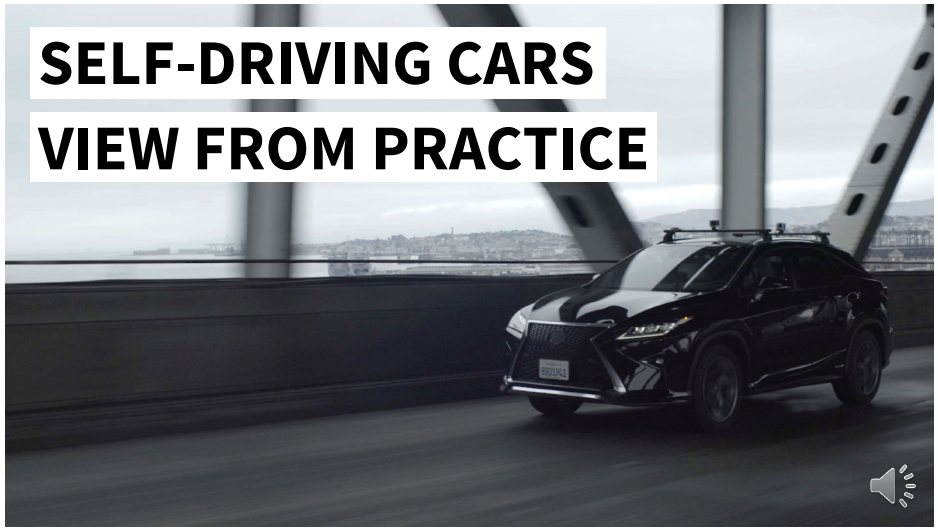


Photogrammetry & Robotics Lab

A View from Practice on Self-Driving Cars

Igor Bogoslavskyi, Apex.AI

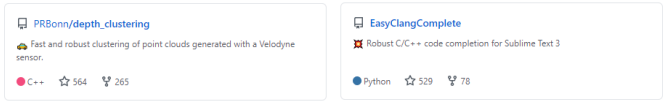
Part of the Course: Techniques for Self-Driving Cars by
C. Stachniss, J. Behley, N. Chebrolu, B. Mersch, I. Bogoslavskyi

Apex.AI

AUTONOMOUS VALET PARKING

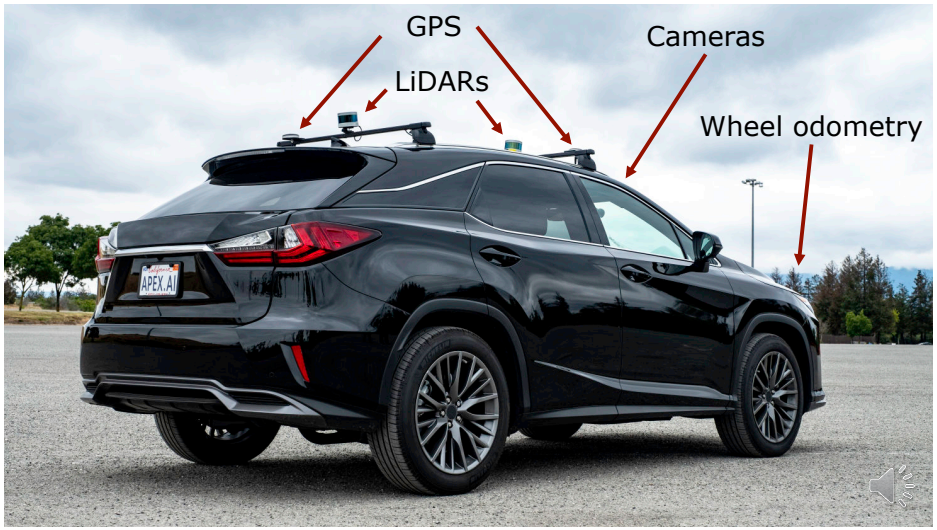
Open source

- More than 1000 stars on GitHub: /niosus



- AUTOWARE.AUTO on Gitlab.com: https://www.autoware.auto

What is this lecture about?



The car must be able to

- Sense
- Understand
- Plan
- Act

The car must be able to

- Read data from sensors
- Preprocess data
- Detect objects
- Localize itself
- Track objects
- Estimate ego-motion
- Plan path
- Plan motion
- Follow the path

Typical approach in academia

- Focus on publishing **novel** scientific results
- Hard to publish non-novel but robust algorithms
- **Focus is on a single part** of the autonomous driving stack, not on the full system
- Performance is measured mostly on publicly available datasets, sometimes quite old, **overfitting** to a particular one is quite common
- **Rarely tested in real world** as they require a readily-available autonomous driving stack where a component can be replaced for evaluation

Typical approach in industry

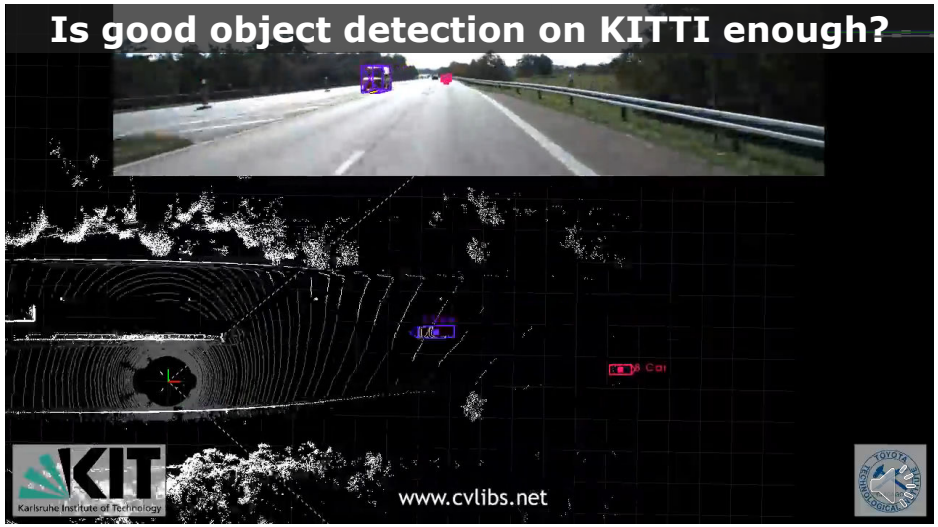
- Only big companies focus on **novel** scientific results
- The **simplest method** that does the job is the best
- **Focus is on the whole stack**, not on a single component
- Everything **must work in real world**
- **The tail is long**, it is not enough to nail 99% performance

The car must be able to

- Read data from sensors
- Preprocess data
- **Detect objects**
- Localize itself
- Track objects
- Estimate ego-motion
- Plan path
- Plan motion
- Follow the path

Object detection in academia

- Nowadays, **mostly learning-based** methods
- Both camera- and LiDAR-centric as well as the combination of the two
- Potentially also tracking through panoptic segmentation in combination with a Kalman Filter
- Mostly tested on some pre-recorded as well as publicly available data, e.g. KITTI dataset

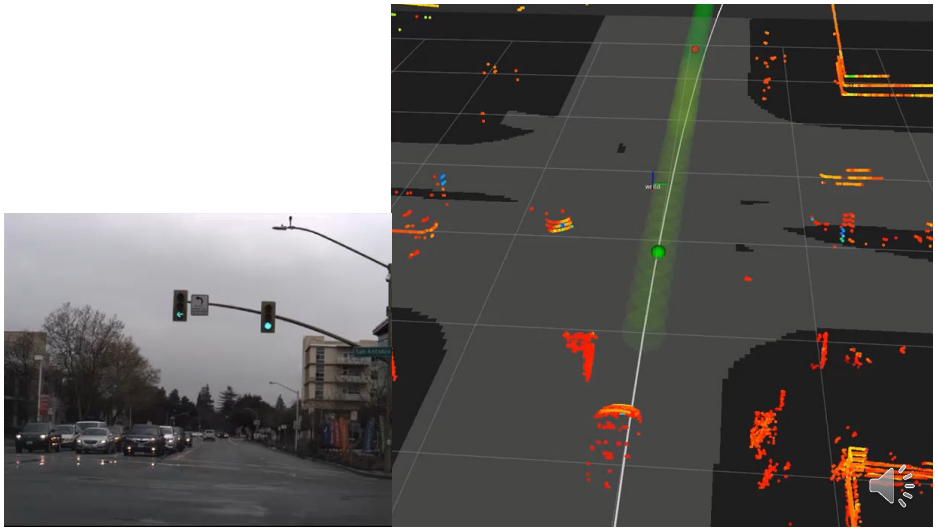


Dealing with strange situations

- The real world is complex
- KITTI has well-behaving road participants and no unidentifiable objects on the road
- While good performance on KITTI and alike datasets is a must, it is not enough for the real world
- Let us look at examples that are usually neglected in academic research

Non-learnable objects

21



Small objects lying on the road

- A car tire is about 20cm
- Typical LiDAR vertical resolution is 0.33°
- At 50m it is hard to detect such objects
- We can train a detector for car tires but not for all possible similar objects, e.g. squirrels, boxes etc.
- A free space classifier can be trained on images
- Sensor redundancy is still an open issue
- What can we run over?

23

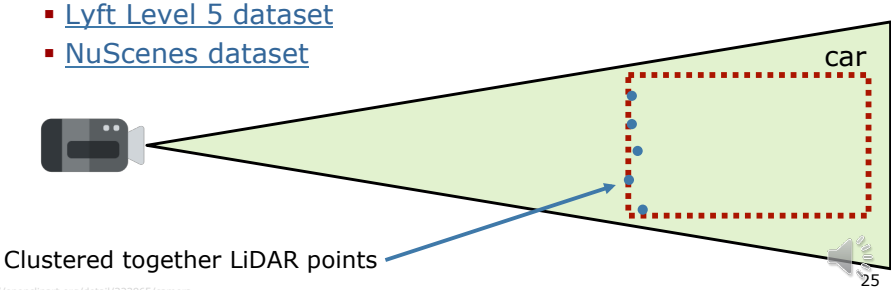
Objects very far away (~200m)

- At 0.33° vertical LiDAR resolution there will be a 2m difference between points from different lasers
- At 0.02° horizontal LiDAR resolution there will be a 13cm distance between the points from one laser
- There will only be around 10 points on a car at this distance stemming a single line
- Driving at 50 km/h we only have 7 frames to track a car driving towards us with the same speed
- We don’t want to waste even a single frame!

24

Objects that are very far away (~200m)

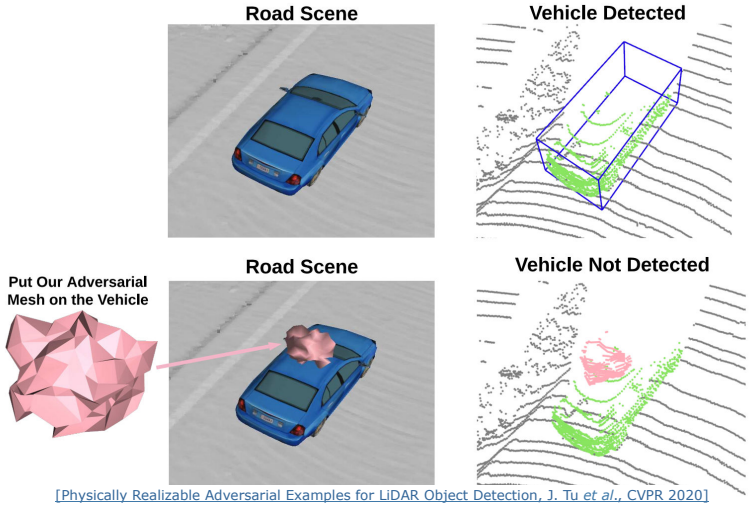
- In combination with camera detection, we can already track that object 200m away
- Datasets that offer such examples:
  - Lyft Level 5 dataset
  - NuScenes dataset



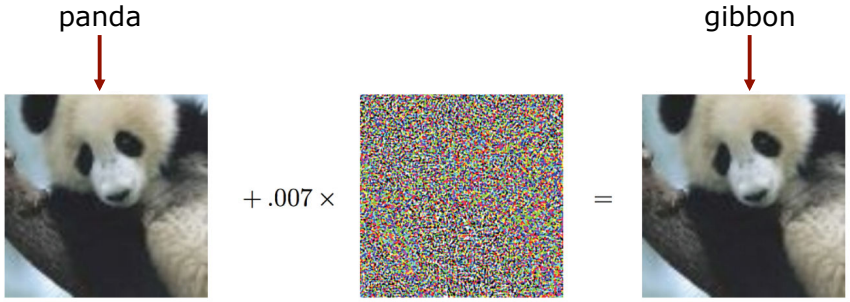
https://openclipart.org/detail/323965/camera

A word on deep learning

- In recent years camera object detection has been dominated by deep learning methods
- Same happens with LiDAR object detection: deep learning methods outperform classical ones
- However, there are issues:
  - Deep learning methods mostly detect pre-learned object
  - Prone to adversarial attacks images and point clouds
  - Hard to predict the failures
- A fallback classical geometric approach is needed



[Physically Realizable Adversarial Examples for LiDAR Object Detection, J. Tu *et al.*, CVPR 2020]



[Explaining and Harnessing Adversarial Examples, I. J. Goodfellow *et al.* ICLR 2015]

Which sensors to use for perception?

- **No single sensor is enough**
  - Cameras are hard to work with in the dark and don't provide metric information
  - LIDAR does not work well when it is wet, snowy or when objects don't reflect light, detect phantom objects
  - Radar has low resolution but measures speed and sees through fog and rain
- **No single method is enough**
  - Geometric methods don't provide enough accuracy
  - Deep learning methods can fail unpredictably

Typical perception setup

- Camera object detection and classification
- Geometric 3D clustering, object detection and tracking, potentially including radar
- Top-down object detection on occupancy 3D grid
- Nowadays also deep learning object detection on 3D clouds directly or on range images + semantic segmentation
- Free space detection with cameras and from 3D

The car must be able to

- Read data from sensors
- Preprocess data
- Detect objects
- Localize itself
- Track objects
- Estimate ego-motion
- Plan path
- Plan motion
- Follow the path

The car must be able to

- Read data from sensors
- Preprocess data
- Detect objects
- Localize itself
- Track objects
- Estimate ego-motion
- Plan path
- Plan motion
- Follow the path

The car must be able to

- Read data from sensors
- Preprocess data
- Detect objects
- Localize itself
- Track objects
- Estimate ego-motion
- Plan path
- Plan motion
- Follow the path

Localization for self driving cars

- Most of the components of the self driving car stack rely on localization
- The car must know where it is to:
  - Estimate its own movement
  - Estimate movement of other objects
  - Plan where to drive
  - Execute the planned maneuvers
- To localize, the car needs a map

Maps have different layers

1. A **3D layer** created by LiDARs
2. A **semantic layer** with information about:
   - Road lanes with driving directions and connections
   - Speed limits and other information
   - Positions of interesting objects like traffic lights
3. Optionally a **layer with dynamic changes**, e.g. road construction or roadblocks

Creating a 3D map layer

- Data from 3D LiDAR(s)
- Sequential registration with some flavor of Iterative Closest Point (ICP) algorithm
- Loop closure when entering already visited areas
- Pose graph optimization upon loop closures
- Stored as a collection of point clouds that can be loaded on demand

3D Mapping

Real world is much more complex

- There might be dynamic objects in the data
- Perception integration needed to detect and remove the dynamic objects
- Maintaining a full 3D map is complicated
  - Detecting changes is a non-trivial task
  - 3D maps require lots of storage
  - Big 3D maps are expensive to compute
- Online 3D map construction without the dynamics is still a non-trivial task

Mapping in academia vs industry

- In academia **online SLAM** is still a vibrant topic
- The papers strive to provide robust methods for map construction online
- Autonomous driving companies rarely need map construction to happen online
- Usually, **SLAM is performed offline**
- Systems are designed in a way to easily modify loop closure connections to improve the resulting map

41

Localization in a 3D map is largely "solved"

- Data comes from 3D LiDAR(s)
- Point clouds are registered **to a given map** with some flavor of Iterative Closest Point (ICP)
- A semantic map layer can help but it is not a must
- Alternatively, localization can be done visually (not in this lecture)

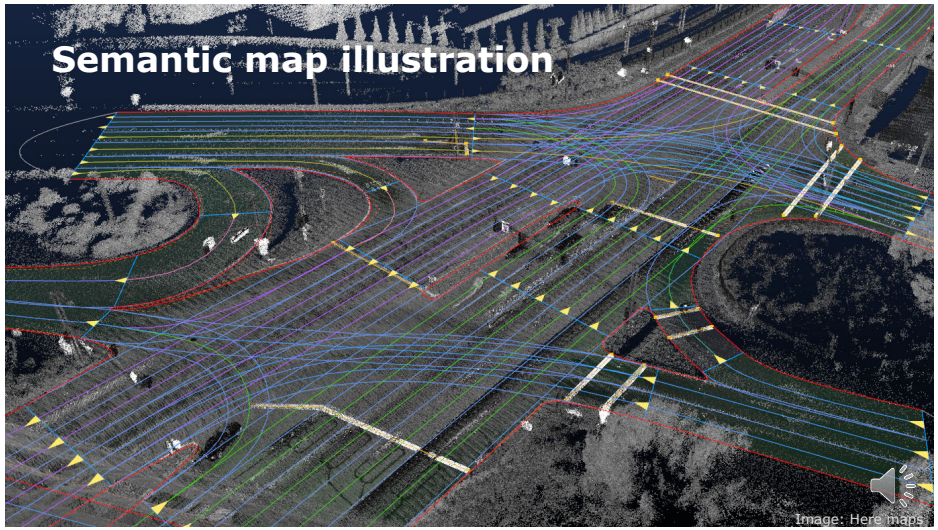
42

43

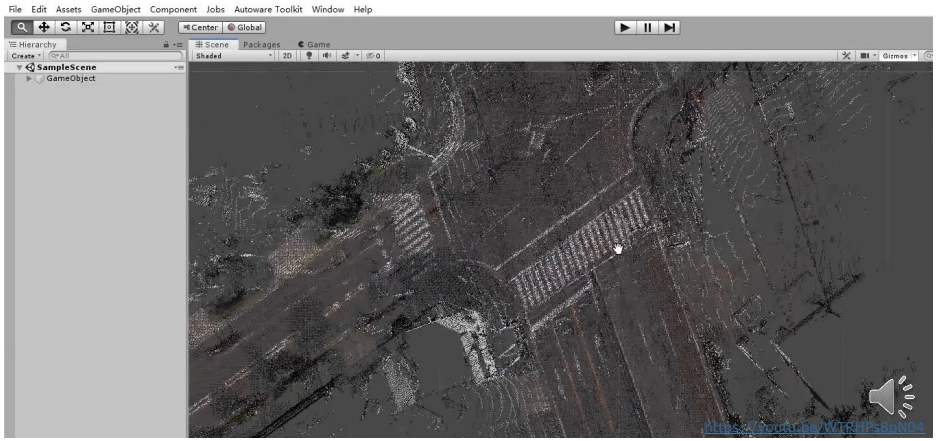
Semantic map layer

- Contains additional information about the environment:
  - Road lanes with driving directions
  - Traffic lights
  - Speed limits
  - Additional features
- All of these are aligned to a 3D map
- Process of aligning is semi-manual

44



Creating a semantic map is labor-intensive



Semantic maps for traffic light detection

- Semantic map contains positions of all traffic lights
- Semantic map is aligned with the 3D map
- The car is localized in the 3D map
- This provides a good guess where the traffic lights are with respect to the car position
- This changes the traffic light detection & classification problem into pure classification

Vanilla traffic light detection

Semantic maps for traffic light detection

What to do when the world changes

- Construction sites open and close, speed limits change, lanes are added and removed
- **Dealing with changes is very hard!**
- **Typical approaches:**
  - Have a constantly-deployed mapping-vehicle fleet that updates the map for all other vehicles
  - Implement change detection algorithms that allow the whole fleet to suggest changes
- Regardless of the approach manual supervision is usually needed

What if we can't align to a 3D map?

- Maintaining a global 3D map is hard
- It is hard to scale over multiple cities/countries
- It takes a lot of processing power to align to a 3D map with full 3D point clouds
- Not all cars have a high-definition LiDAR on the roof

Aligning to an HD Feature Map instead

- HD Maps are very **similar to semantic maps** but with additional features
- A decision on important information that the map must hold has to be taken before-wise
- Localization in such a map happens **based on features** extracted from the sensor data
- Sensor data is not as descriptive as with a 3D LiDAR
- Not suitable for levels 4/5 autonomy, but usually enough for levels 2/3

Short mapping and localization summary

- Maps are usually layered
- 3D mapping and localization is mostly "solved"
- Semantic maps allow cars to know where to drive and help making some tasks easier
- The world is changing and keeping up is hard
- Not everyone can use full 3D mapping and localization, so HD feature maps are used instead
- Localization in HD feature maps is still not solved

The car must be able to

Read data from sensors

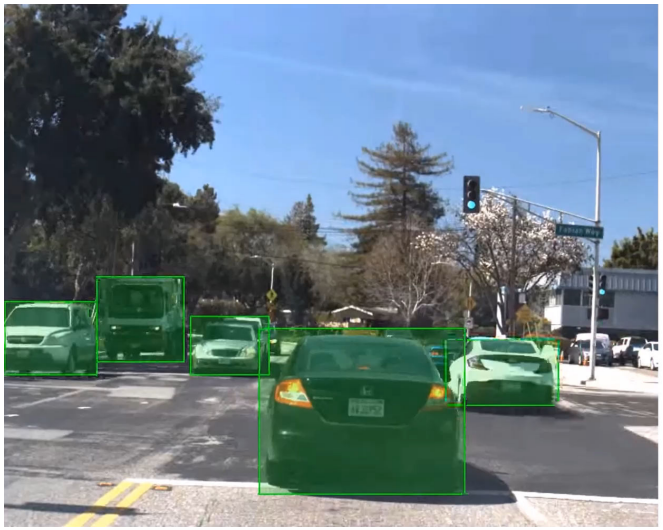
Preprocess data

Detect objects	Localize itself
Track objects	Estimate ego-motion

Plan path

Plan motion

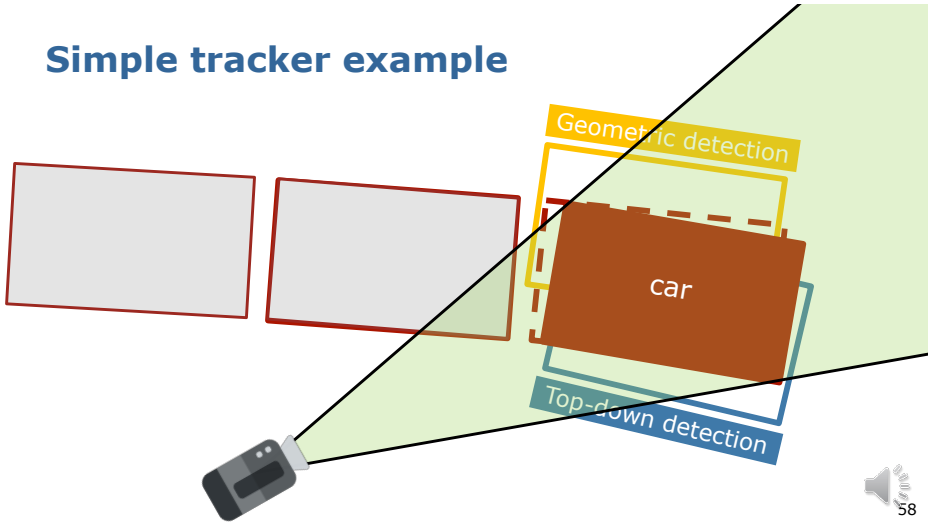
Follow the path



Detection, tracking, and prediction pipeline

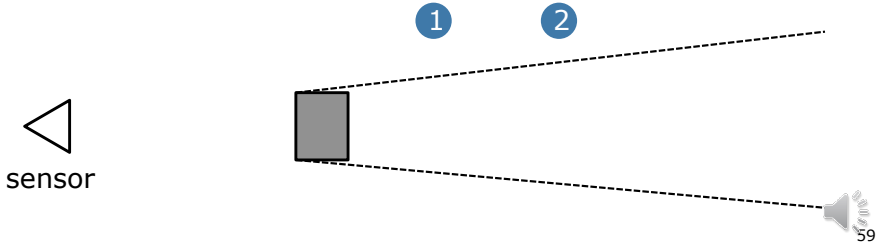
- Objects can be detected from **multiple sources**:
  - LiDAR-based geometric bounding boxes or convex hulls
  - LiDAR-based birds eye view (BEV) object detection
  - Camera object detections (provides object class)
  - Radar readings
- All of these must be assigned to each other and tracked jointly to predict what happens next
- Usually some form of **Extended Kalman Filter** or **Particle Filter** is used to track and predict
- Prediction is a huge topic

Simple tracker example



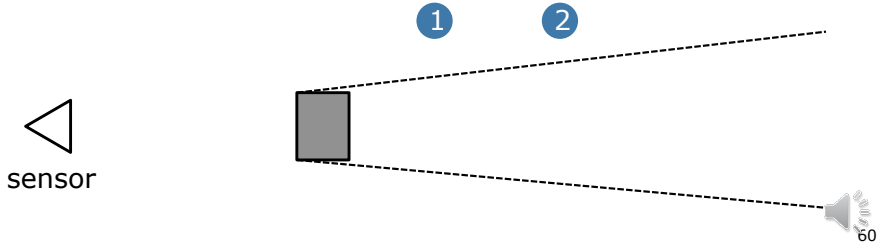
Tracking and prediction is hard

- A track can be hidden from the sensors for a time
- Objects can split or merge as a result of detection errors: how to assign new objects to existing ones?
- An object can change direction while unobserved



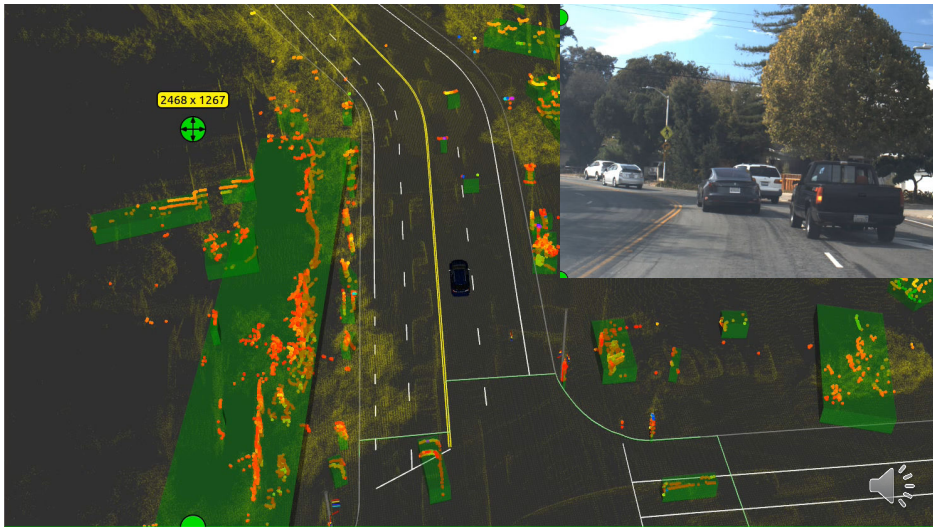
Tracking and prediction is hard

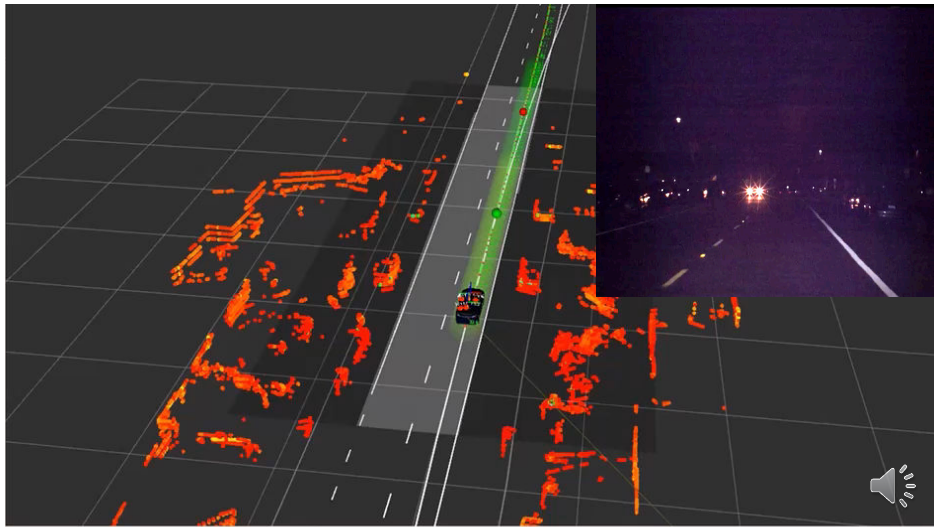
- A track can be hidden from the sensors for a time
- Objects can split or merge as a result of detection errors: how to assign new objects to existing ones?
- An object can change direction while unobserved



Typical approach to tracking

- Standard way is to use an **Extended Kalman Filter** or a **Particle Filter** to track and predict objects
- Panoptic segmentation allows to assign an id to any object, simplifying tracking
- Prediction is still a very much open topic
- Relatively **easy to predict expected behavior** on a semantic map along lanes
- **Hard to predict unexpected behavior**





Short summary on tracking

- Robust tracking is the at the core of every self-driving car stack
- Tracking takes data from multiple sources
- Making tracking robust is still a complex task
- Predicting what the tracked objects do is even harder, especially when unexpected behavior occurs
- There is no integrated robust solution in academia

Runtime – making everything run fast

- Academic research often downplays the runtime
- It is common to use "real time" in place of "at sensor frame rate" in academia
- Real time systems (RTOS) must finish every operations in a specified amount of time
- Worst-case runtime is must be **guaranteed**
- If a component takes longer than expected it might be killed which imposes **additional constraints on system design**

Best-case scenario is not enough

- The code must fulfill performance guarantees
- Also if the processor is under load
- Also if the network is congested
- There must be performance tests present
- Careful worst-case scenario analysis is needed
- The code must be written in a way that maximizes performance by efficiently using available resources, i.e., thinking about cache locality, which work is done by which thread, priorities, etc.

Algorithms are written and run fast! Are we ready to hit the streets?

Safety is extremely important!

- The car must be safe for occupants and for the surrounding people
- Safety deals with situations like:
  - What happens if a tire blew up
  - What if a sensor is unresponsive
  - What if a sensor sends bogus data
  - What if car software fails
  - How to deal with random/systematic failures
- There must be fallback options for anything that can go wrong to avoid endangering people

How do we know the system is safe?

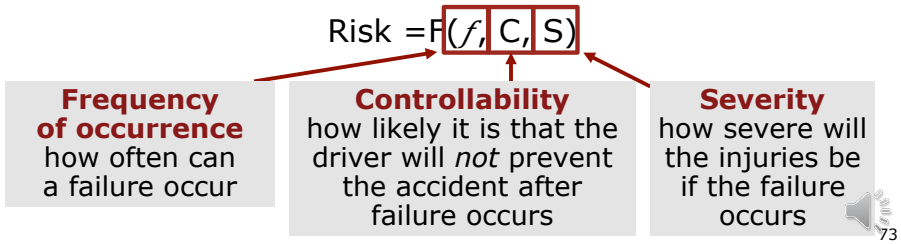
- It is extremely hard to build safety into the system after it is designed and implemented!
- **We design systems to be safe from the start!**
- Audit the resulting system to ensure the implementation matches the design
- The design is guided by **HARA**:
  Hazard **A**nalysis and **R**isk **A**ssessment

ISO 26262 functional-safety standard

- Is an **international standard** for functional safety of road vehicles that covers:
  - **Formal definitions for errors** that might happen during the nominal operation of a car
  - **Formal definition of risk** with relation to the possible errors and ways of risk assessment and mitigation
  - **Processes to be followed** to ensure safety
- Enforces an **independent safety assessment**

Risk and safety

- Risk plays the central role in safety assessment
- **Safety = absence of unreasonable risk**
- Unreasonable risk definition depends on many factors and is still widely debated



Frequency of occurrence

- Consists of two parts:
  - **Exposure:** how often is the car in a situation where a safety hazard due to a failure can occur, denoted by **E**
  - **Failure rate:** The probability of a particular piece of hardware/software to fail
- Failure rate is **not** considered in risk assessment
- Instead, every item is assigned a so-called **ASIL** level that mitigates such risk
- ASIL level trickles down to individual components which defines how reliable they must be

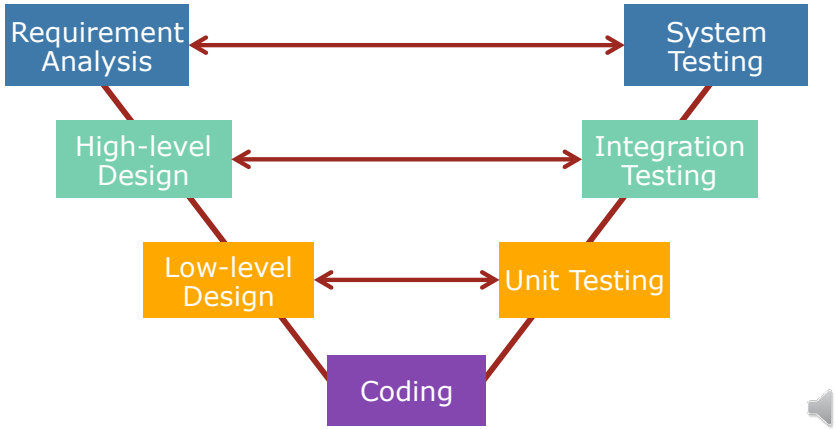
ASIL: Automotive Safety Integrity Level

- ASIL is a function of:
  - Severity **S**: ranging from **S0** to **S3**
  - Exposure **E**: ranging from **E0** to **E4**
  - Controllability **C**: ranging from **C0** to **C3**
- ASIL levels can be **QM**, **A**, **B**, **C**, **D**
- Levels from **A** to **D** grow in safety consideration
- Table 4 of the ISO 26262-3 standard suggests an ASIL level based on the values for **S**, **E**, **C**

Example: adaptive cruise control braking

- Adaptive highway cruise control function consists of braking, steering and propulsion systems
- Braking is an **item** in terms of ISO 26262
- This item gets an **ASIL D** level calculated from:
  - Severity **S3** – fatal injuries are likely
  - Exposure **E4** – can happen at any time on a highway
  - Controllability **C3** – complex abrupt action by the driver needed to avoid the crash
- We now derive a safety goal for the braking function of an adaptive cruise control

The V-model



ISO 26262-9:2018: Figure 1

- The process starts with requirement analysis on the falling edge of the diagram
- Requirement analysis corresponds to system testing on the rising edge of the diagram
- **Example requirement:**
  The car **shall** brake if a slow car merges into its lane at an intersection
- Requirement satisfaction is tested by system tests

System tests

- System tests test the **behavior** of the system without knowledge of the implementation details
- These tests can be created from simulated data as well as pre-recorded real-world data
- Every test has **criteria to pass**, e.g.:
  - The car drove at least (or not more than) some distance
  - The car did not crash
  - The car reached a certain speed
  - …

Intersection system test

High-level Design ⟷ Integration Testing

- **High level design defines the architecture**
- In our example the architectures for throttle control, object detection and prediction must be defined
- The throttle control module **shall** brake if an obstacle is predicted to be in its path
- **Integration tests** are then designed to verify the behavior of these components working together as black boxes

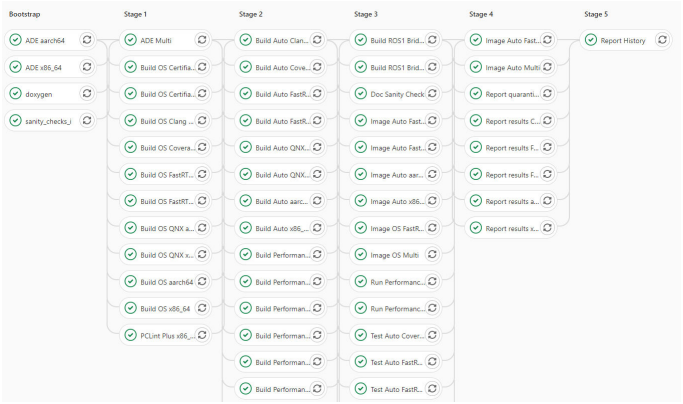
Low-level Design ⟷ Unit Testing

- The architecture is broken down into detailed design documents, UML diagrams etc. for each component
- A component is always a single module, class, etc.
- The **system requirements trickle down** into the single component requirements, e.g.:
  "the class shall have a specific API"
- Each component is tested by its **unit tests**
- `EXPECT_EQ(42, oracle.GetMeaningOfLife());`

Coding

- Best coding practices must be followed to avoid possible failures
- Typical coding standards: MISRA, AUTOSAR
- Static code analysis that checks these is a must
- Commits must be tagged in a specific way for traceability to a specific requirement
- A comprehensive Continuous Integration system is a must to ensure the code is always in a good state

Extensive testing upon every push



Languages can also be unsafe

- This course is in Python
- If you just want to work in perception by designing artificial neural networks, then it is enough
- Otherwise, C++ is currently a standard language (see Modern C++ course by Ignacio Vizzo)
- Needs to have the speed, to manage resources in a deterministic and direct way and to follow guidelines for safety critical development

Example safety consideration: memory

- Dynamically allocating memory is a blocking operation, e.g. `vector.push_back(42);`
- It is hard to predict the runtime of this operation
- All operations in safety-critical system must be deterministically time-bound
- **Solution:**
  - Use static memory where possible
  - Allocate dynamic memory in initialization phase, only access it in the operation phase

ISO 26262 is not the only standard

- ISO 26262 mostly targets vehicles with a driver
- Extensions and modifications are probably needed for full self driving cars
- Other standards are being developed to address specifically autonomous vehicles, e.g. UL4600, ISO/DIS 23150
- Some companies design their own standards
- Expect more to happen in this field in future

Safety certification

- Safe software can be formally certified
- Note, that **certified != safe**
- Manufacturers of technical products must ensure that these are developed according to all aspects of functional safety and state-of-the-art science due to law of **liability**
- In case of damage, the **manufacturer must submit evidence** of the above
- Certification is a good starting point for reasoning

Safety summary

- Car must be safe to drive on public roads
- Safety is the absence of unreasonable risk derived from hazard analysis and risk assessment
- Following ISO 26262 can simplify system safety assessment and helps reducing risk of failure
- Safety certification can help in dealing with liability
- Safety enforces changes onto the system development and management process
- Safety must be a first-class citizen

Closing thoughts

Summary

- It is not enough to have nail 99% of the cases, see a great talk by Andrej Karpathy: https://youtu.be/hx7BXih7zx8
- Most of the work goes to handling corner cases
- Care should be taken about runtime and safety
- There is overlap but also discrepancy between industry and academia
- There is still lots of hard challenges!

Where to go from here

- Start a PhD and work on relevant topics
- Contribute to open source
- Test your algorithms in challenges:
  - Lyft Level 5 Kaggle challenge
  - NuScenes dataset and challenges
  - Waymo dataset and challenges
  - Others…
- Pick people you are passionate about and try to get to work with them on relevant problems in industry


Daft Punk by Juliana Luz
Our work is never over!