SENSOR FUSION & AUTONOMOUS DRIVE VEHICLE ALGORITHMS

SESSION FTF-AUT-N1782

DAVIDE SANTO, GLOBAL ADAS PRODUCT LINE MGR
LEONARDO SURICO, ADAS SYSTEM ARCHITECT
SESSION FTF-AUT-N1782
MAY 18, 2016
AGENDA

• Introduction to HAD and Automated Drive
• Challenges of Automated Drive
• Sensors and Perception
• Sensor Fusion and Bayesian Law
• Occupancy Grid Mapping
• Bayes Filter and Kalman Filter
• Mapping and Localization
• SLAM
• Autonomous Path Planning
• Deep Neural Network
Secure connected, self-driving cars will save >1.3M road fatalities globally

NXP offers a complete, secure ADAS system

<table>
<thead>
<tr>
<th>SENSE</th>
<th>THINK</th>
<th>ACT</th>
<th>BIG DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar Vision</td>
<td>Processing</td>
<td>Powertrain</td>
<td>Digital networking</td>
</tr>
<tr>
<td>Secure V2X</td>
<td>Sensor fusion</td>
<td>Chassis</td>
<td>Infrastructure</td>
</tr>
<tr>
<td></td>
<td>Security</td>
<td>Braking</td>
<td>Security</td>
</tr>
</tbody>
</table>

NXP: Enabling the Secure Connected Car

NXP: Enabling the Secure Connected Car
From Automotive…To Safe & Secure Mobility

SEAMLESS CONNECTED MOBILITY EXPERIENCE

ADVANCED DRIVER ASSISTANCE ➔ SELF-DRIVING

ENERGY EFFICIENCY

Enjoying Life
One hour per day in the car

Saving Lives
1.3M Road Fatalities Every Year

Reducing CO2
EU mandates 20% reduction by 2020
ADAS: Building Blocks of Autonomous Vehicles

- Adaptive Cruise control
- Automatic braking
- Lane keeping
- Partial automated parking
- Traffic Jam Assistance
- Emergency Brake with steer
- Automatic lane change
- Adaptive cruise control with lane centering
- Self parking
- Highway autopilot
- Parking garage pilot

Driver Assistance

- Responsibility for safe operation
- Control of vehicle speed
- Control of complete vehicle
- Control of steering

High Automation

Conditional Automation

Partial Automation
# ADAS Demands More Sensors and Compute Power

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>V2X</td>
<td>NO</td>
<td>SMALL SCALE</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>RADAR</td>
<td>0-3</td>
<td>1-5</td>
<td>3-6</td>
<td>4-8</td>
</tr>
<tr>
<td>CAMERA</td>
<td>0-1</td>
<td>3-5</td>
<td>3-6</td>
<td>3-6</td>
</tr>
<tr>
<td>LIDAR</td>
<td>0</td>
<td>0-2</td>
<td>0-3</td>
<td>2-4</td>
</tr>
<tr>
<td>ULTRASOUND</td>
<td>4-8</td>
<td>8-12</td>
<td>12</td>
<td>Complemented by radar</td>
</tr>
<tr>
<td>COMPUTE PERFORMANCE</td>
<td>1x</td>
<td>5x</td>
<td>20x</td>
<td>100x</td>
</tr>
</tbody>
</table>
The Most Complete ADAS Portfolio

SENSE

- V2X
- Radar
- Camera
- Lidar
- Ultrasonic

THINK

- Fusion

ACT

- Instrument cluster
  - Infotainment
- Vehicle Control
  - Engine control
  - Transmission
  - Brake
  - Steering
  - Airbag
  - Suspension

ADAS Example: Autonomous Emergency Braking

Vehicle State

- \( \mathbf{a}, \mathbf{\omega} \) Acceleration & Rotation
- \( \mathbf{\mathbf{v}} \) Speed

NXP Portfolio

- Sensor & RF
- Processor
- Analog & Power

#NXPFTF
Intelligent Machines

The term “intelligent” in the context of machines is defined by *Webster’s Dictionary* as “guided or controlled by a computer especially: using a built-in microprocessor for automatic operation, for processing of data, or for achieving greater versatility.”

Intelligent Road Vehicles

“A vehicle that performs driving tasks either autonomously or assisting the driver with the goal to make his/her driving task safer, more secure, more efficient and incrementally reducing environmental impacts.”
Intelligent Functions Classification (Michon 1985)

### Driving Task
- **Strategic**
  - Navigation
  - Driving style
- **Tactical**
  - Maneuvering
  - Lane change, parking
- **Operational Control**
  - Steer, Brake, Throttle

### Driving Environment
- **Motorway Traffic**
  - Easiest environment
  - Low driver load
- **Rural Traffic**
  - Mod dense network
  - Higher driver loads
- **Urban traffic**
  - High density
  - Very heavy driver load

### Support Level
- **Informing**
  - Factual/Suggestive
- **Supporting**
- **Automatic**
  - Overrulable
  - Non-overrulable
“Situational Awareness” (SA)
How Sensory Data Relates With Understanding Process

A dependable operation like HAD with attributes of robustness, fault tolerance, graceful degradation and fail-safes imply ‘extracting’ the world model or a sufficient and explicit representation of the real external world.

SA definition (Endsley 1995): the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future.

SA = State of Knowledge & Situation Assessment is the process that leads to it
SA Model in Dynamic Decision Making

Endsley Model (2000)

**Level 1** – Perception: lowest level, refers to acquiring cues (“features”) from raw data

**Level 2** – Comprehension: integration of the information (provided by the perception) takes place.

**Level 3** – Projection: The projection (i.e., prediction) from the current situation taking into consideration the dynamics of the observed phenomena and the anticipation of future events constitute the highest level of situational awareness
World Modeling and Representation

“The SA architecture resulted in information overload on the human side, thus constraining the performance. These issues profoundly surfaced first in the defense application domain. In order to facilitate development and interoperability the conceptualization of the information abstraction led to a commonly used data fusion model, the JDL model, which can be easily matched with situation assessment process.”

This model is today the basis for DATA FUSION

World Model: abstract I/F between Sensoric and Actuation

Improving the world Model enhances Control of the action

Goal is to achieve a dependable system
DATA Sensor Fusion

“Sensor Fusion”: A multi-level process dealing with the association, correlation, combination of data and information from single and multiple sources to achieve refined position, identify estimates and complete and timely assessments of situations, threats and their significance (JDL Workshop)

Three ‘Sub Functions’ of Sensor Fusion:
1. Data alignment (Features extraction)
2. Association (Clustering, Data prob)
3. Estimation (Bayes nets, Kal fil, Part filt, Neur nets, etc.)

5 Levels of Fusion:
Joint Director of Laboratories – DoD 1991

- **Lev 0 Source Pre-processing**: fusion at signal and pixel level. Information extraction process is done here
- **Lev 1 Object refinement**: leverage Lev 0 Data. Implements Spatial-temporal association, clustering, features association with images. Classification, Tracking
- **Lev 2 Situation Assessment**: identify likely solution based on Lev 1 data. Establish relations (proximity, comms). Identifies high-level inferences and patterns
- **Lev 3 Impact Assessment**: evaluates impacts of activities from Lev 2 and future perspectives/projections are evaluated (High End Processing)
- **Lev 4 Process Refinement**: provides resources and sensor management. Overlooks at reconfiguration of resources to guarantee functions
ADAS & HAD Connection

ADAS – Smart sensing
- Sensing
- Detection and Classification
- Tracking
- Motion Prediction

HAD – Intelligent processing
- Sense, classify & predict
- Localization & Mapping
- V2X Communication
- Path Finder and controller
AMP ADAS and HAD Portfolio

**Target Markets**

- Radar Based ADAS
  - Front L/M Range
  - Corner Radar
  - Highly Integrated Sensor

- Vision ADAS & Automated Driving
  - Mono/Stereo Vision
  - Surround Vision
  - Highly Automated Driving

**Products**

- **S32R**
  - Scalable, highly integrated, safe and secure family driving the digitalization of radar and sensor data fusion

- **S32V**
  - Automotive open platform built on quality, highly performing, fully abstracted accelerators, uncompromised safety and security

- **S32A**
  - General purpose computing with MASSIVE performance for environmental modeling acceleration, automotive quality, fault tolerance and security

**Technology**

- Software Dev Kit & Linux
- ARM Cortex Safe & Secure Architecture
- Best in Class Cognitive Acceleration
SENSOR FUSION AND AUTONOMOUS DRIVE VEHICLE ALGORITHMS
What Problems Do Autonomous Drive Vehicles Have to Solve?

Where am I? Localization
Where was I? Mapping
Where am I going? Path planning
How do I get there?

Will I hit anything? Obstacle detection
What will I hit? Classification
How can I avoid it? Path planning

Challenge: Distinction of static and moving obstacles
Is this object a parked vehicle or a vehicle waiting for precedence at intersection?
Autonomous Drive - Mobile Robot Systems

Localization & mapping

- Where am I?
- Where was I?

Map building

- Received / Local MAPs
- Traffic conditions
- GPS
- Compass
- Environment Model

Information extraction and interpretation

- Sensors

Perception

- What is moving?
- What is static?
- Will I hit anything?
- What will I hit?

Cognition & planning

- Where am I going?
- How do I get there?
- How can I avoid it?

Path planning

- CAR Dynamic Model
- Navigation Commands
- Drive Stile

Drive

- Execution
- actuators command

Motion

- Actuators

Real world

Knowledge database

V2X

Localized in the Map

Drive

Localization & mapping

- Map building
- Environment Model

Raw data

Information extraction and interpretation

Sensors

Localization & mapping

- Where am I?
- Where was I?

Map building

- Received / Local MAPs
- Traffic conditions
- GPS
- Compass

Environment Model

Information extraction and interpretation

- Sensors

Perception

- What is moving?
- What is static?
- Will I hit anything?
- What will I hit?

Cognition & planning

- Where am I going?
- How do I get there?
- How can I avoid it?

Path planning

- CAR Dynamic Model
- Navigation Commands
- Drive Stile

Drive

- Execution
- actuators command

Motion

- Actuators

Real world

Knowledge database

V2X

Localized in the Map

Drive

Localization & mapping

- Map building
- Environment Model

Raw data

Information extraction and interpretation

Sensors

Perception

- What is moving?
- What is static?
- Will I hit anything?
- What will I hit?
Perception

LIDAR
- 600K 3D/2D points/sec (2.5 Gbyte/sec)
- Range up to 100-200 meters, resolution 0.1 – 5 deg
- Field of View FOV up to 360 deg Horizontal, 0-30 deg Vertical
- Accuracy: 5 to 1.5 cm

Perception infuses cars with the intelligence they need to react to the environment (like people do).

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Sensor Type</th>
<th>Max. Range</th>
<th>Vertic. Angle</th>
<th>Horiz. Angle</th>
<th>Features used for tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>SICK LMS291</td>
<td>Scanning Laser, 1 level</td>
<td>80 m</td>
<td>0.25°</td>
<td>180°</td>
<td>Edge Target/2D coordinates of detection</td>
</tr>
<tr>
<td>IBEO AlascaXT</td>
<td>Scanning Laser, 4 level</td>
<td>200 m</td>
<td>3.2°</td>
<td>240°</td>
<td>Edge Target/2D coordinates of detection</td>
</tr>
<tr>
<td>Velodyne HDL-64E</td>
<td>Scanning Laser, 64 beams</td>
<td>120m</td>
<td>26.8°</td>
<td>360°</td>
<td>Edge Target/2D coordinates, height information target for validation</td>
</tr>
</tbody>
</table>
Perception

LIDAR
- 600K 3D/2D points/sec (2.5 Gbyte/sec)
- Range up to 100-200 meters, resolution 0.1 – 5 deg
- Field of View FOV up to 360 deg Horizontal
  0-30 deg Vertical
- Accuracy: 5 to 1.5 cm

Perception infuses cars with the intelligence they need to react to the environment (like people do).
Perception

LIDAR
- 600K 3D/2D points/sec (2.5 Gbyte/sec)
- Range up to 100-200 meters
- Field of View FOV up to 360 deg Horizontal, 0-30 deg Vertical
- Accuracy: 5 to 1.5 cm, 3 - 0.1 degree

RADAR
- 500 Kbyte/sec
- short range (30 - 70 meters)
- long range (70-200 meters)
- FOV V:5degree H: 60 degree
- accuracies 5cm, 0.5m/Sec (~18 Km/hr), 1-3 deg

targets
Perception

**LIDAR**
- 600K 3D/2D points/sec (2.5 Gbyte/sec)
- Range up to 100-200 meters
- Field of View FOV up to 360 deg Horizontal
  0-30 deg Vertical
- Accuracy: 5 to 1.5 cm, 3 - 0.1 degree

**RADAR**
- 500 Kbyte/sec
- short range (30 - 70 meters)
- long range (70-200 meters)
- FOV V:5degree H: 60 degree
- accuracies 5cm, 0.5m/Sec (~18 Km/hr), 1-3 deg
Perception

**LIDAR**
- 600K 3D/2D points/sec (2.5 Gbyte/sec)
- Range up to 100-200 meters
- Field of View FOV up to 360 deg Horizontal 0-30 deg Vertical
- Accuracy: 5 to 1.5 cm, 3 - 0.1 degree

**RADAR**
- 500 Kbyte/sec
- Short range (30 - 70 meters)
- Long range (70-200 meters)
- FOV V:5degree H: 60 degree
- Accuracies 5cm, 0.5m/Sec (~18 Km/hr), 1-3 deg

**CAMERA**
- 30/60/120 Megabyte/sec
- Range: 3 to 100 meters
- Field of View +/- 50
- Resolution depends on sensor/optics/distance (pix 4um, f: 5.7mm, d:70m ~50 mm)

Considering 30fps cameras

To recognize a pedestrian ~8 frames = 250mSec
Decision time (50mSec) so max latency breaking time is 300mSec

At 60km/h (~37mls/h) the car makes ~17m/Sec
So it will break in ~5.6m (45 fps is 4.4m)
Perception

LIDAR
• 600K 3D/2D points/sec (2.5 Gbyte/sec)
• Range up to 100-200 meters
• Field of View FOV up to 360 deg Horizontal
• Field of View FOV up to 0-30 deg Vertical
• Accuracy: 5 to 1.5 cm, 3 - 0.1 degree

RADAR
• 500 Kbyte/sec
• short range (30 - 70 meters)
• long range (70-200 meters)
• FOV V:5degree H: 60 degree
• accuracies 5cm, 0.5m/Sec (~18 Km/hr), 1-3 deg

CAMERA
• 30/60/120 Megabyte/sec
• Range: 3 to 100 meters
• Field of View +/- 50
• Resolution depends on sensor/optics/distance
  (pix 4um, f: 5.7mm, d:70m ~50 mm)

Max detectable range [m]

<table>
<thead>
<tr>
<th></th>
<th>60deg</th>
<th>80deg</th>
<th>100deg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle</td>
<td>160</td>
<td>110</td>
<td>75</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>Child</td>
<td>45</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Adult</td>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>Bicycle</td>
<td>90</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>Traffic Sign</td>
<td>60</td>
<td>35</td>
<td>25</td>
</tr>
<tr>
<td>Lane</td>
<td>80</td>
<td>70</td>
<td>50</td>
</tr>
</tbody>
</table>

Bottom of FOV
Perception

LIDAR
- 600K 3D/2D points/sec (2.5 Gbyte/sec)
- Range up to 100-200 meters
- Field of View FOV up to 360 deg Horizontal 0-30 deg Vertical
- Accuracy: 5 to 1.5 cm, 3 - 0.1 degree

RADAR
- 500 Kbyte/sec
- short range (30 - 70 meters)
- long range (70-200 meters)
- FOV V:5degree H: 60 degree
- accuracies 5cm, 0.5m/Sec (~18 Km/hr), 1-3 deg

CAMERA
- 30/60/120 Megabyte/sec
- Range: 3 to 100 meters
- Field of View +/- 50
- Resolution depends on sensor/optics/distance (pix 4um, f: 5.7mm, d:70m ~50 mm)
## Sensors for Perception – Why Do Our Sensors Need Fusion?

<table>
<thead>
<tr>
<th><strong>LIDAR</strong></th>
<th><strong>CAMERA</strong></th>
<th><strong>RADAR</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• 600K 3D/2D points/sec (2.5 Gbyte/sec)</td>
<td>• Inexpensive</td>
<td>• Expensive (today)</td>
</tr>
<tr>
<td>• Range up to 100-200 meters</td>
<td>• Works by received lights (recognize color and pattern)</td>
<td>• In hot conditions, Lidar doesn’t distinguish between a big dust cloud and a brick wall</td>
</tr>
<tr>
<td>• Field of View FOV up to 360 deg Horizontal 0-30 deg Vertical</td>
<td>• Has good accuracy in lateral measurement</td>
<td>• Limited information about texture or color of surfaces</td>
</tr>
<tr>
<td>• Accuracy: 5 to 1.5 cm, 3 - 0.1 degree</td>
<td>• Has good accuracy in lateral measurement</td>
<td>• Challenged for identifying non metallic objects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• No information about texture or color (no TL, no TSR)</td>
</tr>
</tbody>
</table>

**LIDAR**
- Is virtually immune to problematic visibility and lighting conditions
- Has a good accuracy in longitudinal distance measurement
- return speed and distance
- Challenged by lights conditions
- Challenged by weather conditions
- Need Multiple-camera systems to provides reliable 3D information

**CAMERA**
- 30/60/120 Megabyte/sec
- Range: 3 to 100 meters
- Field of View +/- 50
- Resolution depends on sensor/optics/distance (pix 3.75um, f: 5.7mm, d:70m ~50 mm)

**RADAR**
- 500 Kbyte/sec
- short range (30 - 70 meters)
- long range (70-200 meters)
- FOV V:5degree H: 60 degree
- accuracies 5cm, 0.5m/Sec (~18 Km/hr), 1-3 deg
- Challenged by lights conditions
- Challenged by weather conditions
- Need Multiple-camera systems to provides reliable 3D information
Challenging Scene (From Heidelberg Database)

Driving requires that the system "never miss" anything that is in a position to be a safety concern

Why Sensor Fusion?

Environment Model

- Where am I?
- Where was I?
- Localized and mapping
- Map Building
- Global / Local Map position

Perception
- Information Extraction and Interpretation
- Sensors
- Raw data

Cognition and planning
- Path Planning
- Where am I going?
- How do I get there?
- How can I avoid it?

Path Planning
- Drive
- Execution
- Actuators
- actuators command

Motion

- What is moving?
- What is static?
- Will I hit anything?
- What will I hit?
Why Sensor Fusion?

Moving toward autonomous driving, no one sensor by itself is enough to guarantee the perception system 100% reliable in all conditions.

**False negatives (Blindness)**

- That can be catastrophic if it happens for long enough so that you might be unable to safely avoid hitting the obstacle
- A good system will almost never (one in many millions) get a false negative
- It may occasionally take a little bit of extra time to fully understand an obstacle, and one may even blip out for brief flashes, but a persistent failure can mean a crash

Related to the above is a misclassification. This can mean mistaking a cyclist for a pedestrian, or mistaking two motorcycles for a car. Even without identification, you know not to hit the obstacle, but you might incorrectly predict where it is going or how best to react to it.

**False Positive (Ghost object)**

- The system sees an obstacle that isn't really there
- This will cause the vehicle to brake or swerve
- This is annoying to the occupants
- And it can also cause accidents if the vehicle is being followed too closely or swerves dangerously or brakes too hard
- Car makers will recall cars

The solution is diversity and redundancy; you will see some combination of Lidar, Radar, and high-resolution video cameras.

We need to tailor the sensors to the problem they are supposed to solve, and to the criticality.
Why Sensor Fusion?

- The consensus is that sophisticated sensor fusion will be necessary to forge, from several different kinds of perception data, an accurate account of the vehicle’s surroundings: the objects, their velocities and accelerations, and their probable behavior.

- How much processing occurs where in the system?
  - Concentrated solution
  - Distributed solution
Sensor Fusion

Concentrated solution

Feed all the raw data into a fusion box.
- Impressive amount of performances required (DMIPS)
- Difficult to partition task btw various parallel engine (CPU, GPU, Vectors engine)
- Extremely high concentrated ECU power requirements.

Distributed solution

Build simpler systems to process the sensor data, and then fuse the preprocessed data at a higher level of abstraction.
- Keep it simple is a logical sequence of processing steps
- Each sensor would do local signal conditioning
- Then each sensor would locally do object estimation based on its particular expertise, and attach attributes to these putative objects returning metadata to the fusion box.
Sensor Fusion: How is It Achieved?

Where am I?  Where I was?
Localization and mapping
Map building

Global / Local Map position
Path planning
Cognition and planning

Environment Model
Information Extraction and Interpretation
Sensors

Where am I going? How do I get there? How can I avoid it?

What is moving? What is static? Will I hit anything? What will I hit?

Execution
Actuators

Motion

Drive
Uncertainty in Robotics and Probabilistic Robotic

**Environments.** Physical worlds are inherently unpredictable. While the degree of uncertainty in well-structured environments such as assembly lines is small, environments such as roads and private homes are highly dynamic and unpredictable.

**Sensors.** Sensors are inherently limited in what they can perceive. Limitations arise from two primary factors.
1) range and resolution of a sensor is subject to physical laws
2) sensors are subject to noise, which perturbs sensor measurements in unpredictable ways and hence limits the information that can be extracted from sensor measurements.

**Models.** Models are inherently inaccurate. Models are abstractions of the real world. As such, they only partially model the underlying physical processes of the robot and its environment.

**Computation.** Robots are real-time systems, which limits the amount of computation that can be carried out. Many state-of-the-art algorithms are approximate, achieving timely response through sacrificing accuracy.
Deterministic and Stochastic Models

A deterministic system no randomness is involved in the development of future states of the system.

A deterministic model always produce the same output from a given starting condition or initial state.

A stochastic process is a collection of random variables, representing the evolution of some system of random values over time. This is the probabilistic counterpart to a deterministic system. Instead of describing a process which can only evolve in one way (as in the case, for example, of solutions of an ordinary differential equation), in a stochastic or random process there is some indeterminacy: even if the initial condition (or starting point) is known, there are several (often infinitely many) directions in which the process may evolve.

A Markov chain is a random process that undergoes transitions from one state to another on a state space. It must possess a property that is usually characterized as "memoryless": the probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it. This specific kind of "memorylessness" is called the Markov property. Markov chains have many applications as statistical models of real-world processes.

Markov Assumption
Future is Independent of Past Given Current State
Assume Static World
Introduction to Bayesian Law

A technique for statistical dependencies between stochastic variables and also works as a probabilistic estimation machine.

<table>
<thead>
<tr>
<th>p(C=vehicle)</th>
<th>p(C= pedestrian)</th>
</tr>
</thead>
<tbody>
<tr>
<td>70%</td>
<td>30%</td>
</tr>
</tbody>
</table>

| given C | p(R=strong |C) | p(R=weak |C) |
|---------|---------|---------|
| vehicle| 60%     | 40%     |
| pedestrian| 20%   | 80%     |

| C = veh | p(R=strong |C) | p(R=weak |C) |
|---------|---------|---------|
| vehicle| True positive | False Positive |
| pedestrian| False Negative | True Negative |

**I make a measure and I get R=strong**

**How Accurate Is my measure? (Real probability to have a vehicle)**

How can I combine the probabilities?
How can I calculate the estimation?

The chances of a *true positive* = chance you have a vehicle * chance radar caught the vehicle = 70% * 60% = 42%

The chances of a *false positive* = chance you don’t have a vehicle * chance radar caught the vehicle anyway = 30% * 40% = 12%

| p(R=strong |C) | p(R=weak |C) |
|---------|---------|---------|
| vehicle| 42%     | 12%     |
| pedestrian| 6%    | 24%     |
Introduction to Bayesian Law

How I combine the probabilities? How I calculate the estimation?

<table>
<thead>
<tr>
<th>C</th>
<th>( p(R=\text{strong} \mid C) )</th>
<th>( p(R=\text{weak} \mid C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle</td>
<td>42%</td>
<td>12%</td>
</tr>
<tr>
<td>pedestrian</td>
<td>6%</td>
<td>24%</td>
</tr>
</tbody>
</table>

Probability = desired event / all possibilities
= \( P(R=\text{strong}) \) / \( P(\text{All possibility}) \) = 42% / (42% + 12%) = 78%

\[
\Pr(A \mid X) = \frac{\Pr(X \mid A) \Pr(A)}{\Pr(X)}
\]

\[
\Pr(C=\text{Vehicle} \mid R=\text{Strong}) = \frac{\Pr(R=\text{Strong} \mid C=\text{Vehicle})}{\Pr(R=\text{Strong} \mid C=\text{Vehicle}) + \Pr(R=\text{Weak} \mid C=\text{Vehicle})} = 78\%
\]

\[
\Pr(C=\text{Vehicle} \mid R=\text{Weak}) = \frac{12\%}{12\% + 24\%} = 22\%
\]

\[
\Pr(C=\text{Pedestrian} \mid R=\text{Weak}) = \frac{24\%}{6\% + 24\%} = 80\%
\]

\[
\Pr(C=\text{Pedestrian} \mid R=\text{Strong}) = \frac{6\%}{6\% + 24\%} = 20\%
\]
Stochastic Models - Simple Example

The radar sensor returns a measure $R$ (reflection) = weak.

The camera sensor classified the object as $W = $ pedestrian

What object is that? (I want to classify the object $C$)

I have databases built from historical knowledge

<table>
<thead>
<tr>
<th>$p(C=\text{vehicle})$</th>
<th>$p(C=\text{pedestrian})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>70%</td>
<td>30%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>given $C$</th>
<th>$p(R=\text{strong} \mid C)$</th>
<th>$p(R=\text{weak} \mid C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle</td>
<td>60%</td>
<td>40%</td>
</tr>
<tr>
<td>pedestrian</td>
<td>20%</td>
<td>80%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p(W=\text{wide} \mid C)$</th>
<th>$p(W=\text{narrow} \mid C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>given $C$</td>
<td></td>
</tr>
<tr>
<td>vehicle</td>
<td>99%</td>
</tr>
<tr>
<td>pedestrian</td>
<td>1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p(R=\text{strong} \mid C)$</th>
<th>$p(R=\text{weak} \mid C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>given $C$</td>
<td></td>
</tr>
<tr>
<td>vehicle</td>
<td>42%</td>
</tr>
<tr>
<td>pedestrian</td>
<td>6%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p(W=\text{wide} \mid C)$</th>
<th>$p(W=\text{narrow} \mid C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>given $C$</td>
<td></td>
</tr>
<tr>
<td>vehicle</td>
<td>69.3%</td>
</tr>
<tr>
<td>pedestrian</td>
<td>0.3%</td>
</tr>
</tbody>
</table>
Bayesian Network

A technique for statistical dependencies between stochastic variables and also works as a probabilistic estimation machine.

Object (?)

Classification

Radio reflection intensity (Radar)

Camera classification

How I combine the probabilities?

How I calculate the estimation?

\[ \Pr(A|X) = \frac{\Pr(X|A) \Pr(A)}{\Pr(X)} \]

\[ \Pr(C=\text{pedestrian} \mid R=\text{weak} \& W=\text{narrow}) = \]

\[ \Pr(R=\text{weak} \& W=\text{narrow} \mid C=\text{pedestrian}) \cdot \Pr(C=\text{pedestrian}) \]

\[ \Pr(R=\text{weak} \& W=\text{narrow} \mid C=\text{pedestrian}) \cdot \Pr(C=\text{pedestrian}) + \Pr(R=\text{weak} \& W=\text{narrow} \mid C=\text{vehicle}) \cdot \Pr(C=\text{vehicle}) \]

\[ \frac{(80\% \cdot 99\%) \cdot 30\%}{(80\% \cdot 99\%) \cdot 30\% + (40\% \cdot 1\%) \cdot 30\%} = 99.5\% \]
Bayesian Stochastic Model

- If a sensor’s performance changes, only the corresponding node’s CPT has to be changed and other CPTs or model structure can be left untouched.

- To treat continuous variables such as length or position, probability density functions (PDF) can be assigned to these probabilities. Instead of CPT, conditional probability density function (CPDF) is used.

\[
P(X|X_a=a, X_b=b) = \frac{p(X) \cdot p(X_a=a|X) \cdot p(X_b=b|X)}{\int_X P(X, X_a=a, X_b=b) dX}
\]

\[
= \frac{1}{Z} p(X_a=a|X) \cdot p(X_b=b|X)
\]
Sensors Stochastic Output

LIDAR

RADAR

CAMERA
Perception in 3D (with Mapping)
Occupancy Grid Mapping

Known poses of the car
Occupancy Grid

We will apply Bayes Law

\[ p(A \mid B) = \frac{p(B \mid A) \cdot p(A)}{p(B)} \]

- where \( A \) is \( \text{occ}(i,j) \)
- and \( B \) is an observation \( r=D \)

\[
\text{Grid\_new}(x,y) = \frac{\text{M}(x,y) \cdot \text{Grid}(x,y)}{\text{M}(x,y) \cdot \text{Grid}(x,y) + (1-\text{M}(x,y)) \cdot (1-\text{Grid}(x,y))}
\]
There can be multiple grid maps
- Static objects
- Moving objects
- Empty space
Occupancy Grid Mapping

Discrete 3D grids
Bayes Filters

- Is a mathematical probabilistic approach for estimating a PDF recursively over time using incoming measurements.

- Bayes filters allow robots to continuously update their most likely position within a coordinate system, based on the most recently acquired sensor data.

- The recursive algorithm consists of two parts: prediction and innovation.
- If the variables are linear and normally distributed the Bayes filter becomes the Kalman filter.
Bayes Filters

The confidence in the knowledge of the position of the car has decreased due to uncertainty of the action (odometer).

Estimated believe or Prior : \( b_{\text{est}}(1) \)

The prior, of an uncertain quantity is the PDF that would express one's beliefs about this quantity before some measure is taken into account.
Bayes Filters

This is the probability of observing $z$ given that the robot is at position $x$, that is: $p(z_t | x_t)$

- State: $x(1)$
- Prior: $b_{est}(1)$

Measurement (noisy) $z(1)$
Bayes Filters

The posterior is the PDF that is assigned after the measurement.
Bayes Filter Algorithm

\[ p(x_t | x_{t-1}, u_t) \]

\[ x_{t-1} \quad x_t \quad u_t \]

\[ \text{bel}_{t-1} \quad \text{bel}_t \]
Bayes Filter Algorithm

The state transition distribution characterizes how state changes over time, possibly as the effect of robot controls. The measurement distribution characterizes how measurements are governed by states. Both laws are probabilistic.
Perception Update

This is the probability of observing $z$ given that the robot is at position $x$, that is: $p(z_t | x_t)$

$$p(x_t | z_t) = \frac{p(z_t | x_t)p(x_t)}{p(z_t)}$$
Bayes Filter Algorithm

Input:
- \( \text{bel}(x_{t-1}) \): previous machine status (belief)
- \( u_t \): recent control action to change the state (motion)
- \( z_t \): recent measurements (perception)

1: \( \text{Algorithm Bayes filter}(\text{bel}(x_{t-1}), u_t, z_t): \)
2: \hspace{1em} for all \( x_t \) do
3: \hspace{2em} \( \text{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \text{bel}(x_{t-1}) \, dx \)
4: \hspace{2em} \( \text{bel}(x_t) = \eta p(z_t \mid x_t) \text{bel}(x_t) \)
5: \hspace{1em} endfor
6: \hspace{1em} return \( \text{bel}(x_t) \)

Motion prediction
Process the motion \( u_t \) that transition from \( x_{t-1} \) to \( x_t \)

Measurement update
Process the motion \( u_t \) that transition from \( x_{t-1} \) to \( x_t \)

Return the new believe

Thrun, Sebastian; Burgard, Wolfram; Fox, Dieter (2005-08-19). Probabilistic Robotics
Kalman Filter

The state transition probability $p(x_t \mid u_t, x_{t-1})$ must be a linear function in its arguments with added Gaussian noise.

$$
x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t
$$

$$
z_t = C_t x_t + \delta_t
$$

$$
\mu_t = A_t \mu_{t-1} + B_t u_t
$$

$$
\Sigma_t = A_t \Sigma_{t-1} A_t^T + R_t
$$

Kalman Filter

- State transition are linear
- Unimodal distribution (single max)

Gaussians are represented by their moments (mean, covariance)
The Kalman Filter Algorithm for Linear Gaussian State Transitions and Measurements.

```
1:  Algorithm Bayes_filter(\text{bel}(x_{t-1}), u_t, z_t):
2:       for all \(x_t\) do
3:           \text{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \text{bel}(x_{t-1}) \, dx
4:       \text{bel}(x_t) = \eta \cdot p(z_t \mid x_t) \cdot \text{bel}(x_t)
5:   endfor
6:  return \text{bel}(x_t)
```

```
1:  Algorithm Kalman_filter(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t):
2:     \tilde{\mu}_t = A_t \mu_{t-1} + B_t u_t
3:     \tilde{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t
4:     K_t = \Sigma_t C_t^T (C_t \Sigma_t C_t^T + Q_t)^{-1}
5:     \mu_t = \tilde{\mu}_t + K_t (z_t - C_t \tilde{\mu}_t)
6:     \Sigma_t = (I - K_t C_t) \Sigma_t
7:  return \mu_t, \Sigma_t
```
Kalman Filter Example

Line fitted through one set of candidate points and parameterized with $\rho$ and $\theta$.

The state vector $x(n)$

$$x(n) = \begin{bmatrix} \rho(n) \\ \dot{\rho}(n) \\ \theta(n) \\ \dot{\theta}(n) \end{bmatrix}^T$$

$$x_t = A_t x_{t-1} + B u_t$$

$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Lane Keeping Warning
Steering model

Lane Keep Assistant
Steering model
# Brief Table of Most Common Bayes Filters Algorithms for Robotics

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Class</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalman Filter &amp; Extended Kalman</td>
<td>Parametric – m, S Gaussian Filters</td>
<td>Well known. Easy to model Motion Update Execute in Polynomial time</td>
<td>Needs to Invert Matrix and uses Jacobian of parameters (tough already with 10-15 states)</td>
</tr>
<tr>
<td>Kalman Filter</td>
<td>Parametric – x, W Gaussian Filters</td>
<td>Easy to incorporate multi-sensors measurement (via Logarithmic Odds form). Execute in Polynomial time</td>
<td>Less Popular than EKF. Needs log/exp to recover original distribution</td>
</tr>
<tr>
<td>Information Filter (log-odd version)</td>
<td>Parametric - x, W Gaussian Filters</td>
<td>Can model multimodal distribution – Easier to compute</td>
<td>Exponential time of execution.</td>
</tr>
<tr>
<td>Histogram Filters</td>
<td>Non-parametric. Split distribution in regions with constant values</td>
<td>Extremely easy to implement and can be adapted to all distribution</td>
<td>Needs large amount of samples (eg.1000) ⇒ Computationally intense</td>
</tr>
<tr>
<td>Particle Filters</td>
<td>Montecarlo Methods – Samples from Posterior and associate Weight</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Autonomous Drive - Mobile Robot Systems

![Diagram of Autonomous Drive System]

- **Map building**
  - Where am I?
  - Where was I?
  - Localization & mapping
  - Environment Model
  - V2X

- **Information extraction and interpretation**
  - Raw data
  - Sensors

- **Cognition & planning**
  - Path planning
  - Where am I going?
  - How do I get there?
  - How can I avoid it?

- **Execution**
  - Actuators command

- **Real world**
  - Knowledge database
  - Drive Stile
  - CAR Dynamic Model
  - Navigation Commands

- **Actuators**
  - Drive

- **Motion**
  - Actuators

- **Perception**
  - Where am I?
  - Where was I?
  - Localization & mapping
  - Environment Model
  - V2X

- **Localization & mapping**
  - Received / Local MAPs
  - Traffic conditions
  - GPS
  - Compass

- **Sensors**
  - What is moving?
  - What is static?
  - Will I hit anything?
  - What will I hit?
How to Improve Perception

DSRC (Dedicated Short Range Communications) is a two-way short- to- medium-range wireless communications capability that permits very high data transmission critical in communications-based active safety applications.

- A heavily dynamic view of things like lane closures, slippery roads, construction at much higher precision that what you are used to with today's traffic information. Processes of real time sensor data integrated with real time traffic information, weather and even road conditions are required

- The ability to pull information from various sources beyond a car's own sensors is crucial to advancing car technology toward automated driving
V2X

- “Seeing around corners”
- 360° 2-way communication
- Long distance (up to ~1-2km)
- Reliable – all-weather
- Ad-hoc network
- No latency & infrastructure!
- US Legislation for crash avoidance & safety

Driving Use Cases:
802.11p
- Intersection assist
- Traffic light info/speed advisory
- Non-line-of-sight ADAS

Non-Driving Cases:
802.11n/ac + BT-LE
- Car2Home
- 11n (11ac) WiFi
- Piloted Parking
- Lock / Unlock

NXP SUPPLIES V2X CHIPSETS INCL. SOFTWARE AND CHIPS FOR DATA SECURITY
Mapping and Localization

MAPPING: What is the world around me?
- Sense from various positions
- Integrate measurements to produce map
- Assumes perfect knowledge of position

LOCALIZATION: Where am I in the world?
- Sense
- Relate sensor readings to a world model
- Compute location relative to model
- Assumes a perfect world model

Where am I?
Mapping

Perception of Nearby Environment

- Humans don’t need maps to drive cars
- Nevertheless, the sensors available in the car gives a perception range less of 150 meters, that means at 100Km/h I have about 5 seconds of sensing horizon
- In a driverless car there are particular maneuvers that are behind the sensors capabilities
- The answer is to provide a rich map of the road with all the details required
Mapping

3D representation of the street environment with accuracy processed to a level of 10-20cm, adequate for the purposes of highly automated driving.

Only ground plane part of 3D map is kept for localization.

Since these ground “images” are not grabbed with normal cameras, some typical pitfalls and problems of imaging are avoided.
## Mapping

<table>
<thead>
<tr>
<th>Stored Maps</th>
<th>HD maps</th>
<th>Sparse recording maps</th>
<th>No maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-drive record of 360deg surround 3D maps</td>
<td>10 cm</td>
<td>10 cm feature based</td>
<td>Perception Only</td>
</tr>
</tbody>
</table>

### Accuracy:
- <10cm
- 10 cm
- 10 cm feature based
- Perception Only

### Byte/Km:
- 1 GByte
- 100MB
- 100KB
- 0

### Complexity of Perception
Measurement Equation and Maps Types

Relation between map and measurement $h()$ is known as Measurement equation:

$$z_k^{(i)} = h(m_k^{(j)}) + e_k,$$

which is sensor technology dependent.

<table>
<thead>
<tr>
<th>Map types</th>
<th>Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature based</td>
<td>Salient Features in the scene are represented (ex: radar map)</td>
</tr>
<tr>
<td>Road Map (as special case of Feature Map)</td>
<td>It models road geometry (lines, curves and clothoids)</td>
</tr>
<tr>
<td>Location based maps</td>
<td>Occupancy Grid</td>
</tr>
<tr>
<td>Intensity based map</td>
<td>Continuous approximation of location based.</td>
</tr>
</tbody>
</table>
Localization

Autonomous cars must also know where they are so they can maneuver in time (to take a given exit at the highway, respect lanes in one-way streets, etc.).

- The car must stay localized against some kind of available map
- Localization should be **efficient, robust** and **in real-time**

Localization is feasible only after building a map of the environment (SLAM), which is then used as a reference for localization (assuming the environment will not change too much).

Firstly, a simplified model is addressed: by just using GPS (2m accuracy) and inertial data, we have a first estimation of the car localization.

Then, the prior maps of the ground plane are matched with the perception map. The one that more accurately match the actual data win; the rest are ignored.
Localization via Occupancy Grid Map

I have available:
- Occupancy Grid
- Car Pose
- Static Map \( m \)

Compute the most likely map:

\[ M = \max P(m | z_1, z_2, \ldots, z_k, x_1, \ldots, x_n) \]

\( p(m) \) has uniform probability

\[ = \arg \max \prod_{t=1}^{T} P( z_t | m, x_t ) \]

And with the map I have now constrain

\[ p(x_t | u_t, x_{t-1}) \neq p(x_t | u_t, x_{t-1}, M) \]
Localization Via Occupancy Grid Map
Features-based Localization – the Radar and Lidar Case

Features map are based on landmarks and they are used together with GPS map to help the path-finding task.

If Radar and Lidar measures are:

\[ z^{(i)} = [r^{(i)} \psi^{(i)}]^T \]

If ego position is:

\[ x_E = [x_E \ y_E \ \psi_E]^T \]

and landmark position is:

\[ m^{(j)} = [x_m^{(j)} \ y_m^{(j)}]^T. \]

Then the measurement equation is:

\[ z_k^{(i)} = h(m_k^{(j)}) + e_k, \quad \begin{bmatrix} r^{(i)} \\ \psi^{(i)} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_m^{(j)} + x_E)^2 + (y_m^{(j)} + y_E)^2} \\ \arctan \left( \frac{y_m^{(j)} - y_E}{x_m^{(j)} + x_E} \right) - \psi_E \end{bmatrix} + \begin{bmatrix} e_r \\ e_{\psi} \end{bmatrix} \]
Understand SLAM

The robot is represented by the triangle. The stars represent landmarks. The robot initially measures using its sensors the location of the landmarks (sensor measurements illustrated with lightning).
Understand SLAM

The car moves so it now thinks it is here (prior).
The car once again measures the location of the landmarks using its sensors but finds out they don’t match with where the robot thinks they should be (given the robot’s location). Thus the robot is not where it thinks it is.
Understand SLAM

Car build the posterior (the location the robot originally thought it was at is illustrated by the dashed triangle).
Understand SLAM

In actual fact the car is here.
SLAM

- Robot **pose** at time $t : x_t \Rightarrow$ Robot **path** up to this time: $\{x_0, x_1, \ldots, x_t\}$

- Robot **motion** between time $t-1$ and $t : u_t$ (control inputs/proprioceptive sensor readings)
  $\Rightarrow$ Sequence of robot relative motions: $\{u_0, u_1, \ldots, u_t\}$

- The **true map** of the environment: $\{y_0, y_1, \ldots, y_N\}$

- At each time $t$ the robot makes measurements $z_i$
  $\Rightarrow$ Set of all measurements (observations): $\{z_0, z_1, \ldots, z_k\}$

- The Full SLAM problem: estimate the posterior
  $p(x_{0:t}, y_{0:n} | z_{0:k}, u_{0:t})$

- The Online SLAM problem: estimate the posterior
  $p(x_t, y_{0:n} | z_{0:k}, u_{0:t})$
Essential Components for SLAM

1. Robust local motion estimation

2. Mapping & loop-closure detection

3. Map management & optimisation
Autonomous Drive - Mobile Robot Systems

**Localization & mapping**
- **Where am I?**
- **Where was I?**

**Environment Model**
- V2X
- Knowledge database
- Local Map
- Received / Local MAPs
- Traffic conditions
- GPS
- Compass

**Information extraction and interpretation**
- Raw data
- Sensors

**Map building**
- Localized in the Map

**Cognition & planning**
- **Where am I going?**
- **How do I get there?**
- **How can I avoid it?**

**Path planning**
- Drive
- Navigation Commands
- Drive Stile

**Motion**
- Execution
- actuators command
- Actuators

**Perception**
- Real world
- What is moving?
- What is static?
- Will I hit anything?
- What will I hit?

**Drive**
- CAR Dynamic Model
- GPS
- Compass
- Localization & mapping

**Navigation Commands**
- Received / Local MAPs
- Traffic conditions
- GPS
- Compass

**Environment Model**
- V2X
- Knowledge database
- Local Map
- Received / Local MAPs
- Traffic conditions
- GPS
- Compass

**Information extraction and interpretation**
- Raw data
- Sensors

**Map building**
- Localized in the Map
An autonomous car needs to calculate path planning (navigate):
• Carrying out locomotion primitives to move between points (closed node)
• Avoiding obstacles
• Respecting the traffic rules
Comfortable Autonomous Drive Style

Theoretically, a car could drive itself at a speed that feels too fast and still be perfectly safe – but it may not necessarily make passengers feel relaxed or comfortable.

By analyzing long-term driver behavior from non-automated cars and understanding how humans drive, it is possible to make the autonomous driving experience more familiar and comfortable.

Factor that affects drive style:
- Road geometry and width
- Oncoming traffic
- Oncoming traffic at night
- Weather conditions
- The condition of the road surface
- Roadside objects such as buildings or trees obstructing the view through bends
- Social and cultural driving norms
- And personal preferences
# Neural Network

## ImageNet

**22K categories and 14M images**

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
  - Food
  - Materials
- Structures
  - Artifact
  - Tools
  - Appliances
  - Structures
- Person
  - Scenes
    - Indoor
    - Geological Formations
    - Sport Activities

*Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009*
And the Winner Is: Deep Learning (Especially Convolutional NN)
Automotive Example - Scene Labeling

• Scene labeling consists in labeling each pixel in an image with the category of the object it belongs to
• Deep networks outperform graphical models on automotive benchmark suites for scene labeling
• Famous Yann LeCun paper combines CNNs, super-pixel segmentation and CRFs (conditional random fields) and achieves impressive performance

https://www.youtube.com/watch?v=ZJMtDRbgH40

Key task for Self Driving
Deep Learning: How Does It Work

4 (2 Hidden) Layers Net

The Parameters count is:

$$|\theta| = 3 \cdot K^2 + K$$

For a 32 x 32 pix picture

$$|\theta| = 3 \cdot 32^4 + 32^2 \approx 3 \times 10^6$$

1. Hard to Train
2. Hard to Initialize
3. Heavy Cost of HW
4. Local Minima
5. “Vanishing Cost”
6. Difficult to debug

Shallow Networks Are insufficient To represent Learning complexity
Deeper Layers May Represent Better Reality but are Complex to Use

- Goal: Learn complex objects independent of pose, illumination, scale, occlusion, clutter
- Context: Previous machine vision leveraged human crafted features. DL proposes self-learnt hierarchical representation of object just as the human learning process does

Input image

Primitive Features

Objects

Parts

Object

Perceptron Model

\[ x = f_w(Z) \]

Shallow networks are insufficient to represent learning complexity
AlexNet: an ‘October Revolution’ in ML (2012)  
A. Krizhevsky, I. Sutskever, G. Hinton

Layer 1: Convolution and Pooling
- 96 Kernel of 11x11x3
- Neurons: 96 x55 x 55 = 290,400

Layer 2: Convolution and Pooling
- 256 Kernel of 5x5x48
- Neurons: 27 x27 x 256 = 186,624

Layer 3 and Layer 4: Convolution with no Pooling
- 384 Kernel of 3x3x256
- Neurons: 13 x13 x 384 x 2 = 129,792

Layer 5:
- 256 Kernel of 3x3x192
- Neurons: 13 x13 x 256 = 43,624

Layer 6-8: fully connected layers
- Neurons: 2 x 4096 + 1000 = 9,192

1000 Objects class – 1.2M pics 256x256xRGB
Non saturated Neurons (RLU) + Dropout CNN
Trained on 2 GPU NVIDIA GTX 580 3GB for 6 days.
Error rate – Top 1: 38% and Top5: 16% -
10% better than competition !!
Deep Learning Summary: What We Can Do with It

Image Cognition
- Classification
- Localization
- Object Detection
- Image Segmentation
- Classic convolutional Neural nets
- Most studied problem

Motion Prediction
- Input: sequences of frames or stereo vision
- Output: motion prediction

Mapping
- Input: images and radar cube
- Output: Grid map
- Explore convolutional neural nets
- New approach

Self-Driving
- Single-multiple agent games
- Markovian Process
- Reinforcement learning via NN
- Meye Driver Policy
- Berkley University
- Very futuristic
BLUEBOX – ENABLING AUTONOMOUS DRIVE DEVELOPMENT
Automated Drive – BlueBox Prototyping Platform

1. Rapid prototype platform – Linux out of the box
2. Performance analysis tool
3. Optimization of acceleration SW
4. Abstract and virtualize (Isolation & Determinism)
S32VLS2-RDB “BlueBox” Today

Board combination of LS2085ARDB-PA + S32V234PCIE and specific modifications.

LS2080A
- Number Cruncher
- ARMv8
- 250GB SSD
- 16GB DDR4
- Preloaded Linux
- Multiple Ethernets
- USB
- SDHC

S32V234
- Auto & Safety
- ARMv8
- SD-Card boot
- Preloaded Linux
- 1x Eth RJ45
- Multiple BroadR-Reach
- 2x CAN
- 1x FlexRay
Interfaces on the BlueBox

**Front Side**

- 4x 10G Ethernet (Optical SFP+)
- 4x .1/1/10G Ethernet (RJ45)
- 2x USB 3 (Type A, micro AB)

**Back Side**

- Multiple BroadR-Reach 100Mbps Ethernet + 1x 1G RJ45
- 1 FlexRay
- 2 CAN
- HDMI, MIPI supportable with lid removal or case modification
SECURE CONNECTIONS FOR A SMARter WORLD