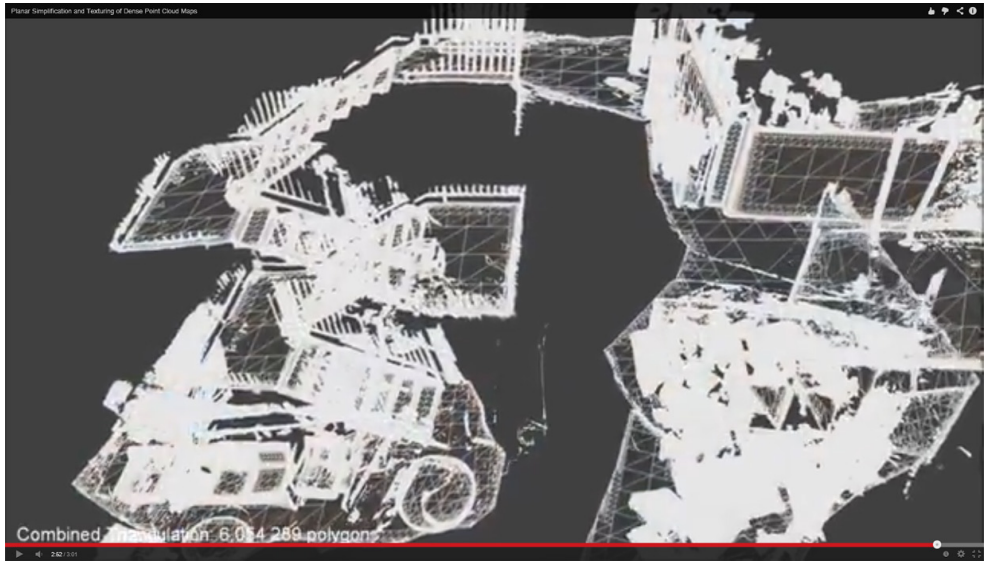
- Few laser-scanner platforms (LIDARs)
- RGB-D mapping system (Kinect)
- Stereo Cameras, Panoramic 360 camera
- Autonomous robot
- Thermal cameras

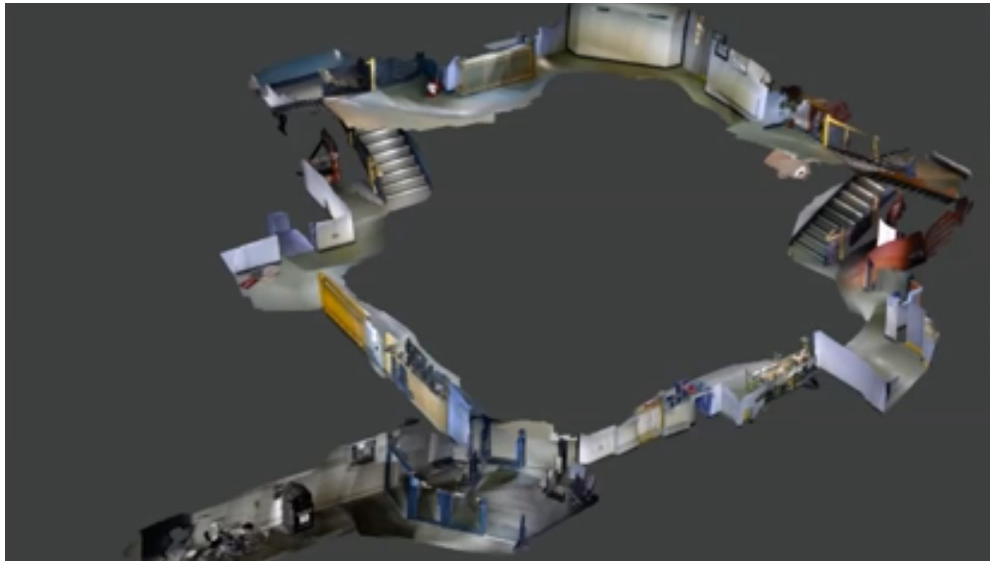
All of them have 1 or more computers inside



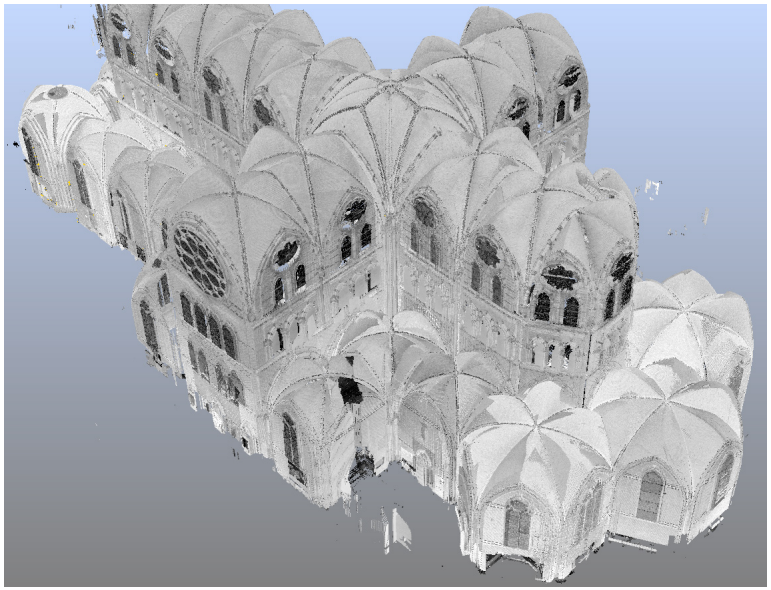
What we do at VCA:
Detailed 3D Reconstruction & Texturing

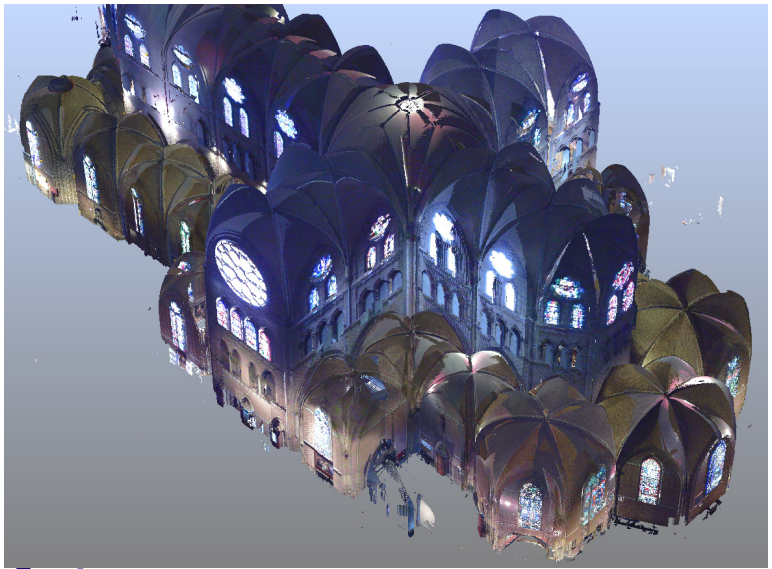
What we do at VCA: Complex spaces into 3D





What we do at VCA: Extremely Large Spaces





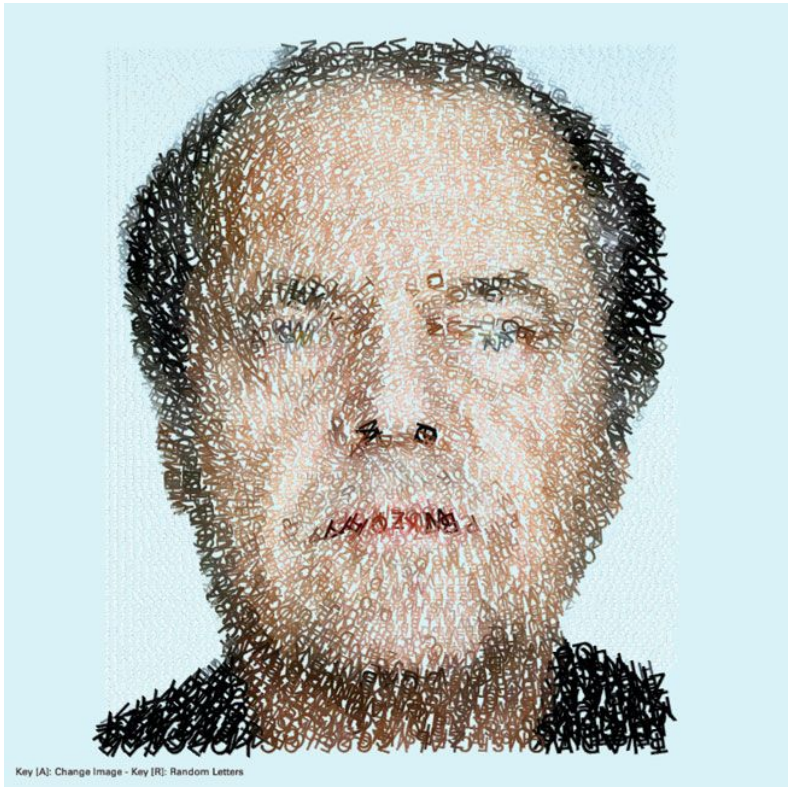
What we do at VCA: Demo. Room in 3D

Welcome!

These hours:

- Computer Vision Introduction
- Vision for Autonomous Vehicles
- Lane Detection
  - Canny filter
  - Hough Transform
- Traffic Signs Detection
  - Next lecture

omputer Vision in Gener



Make computers understand images and videos

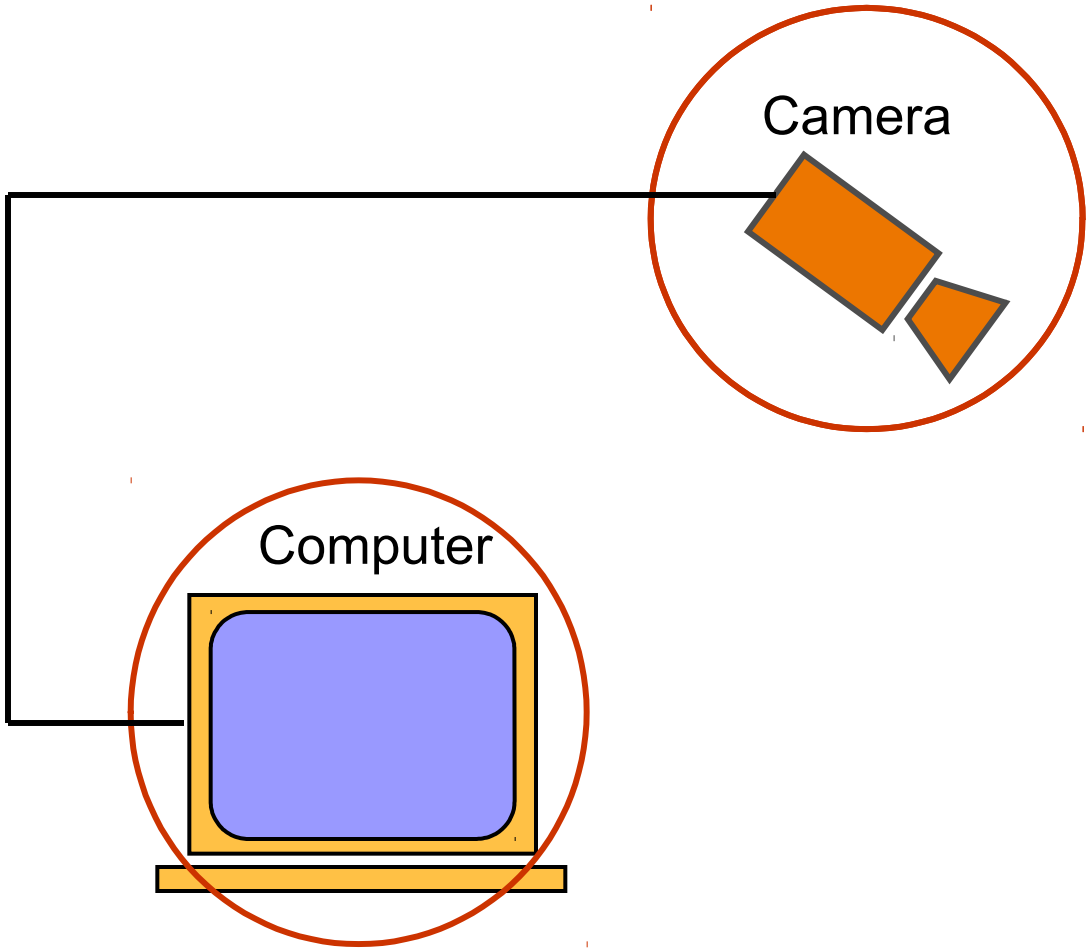
What kind of scene?

Where the camera is located at this scene?

Where are the cars?

How far is the building?

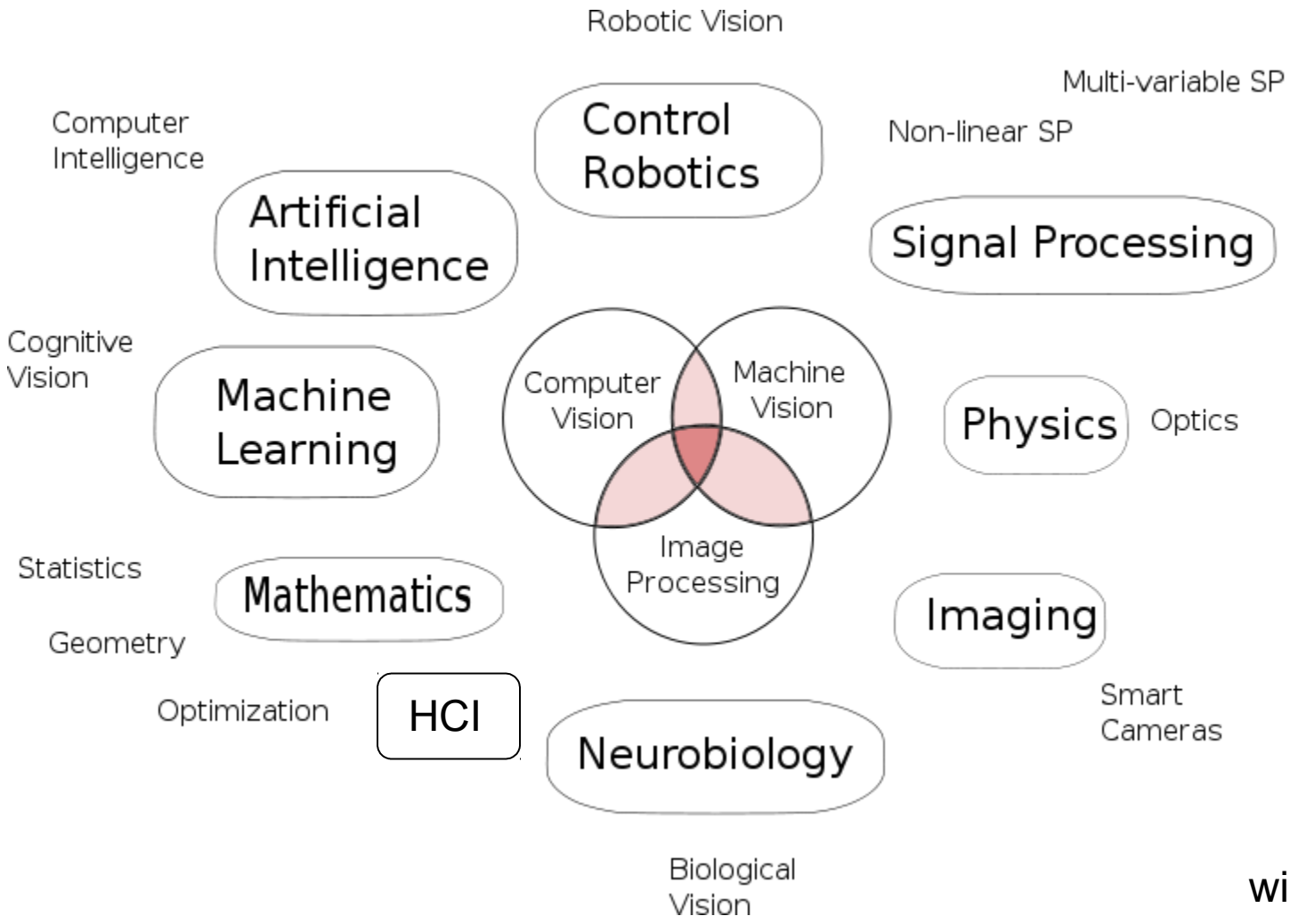
Components of a computer vision system



Lighting

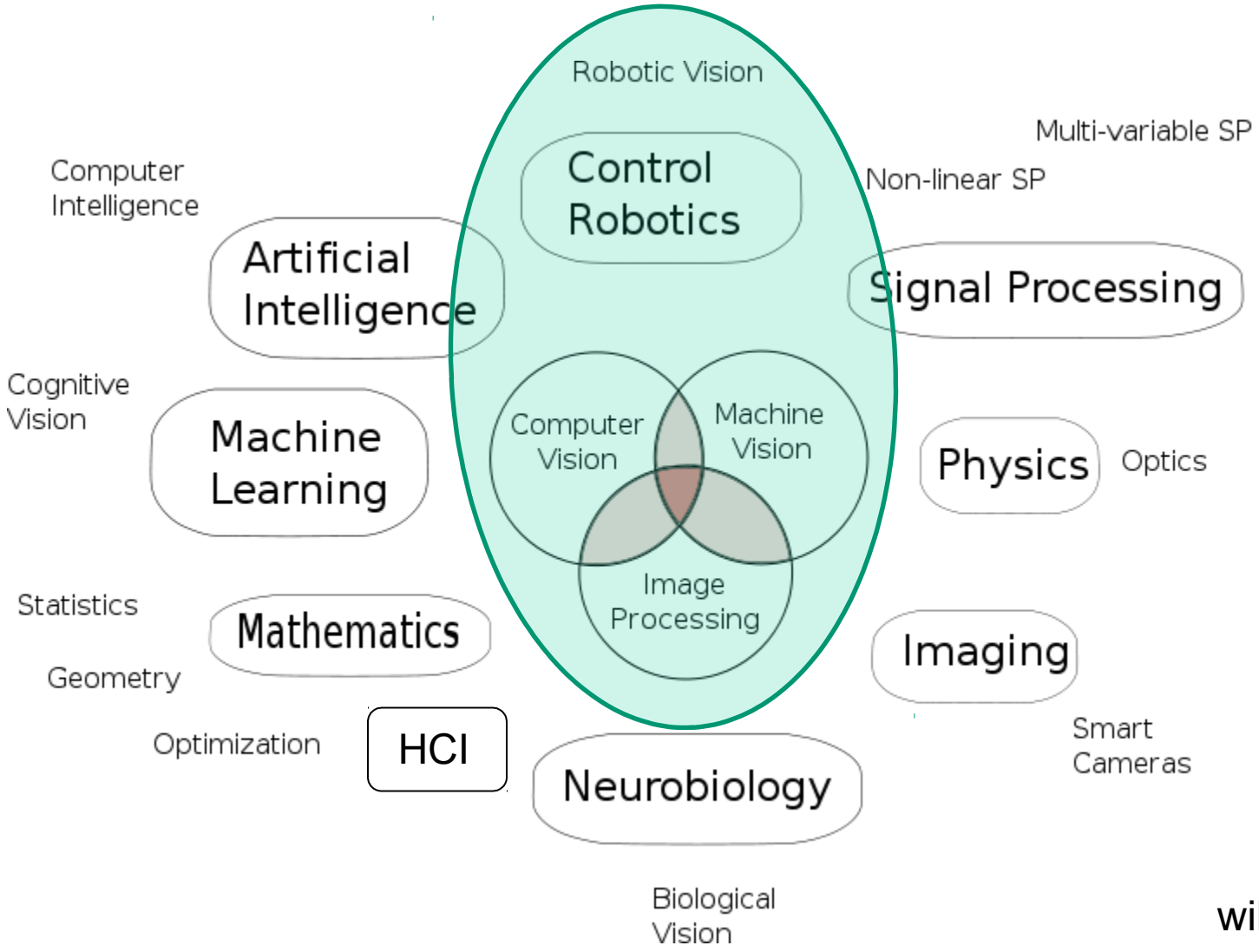
Scene

Application Domains



wiki

Application for Robotics



wiki

TU/e

Computer Vision for Autonomous Vehicles

Vision in Robotics, Autonomous Vehicles

TU/e

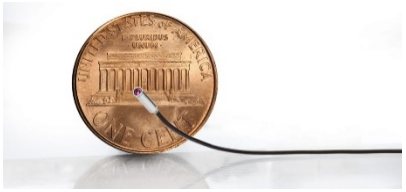
- Major goals
  - Identify camera (car) *position and orientation* at every moment in time - localization
  - Understand surroundings – mapping of scene around
  - Detect and recognize objects:
    - Traffic signs
    - Pedestrians
    - Lanes
    - Other vehicles
  - Track (to analyze situation around_:
    - Road bumps – to control suspension
    - Cars around – for collision avoidance, platooning
  - Compute optimal path from A to B

- Ultimate goal
  - Enable fully autonomous driving

Sensors in Vehicle Vision

RGB camera

Ultrasound sensors



Supportive sensors:

GPS,

Odometers,

IMU

(accelerators,

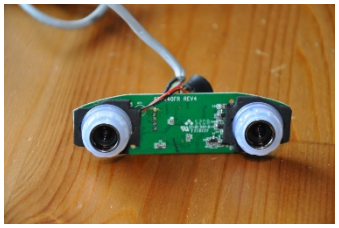
gyroscopes,

magnetometers,

altimeters),

Acoustic

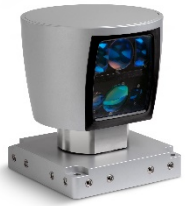
Stereo camera



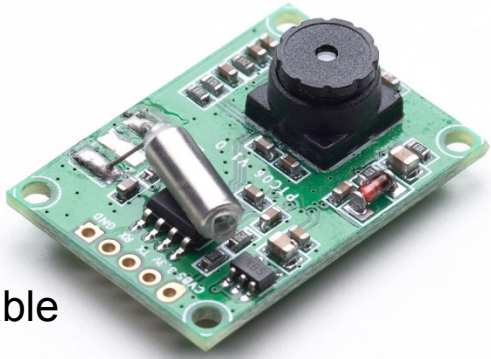
Thermal camera

IR sensors

Laser scanners - LiDAR



Sensors in Vehicle Vision



- RGB camera
  - Advantages
    - Low cost: from 1 euro
    - Long range: till horizon
    - Lots of rgb-based image processing algorithms available
    - High resolution
    - Data post-processing is not required
  - Disadvantages
    - No depth data, 2D picture
    - Low performance for occlusions, cluttered scenes
    - Blind at night, low performance in fog, rain
    - High data rate
  - For which functions
    - All of the described above

Sensors in Vehicle Vision

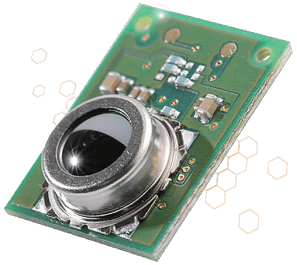
- Stereo camera
  - Advantages
    - Depth data can be obtained!
    - High performance for occlusions, cluttered scenes
    - Long range: till horizon, depth range is limited
    - Depth-based image processing algorithms available, but limited
    - Relatively high resolution (HD)
  - Disadvantages
    - High cost: from 100 euro
    - Data post-processing is required: disparity map computation is heavy
    - Blind at night, low performance in fog, rain
    - Noisy depth data
  - For which functions
    - All *short and medium* range functions described above

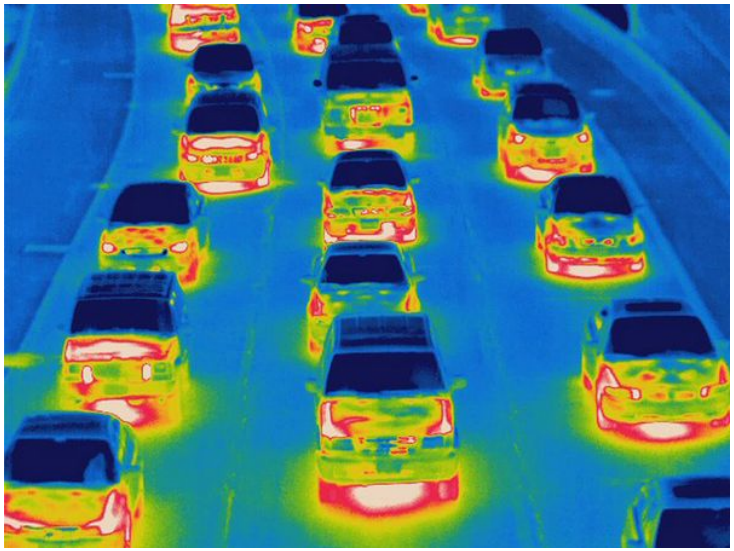
Sensors in Vehicle Vision

- Kinect-like (Structured Light, or Time of Flight)
  - Advantages
    - Depth data is directly available!
    - Data post-processing is not required
    - Can see depth at night
    - High performance for occlusions, cluttered scenes
    - Depth-based image processing algorithms available, but limited
    - Relatively high resolution (HD)
  - Disadvantages
    - Very short range: up to 50 m
    - High cost: from 100 euro
    - Does not work outdoor!
    - Noisy depth data
  - For which functions
    - Only *short* range functions described above

Sensors in Vehicle Vision

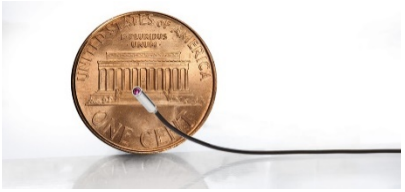
- Thermal camera
  - Advantages
    - Can see perfectly at night
    - Long range
    - Data post-processing is not required
    - Low data rate
  - Disadvantages
    - High cost: from 300 euro
    - Low resolution
  - For which functions
    - Mostly for night vision, human and car detection in cluttered scenes





Sensors in Vehicle Vision

- IR, Ultrasound sensors
  - Advantages
    - Low cost
    - Robust, works at night, rain, fog
  - Disadvantages
    - Short range: up to 20 m
    - Very low resolution
  - For which functions
    - Used mostly for obstacle detection



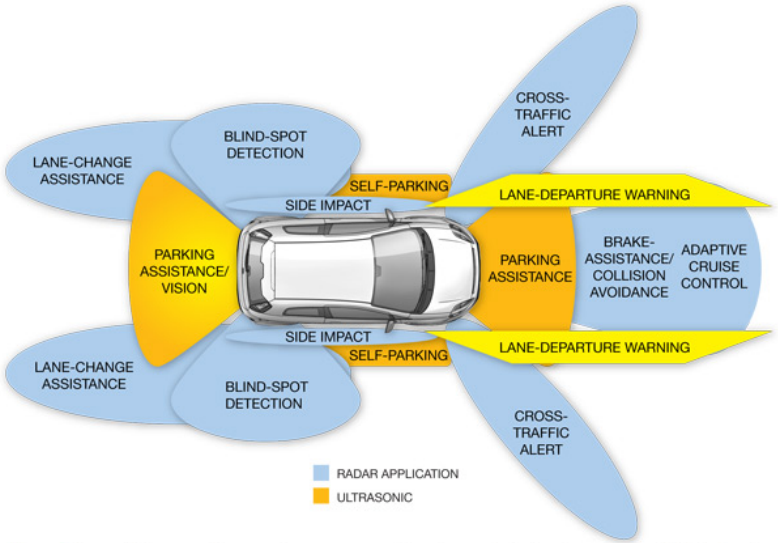
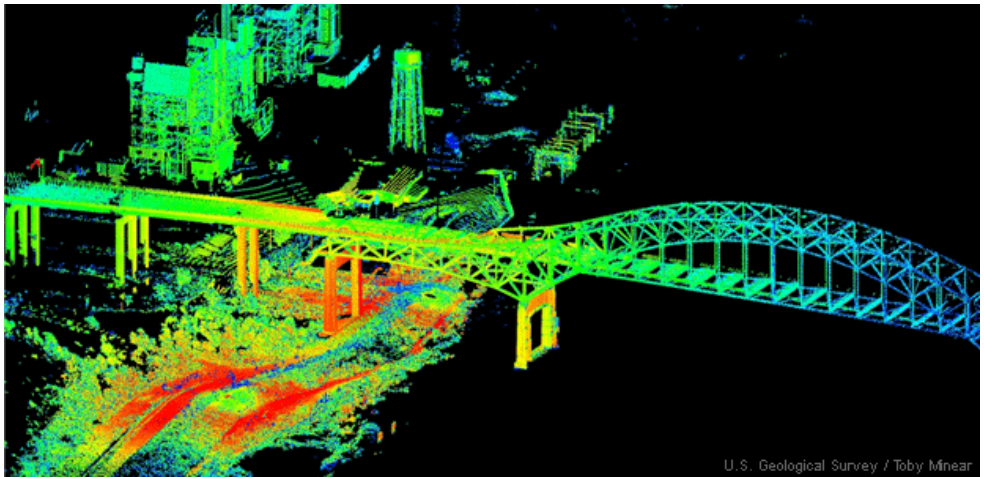


Figure 2 Several driver-assistance systems are currently using radar technology to provide blind-spot detection, parking assistance, collision avoidance, and other driver aids (courtesy Analog Devices).

Sensors in Vehicle Vision

- Laser scanner - LiDAR
  - Advantages
    - Full 3D data around
    - Point-clouds algorithms available, limited
    - High performance for occlusions, cluttered scenes
    - Very high resolution, 10 mln points/second
    - Data post-processing is not required
    - Can see at night, fog and rain
  - Disadvantages
    - Very high cost: 6000 euro
    - Medium range: up to 300 meters
    - Very high data rate
  - For which functions
    - All of the described above + 3D reconstruction of road map



TU/e

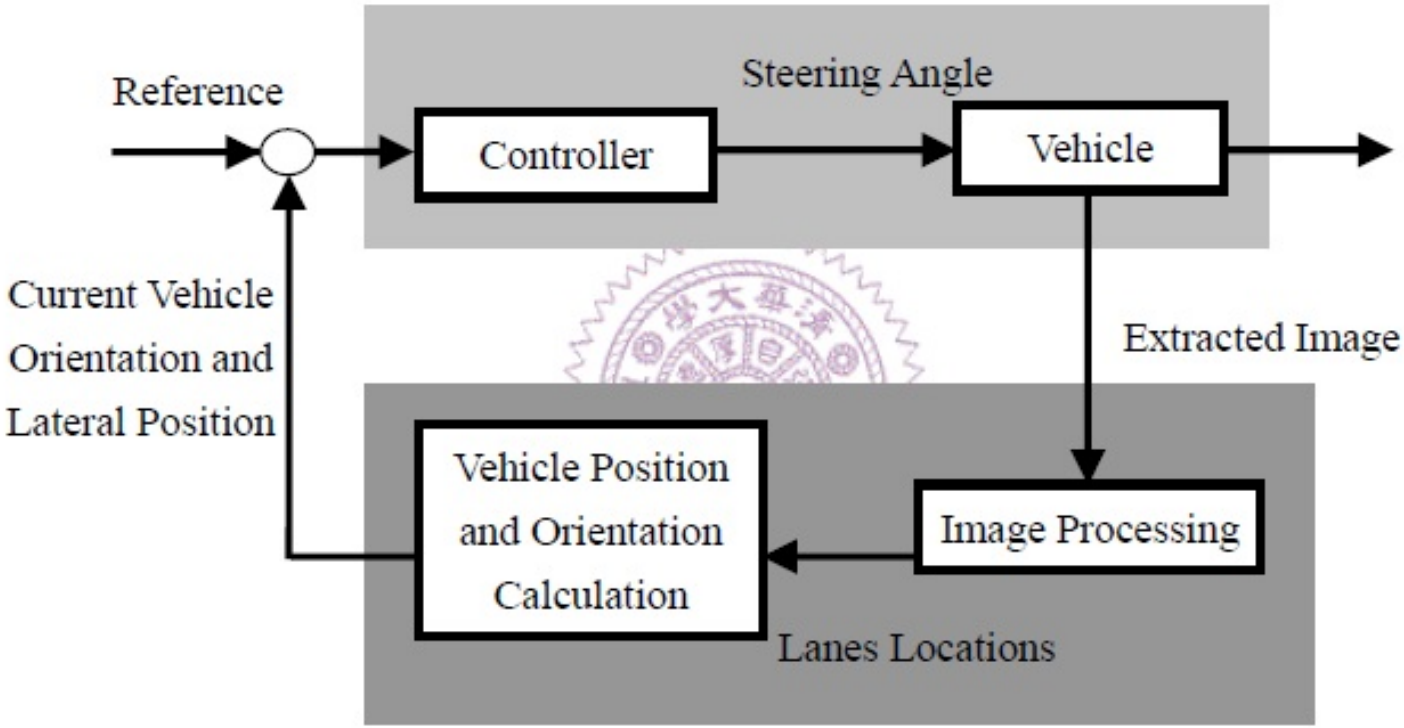
Specific Vision Tasks

Lane Tracking

Lane Tracking

- Challenging task
  - Varying road conditions vary
  - Occlusions (parked vehicles)
  - Lane curves
  - Shadows
  - Slopes
  - Tough weather conditions
- Two types of approaches
  - Model based
    - Less sensitive to lane quality, occlusions
    - Not robust for unexpected scenes and non-standard lane marking
    - Computationally expensive – model fitting optimization
  - Feature based
    - Very sensitive to occlusions, poor road marking
    - Robust to an arbitrary marking
    - Less computationally expensive

Lane Following Pipeline

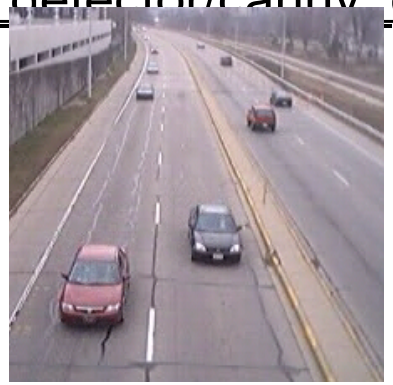


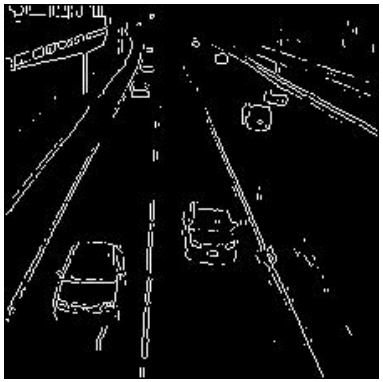
Conventional Lane Detection Pipeline (many others exist)

- Step 0: Pre-filtering: subtract background, filter image
- Step 1: Get the edge information
- Step 2: Hough Transform
- Step 3: Search out the lane marking candidates
- Step 4: Decide the lane marking

Conventional Lane Detection Pipeline

- Step 1: Get the edge information.
  - Use for example Canny detector. OpenCV source code: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html





Canny Detector

1. Filter image to remove noise – Gaussian filter
2. Find the intensity gradient of the image
3. Non-maxima pixels suppression
4. Remove or leave each pixel based on hysteresis

Canny Detector

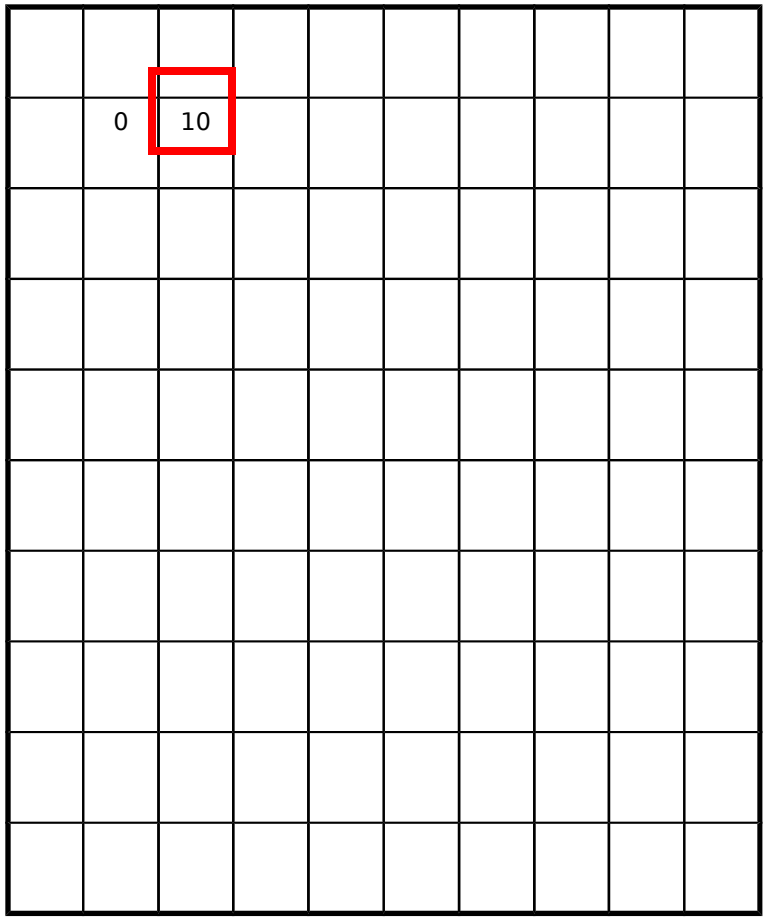
TU/e

1. Filter image to remove noise – Box Filter

$$g[\cdot,\cdot]\ \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

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



Canny Detector

TU/e

1. Filter image to remove noise – Box Filter

$$g[\cdot,\cdot]\ \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

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

	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	

Canny Detector

1. Filter image to remove noise – Guassian Filter

Gaussian coefficients

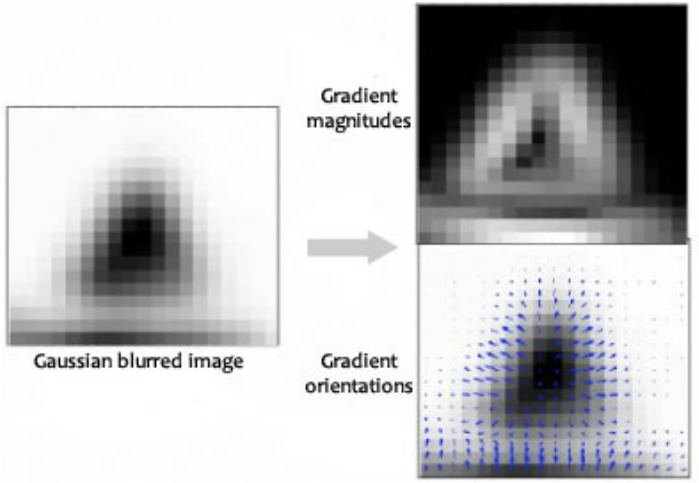
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

Mask 5 x 5

Instead of box filter coefficients

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

Canny Detector



2. Find the intensity gradient of the image

2.1. Apply a pair of convolution masks in X and Y directions:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

2.2. Find the gradient strength and direction with:

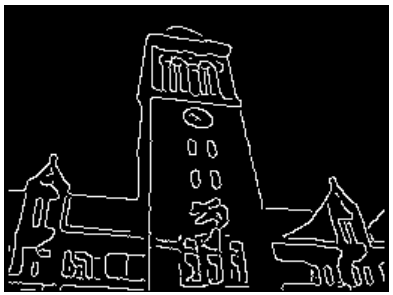
$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

3. Non-maxima pixels suppression:

reducing thickness of the edges

by leaving only pixels with highest gradient



Canny Detector

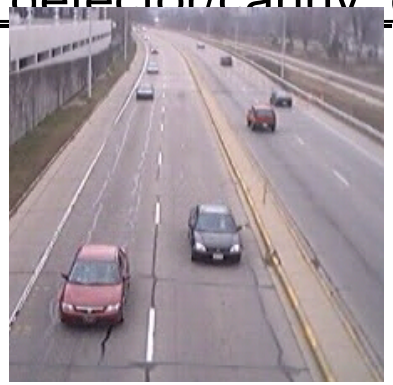
3. Remove or leave each pixel based on *hysteresis*

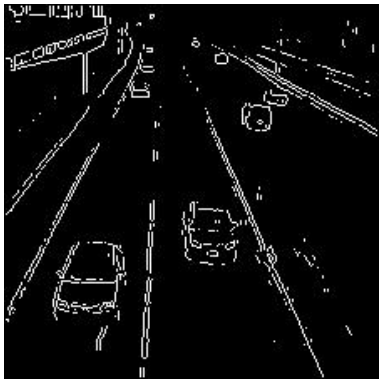
The final step. Canny uses two thresholds (upper and lower):

- If a pixel gradient is higher than the *upper* threshold, the pixel is accepted as an edge
- If a pixel gradient value is below the *lower* threshold, then it is rejected.
- If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the *upper* threshold.

Conventional Lane Detection Pipeline

- Step 1: Get the edge information.
  - Use for example Canny detector. OpenCV source code: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

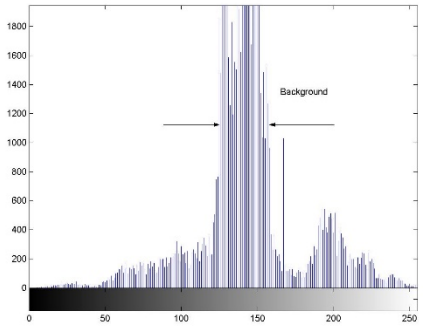




Lost many edges

Conventional Lane Detection Pipeline

- Step 0: Filter the image.
  - Option: background subtraction



Use global histogram to find the background gray range and subtract it from the original image

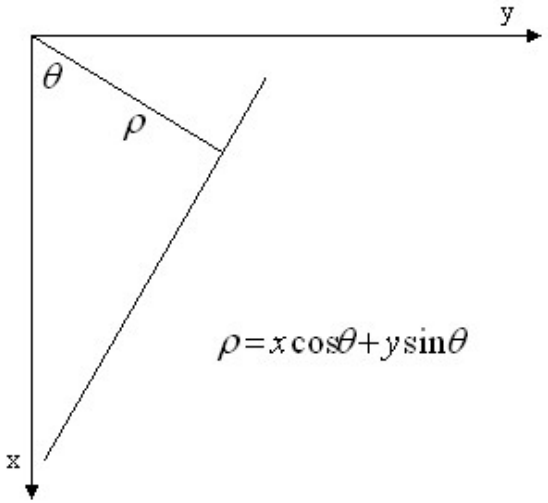
Compare these images

Edges are preserved using background subtraction method

Edge operation

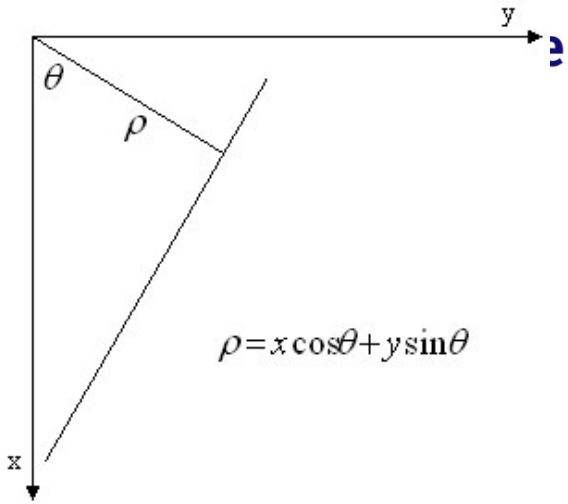
Conventional Lane Detection Pipeline

- Step 2: Hough Transform
  - technique to detect any shape, if you can represent that shape in mathematical form
  - line can be represented as $y = mx+c$ or

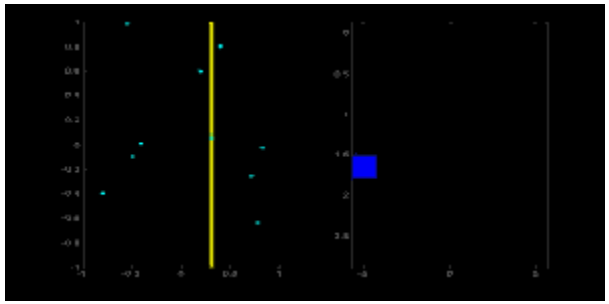


  - where *rho* is the perpendicular distance from origin to the line, and *theta* is the angle

Conventional Lane Detection Pipeline



- Step 2: Hough Transform

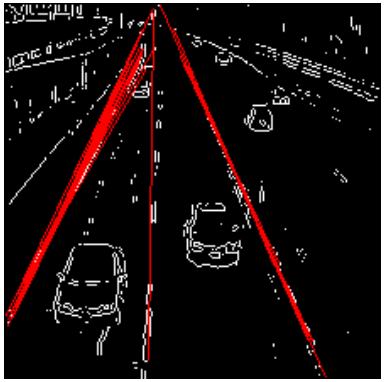


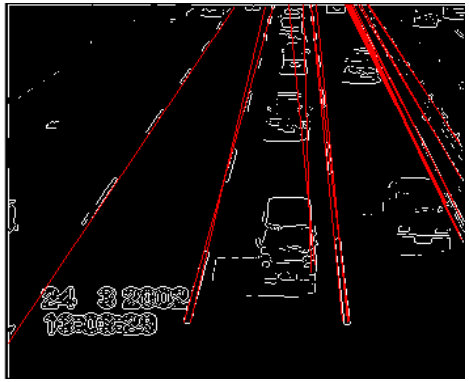
1. Hough Transform creates a 2D array [rho, theta] or accumulator (to hold values of two parameters) and it is set to 0 initially.
2. Take the first Canny pixel. You know its (x,y) values. Now in the line equation, put the values \theta = 0,1,2,....,180 and check the \rho you get. Store every (\rho, \theta) pair into 2D array.
3. Do the same for over pixels found by Canny detector.
4. Count equal elements (clustering) in 2D array – these counts show how many pixels belong to each hypothetical line
5. There will be a lot of non-clustered elements with 1 counts – no line. But some clusters will have from 2 to many counts – candidates for lines.

OpenCV source code: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

Conventional Lane Detection Pipeline

- Step 3: Search out the lane marking candidates.





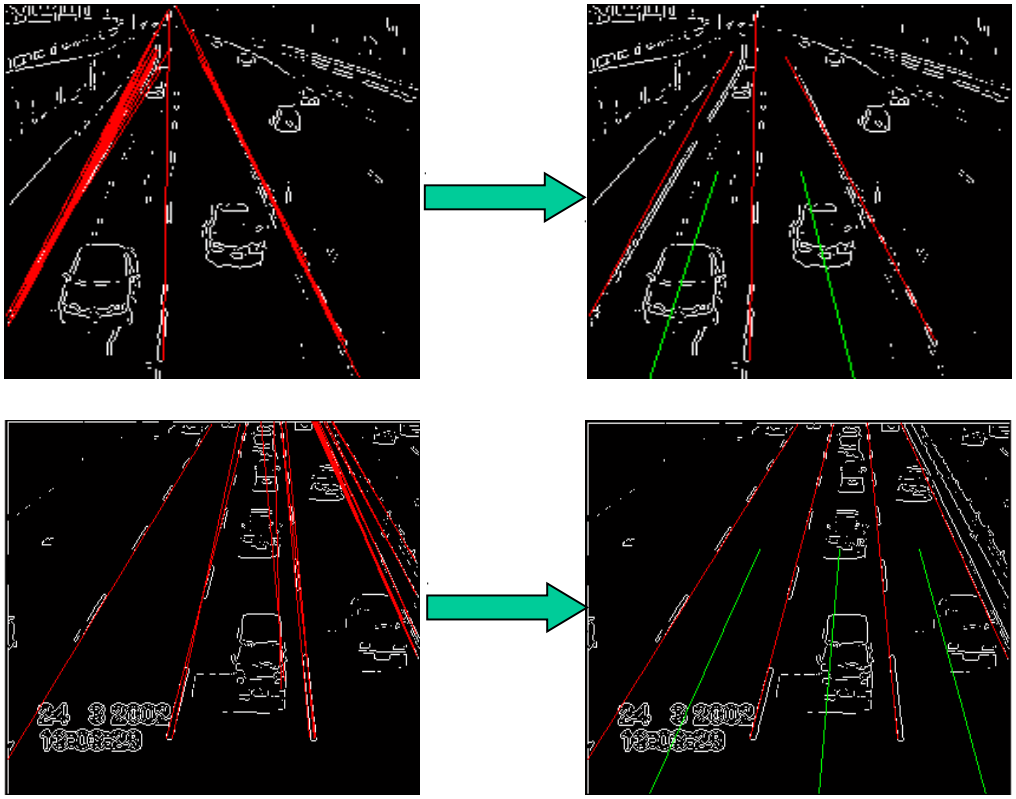
1. The red lines are the first 20 lines which have the biggest count numbers in Hough parameter space.

2. For each lane marking in the real scene, there are many line candidates around it.

3. There are some fake lines caused by the vehicle queue.

Conventional Lane Detection Pipeline

- Step 4: Decide on the lane marking



1. Sort the candidate lines by their position from left to right
2. Around each line cluster, choose the candidate which has the biggest count number as the lane marking in real scene
3. Delete the fake lane marking candidates
4. Calculate the mid-line of each lane (shown as green lines)

Lane Detection. Advanced

- Steps 5-100
  - You may want to project those lines to top view of the road scene.
  - Locate your vehicle on the this top view scene.
  - Make any smart decisions, based on found lanes and your vehicle location.

  - You may target more advanced cases (curved lanes, occlusions, bad road marking) by methods, like:
    - Deformable templates
    - Multi-resolution Hough Transform
    - B-snake
  - Some references:
    - Karl Kluge, Sridhar Lakshmanan, “A deformable-template approach to lane detection”,
    - Gonzalez, J.P.; Ozguner, U.; “Lane detection using histogram-based segmentation and decision trees”,
    - Yu, B.; Jain, A.K.; “Lane boundary detection using a multiresolution Hough transform”,
    - Yue Wang; Eam Khwang Teoh; Dinggang Shen; “Lane detection using B-snake”

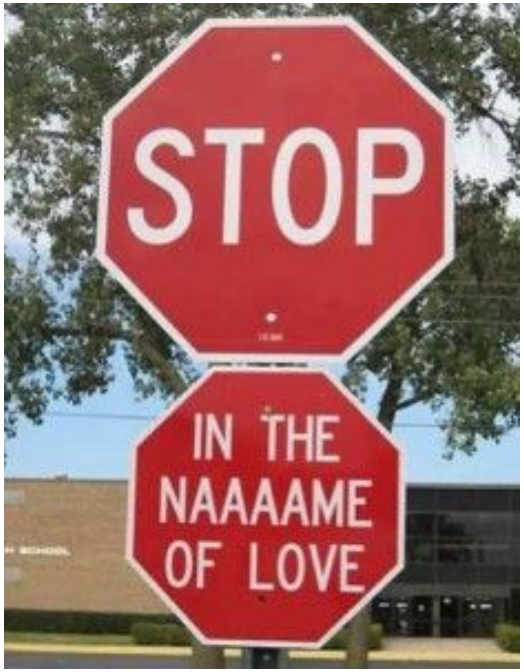
  - Apart from OpenCV, some other open source libraries for lane detection:
    - https://github.com/jdorweiler/lane-detection
    - http://www.vision.caltech.edu/malaa/software/research/caltech-lane-detection/
    - https://github.com/tomazas/opencv-lane-vehicle-track

TU/e

Specific Vision Tasks

Traffic Signs Detection

(As far as we know, photo is public domain)



Recap

- Computer Vision Introduction
- Vision for Autonomous Vehicles
- Lane Detection
  - Canny filter
  - Hough Transform
- Traffic Signs Detection
  - Next lecture