FORMULA STUDENT

Institution of MECHANICAL ENGINEERS

FORMULA STUDENT
Institution of MECHANICAL ENGINEERS

Making a car drive itself

How to prepare for FS-AI

Ignat Georgiev and Oliver Day

Learn to Win
09/11/2018

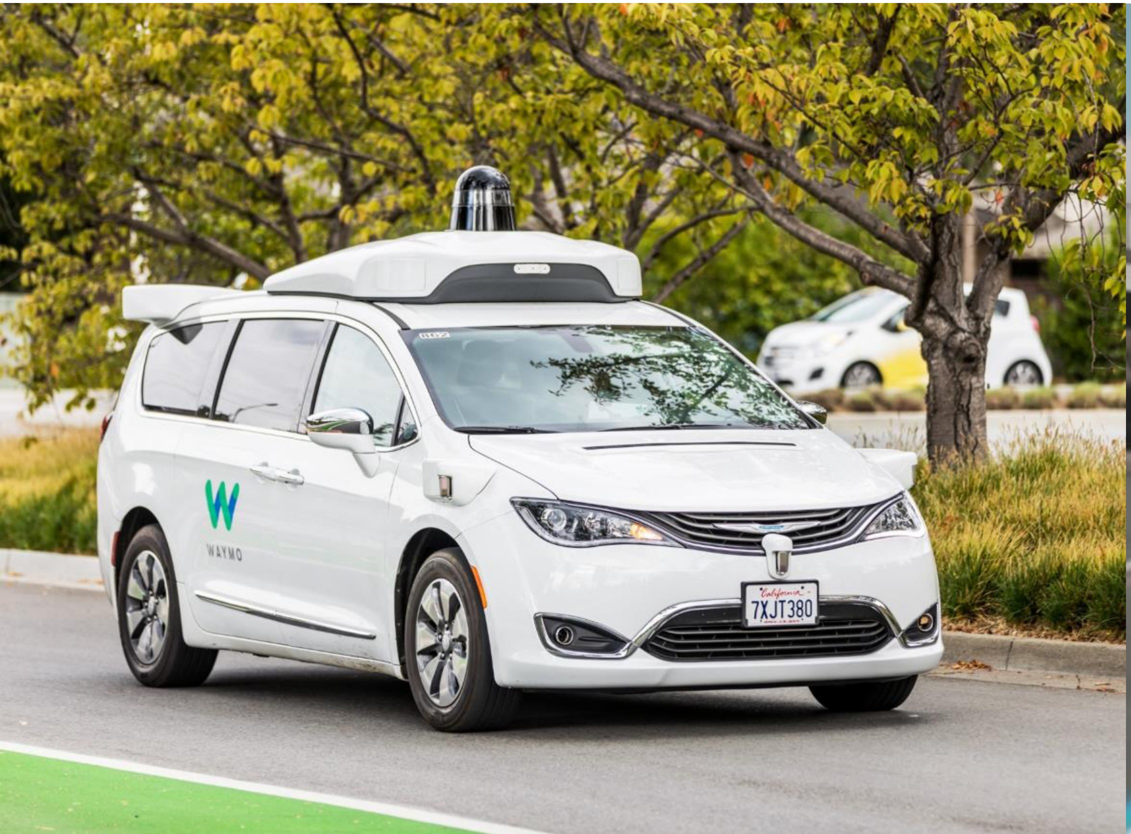
Agenda

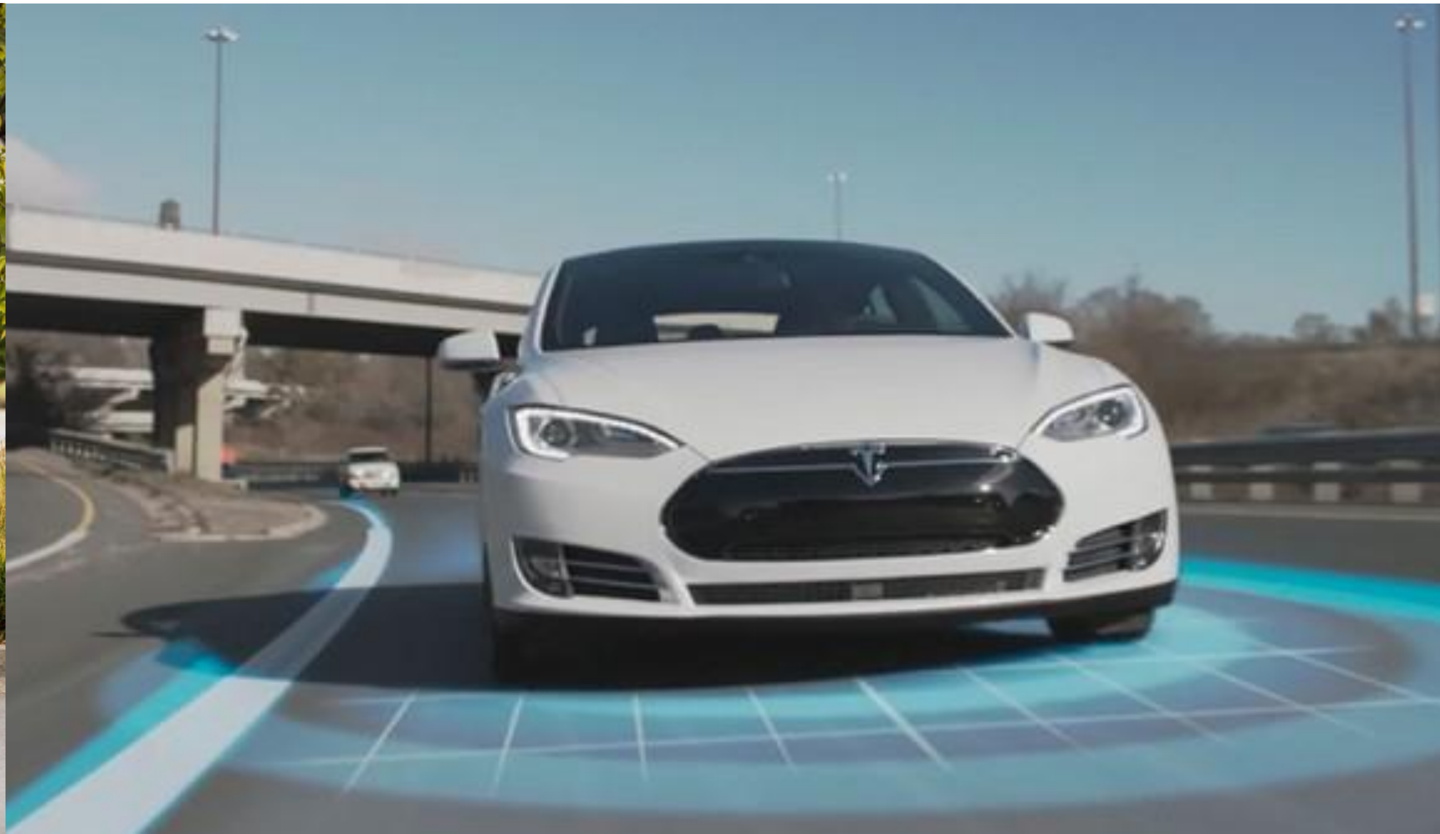
- The challenge
- Why you should do it
- Founding a team
- Technical aspects
- Software tools, development strategies and testing
- Tips for competition

FORMULA STUDENT

Self-driving cars today

- Emerging industry set to revolutionise transport in the next decade
- Blend between engineering and computer science






03
Powered by Diesel
Red Bull
Drivers not required.
MDV
MOHR DAVIDOW VENTURES
ANDROID
Red Bull
MDV
Electronics Research Laboratory

The FS-AI challenge

- Create a FS car that can drive itself
- Split into two categories:

DDT

ADS

Why should you join?

- Exciting and innovative
- Actively contribute to the field
- Get hands-on experience
- Make a career out of it
- Fun

"Self-driving cars industry in the UK will be worth £28 billion in the next 17 years."
- WIRED UK

https://www.wired.co.uk/article/driverless-cars-uk-self-driving-cars

Founding a team

- Multi-disciplinary team
- Bridge the gap between your Engineering and CS departments
- Autonomous cars aren't cheap – start with a simulation
- Find academics to support you
- Start small (<10 people) until you figure out what you need to do
- ADS and DDT can be developed in parallel. Software can be made to be applicable to both

Integrating to your existing FS team

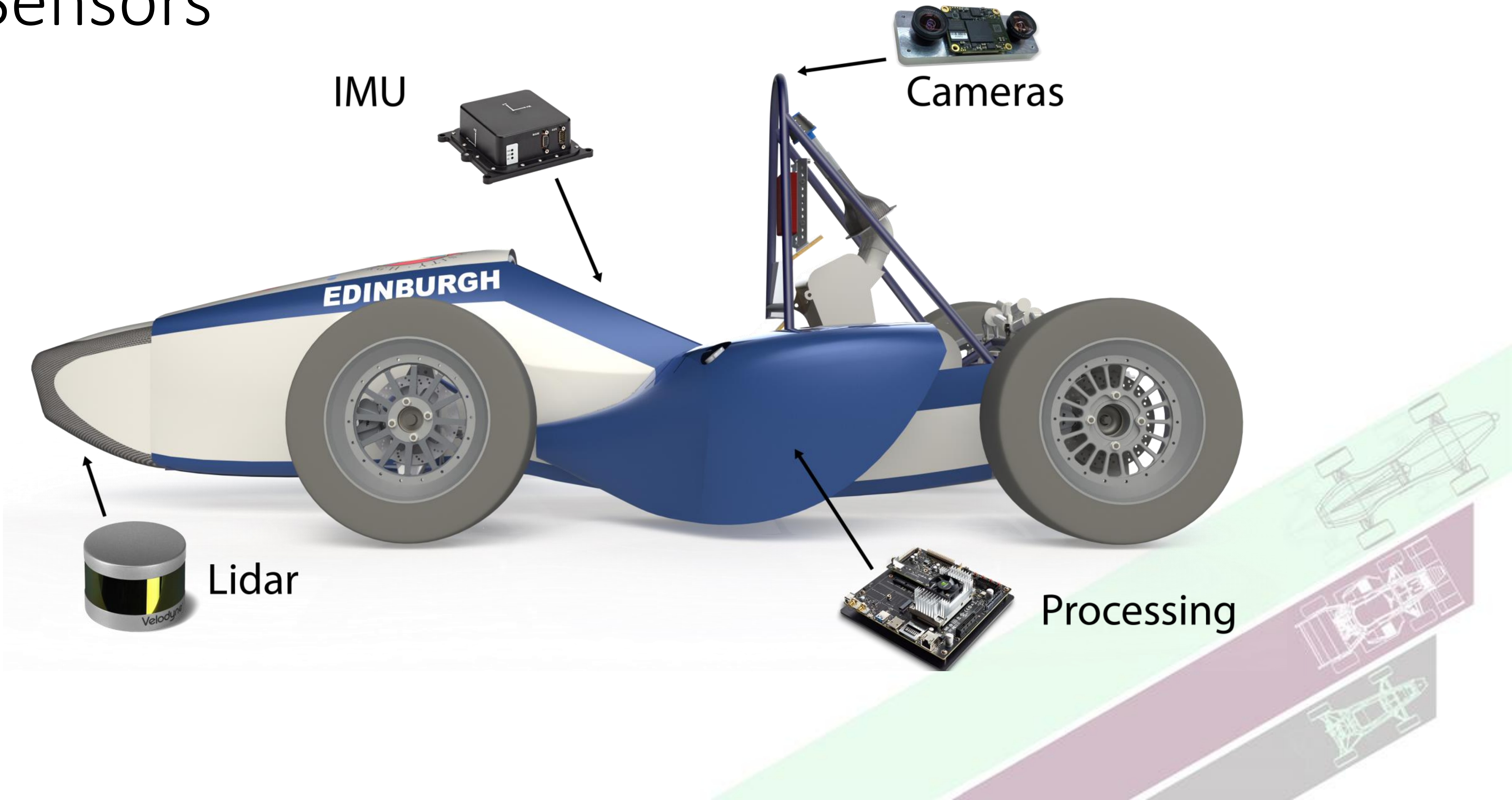
- Share your old platform
- Share knowledge and skills
  - But set clear boundaries at the beginning of the year
- Share resources – tools, workshops, transportation, etc.
- Things to consider
  - Recruitment conflicts
  - Sponsors conflicts
  - Operations and events
  - Budgeting and sharing resources
  - Team identity – social media, newsletters, etc.

Management

- Keep developers focused – have people to deal solely with operations, events, financials
- Tight communication between subteams
- Plan the year ahead
- Regular review of progress
- Balance between engagement and hard work

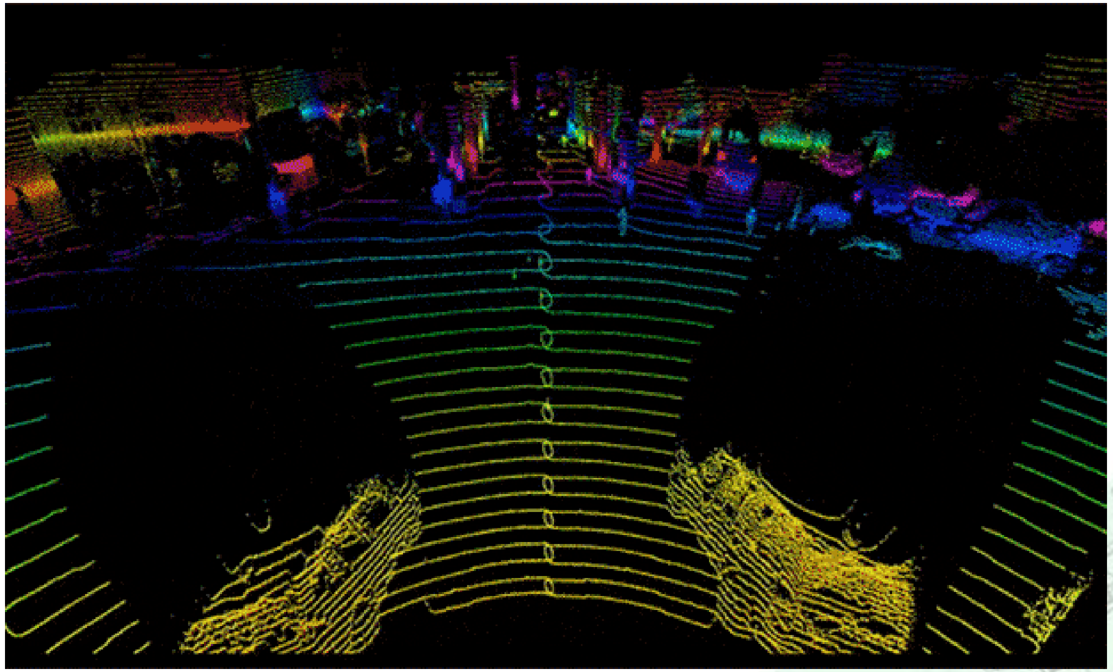
Technical

Sensors



Lidars

- High accuracy
- High output rate
- Output point cloud
- Expensive
- Fragile



Types:

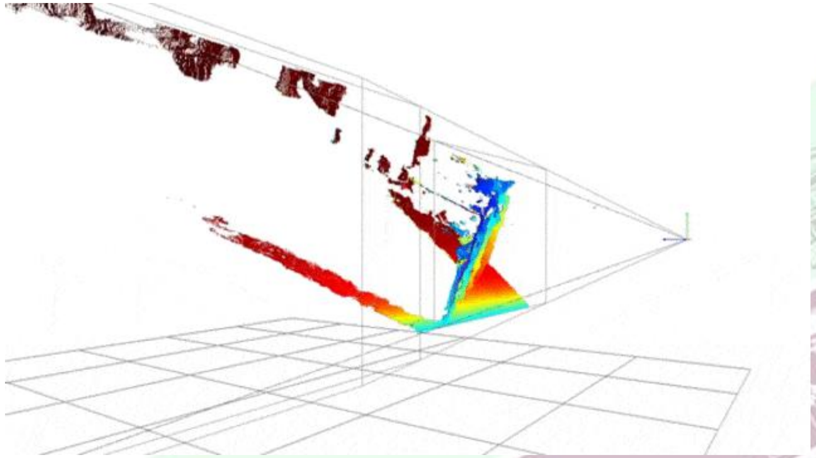
- Planar
- Multi-beam
- Image-like
- Solid-state

Cameras

- High output rate
- Lots of data
- Versatile
- Cheap
- Computationally expensive
- Huge variety – choose carefully

Types:

- Monocular
- Stereo



IMU and GPS

- Huge variety!
- Estimate location in 3D space
- From 5$ to 5,000$
- Quality varies greatly

- You can get standalone IMU and GPS
- You can get integrated INS (IMU+GPS)
- You can also get RTK GPS

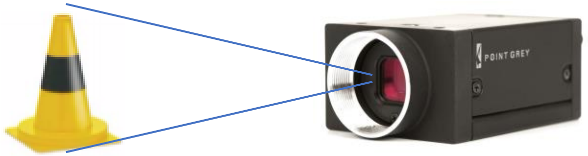
Computing

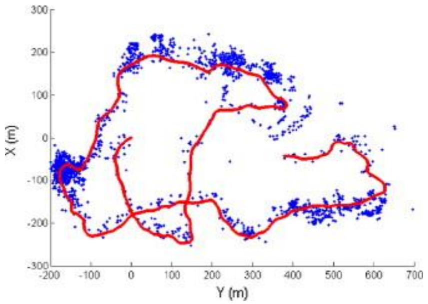
- Process intensive tasks
- Desktop PC vs industrial computers
- Arm64 vs x86
- Difficult to estimate how much power you need. To be safe go with more than you need right now
- Distributed computing
- Dedicated computers

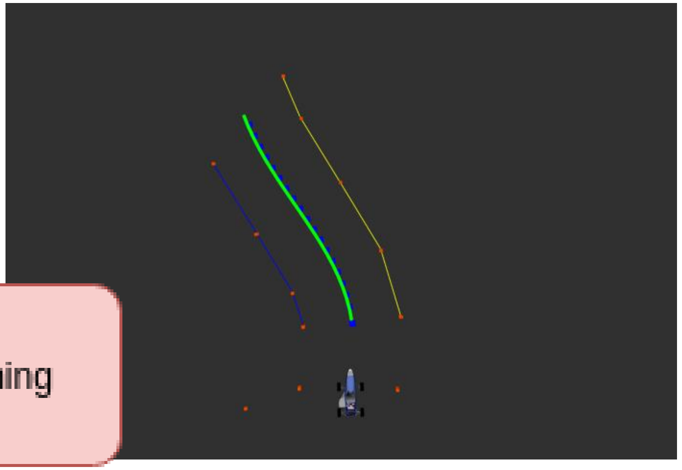
- Strap a laptop first before you buy PCs

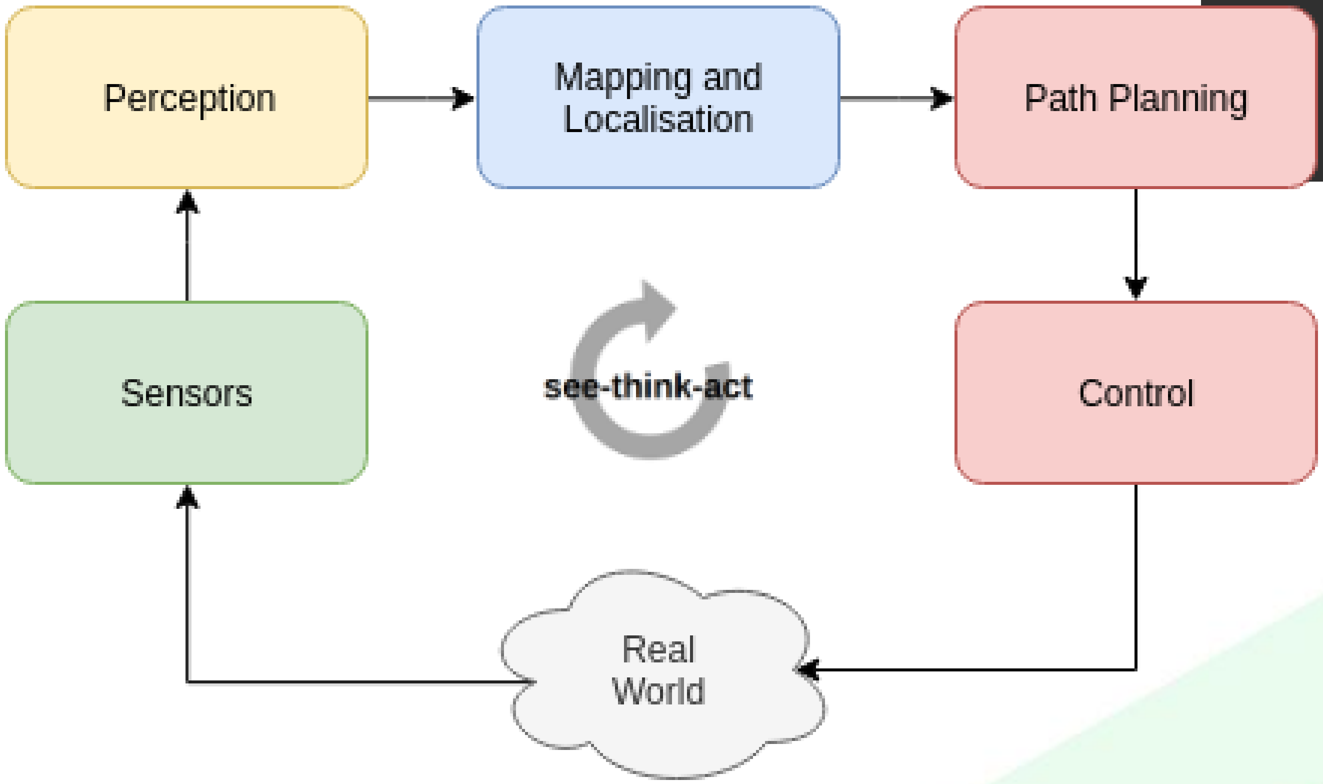


System overview



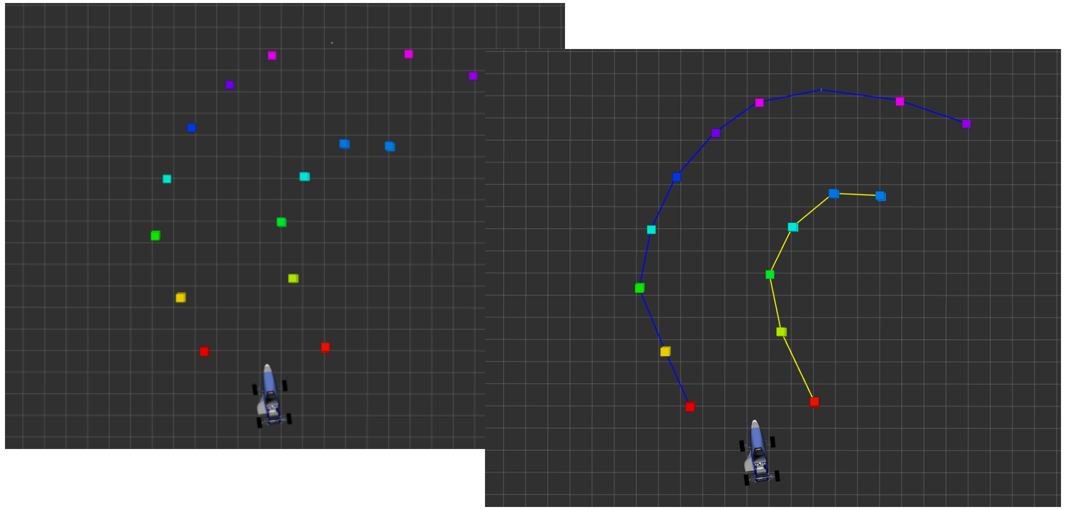






Perception

- Extracting meaningful information from sensors
- ie. from sensor data figure out where the track is
- What's a cone and where are they relative to the car (in 3D)?
- Could also be pedestrian detection as safety feature

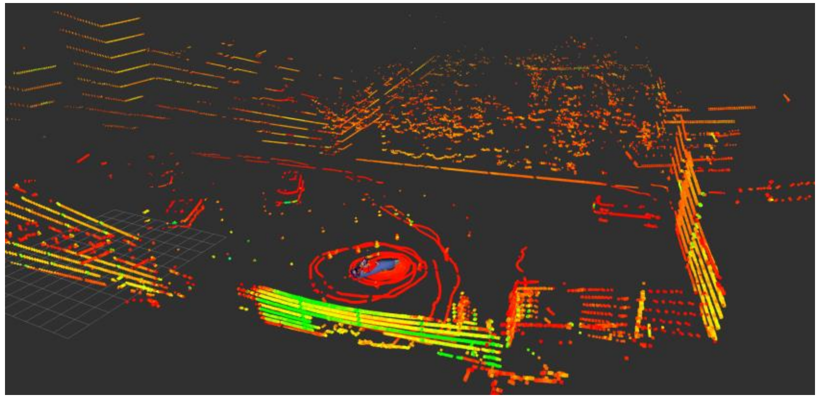


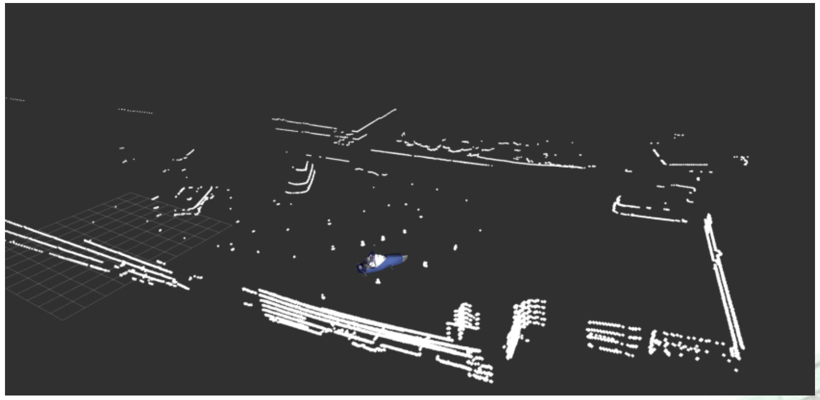
LiDAR → Detect cones ← Camera

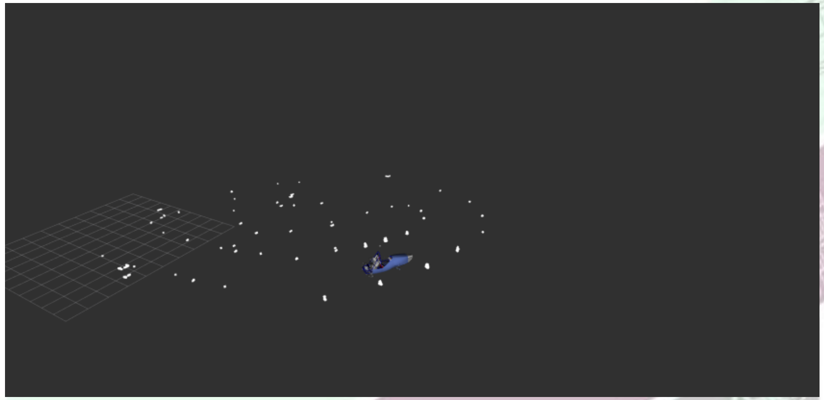
Detect cones
↓
Extract 3D cone locations
↓
Estimate track boundaries
↓
Onto planning / control and localisation / mapping...

Perception (LiDAR)

- Process LiDAR:
  - Down sample
  - Extract ground plane
  - Limit interested region eg. Heights
- Cone detection in 3D:
  - Euclidean clustering – basic approach
  - Feature matching eg. corners, edges
  - CNNs for pointclouds – get the algorithm to learn the features

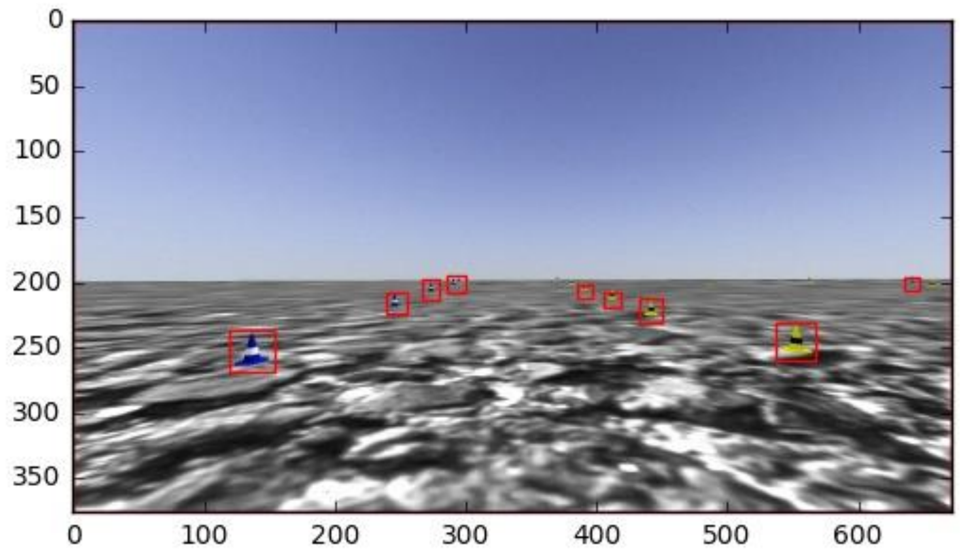




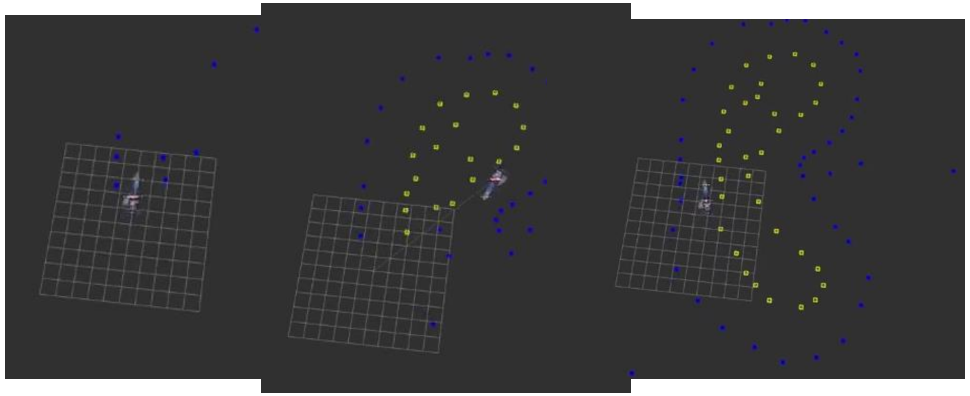


Perception (Camera)

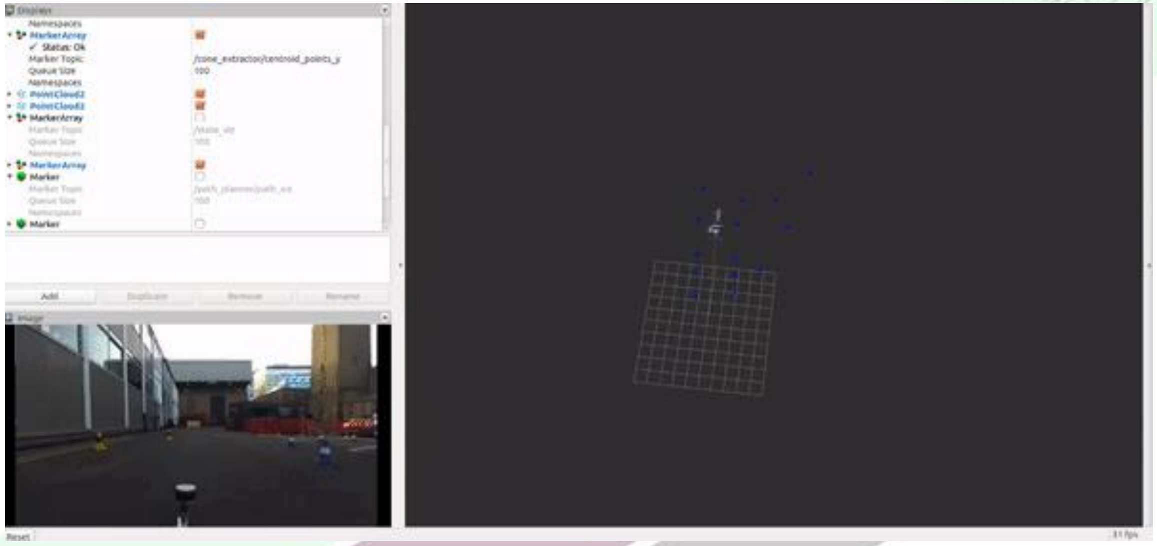
- Process camera:
  - Compression?
  - Grayscale?
- Detect cones:
  - Colour thresholding and tracking
  - Feature matching eg. SIFT
  - CNN object detection (YOLO) - get the algorithm to learn the feature matching
- 3D location:
  - Structure from stereo
  - Structure from motion
  - LiDAR – Camera calibration
- Drivable area / track segmentation



Localisation and Mapping



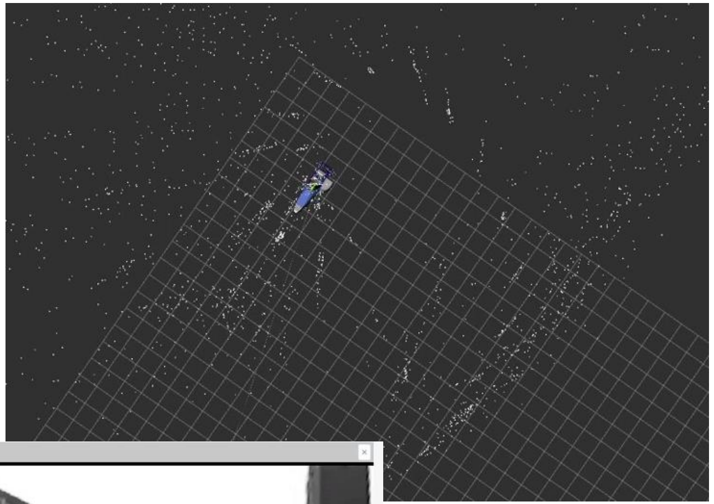
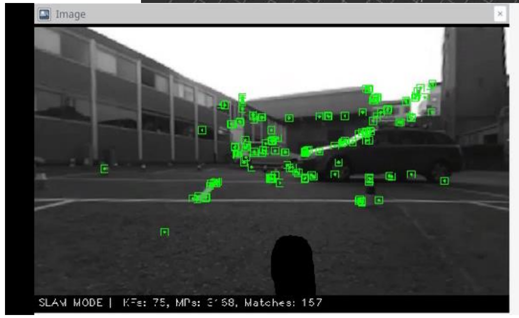
- First time we go around the unknown track we use only sensor data
- Let's use this to build up a map or model of the track and determine car's position
- Helps us to plan what's coming up when we see it again

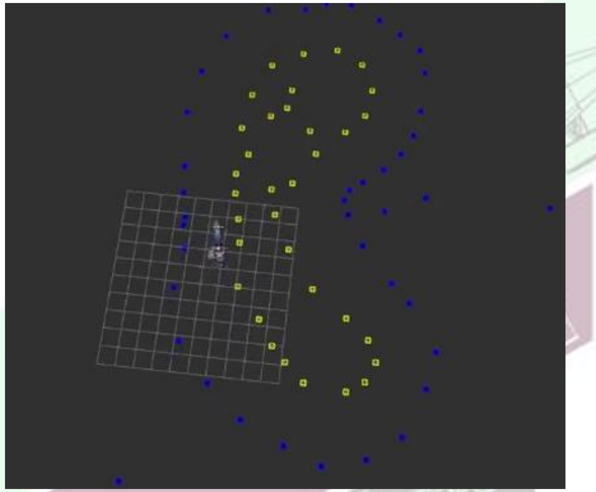


FORMULA STUDENT
Institution of MECHANICAL ENGINEERS

Localisation and Mapping

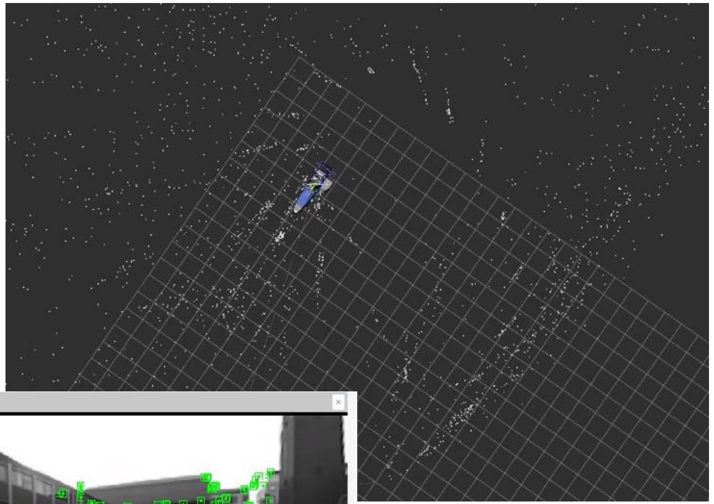
- What to record in our map?
- SLAM – Simultaneous Localisation and Mapping
- Dense SLAM eg. ORB-SLAM
  - Maps more general features eg. Edges, lines
  - More information stored for localisation
- Landmark based SLAM eg. EKF SLAM
  - Maps only what we need - cone locations in 3D
  - Depends on cone extraction eg. From LiDAR
- Also 2D approaches mapping in ground plane

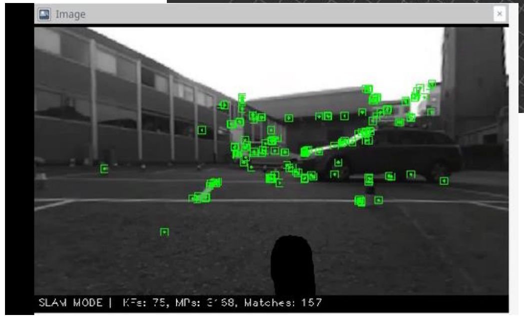



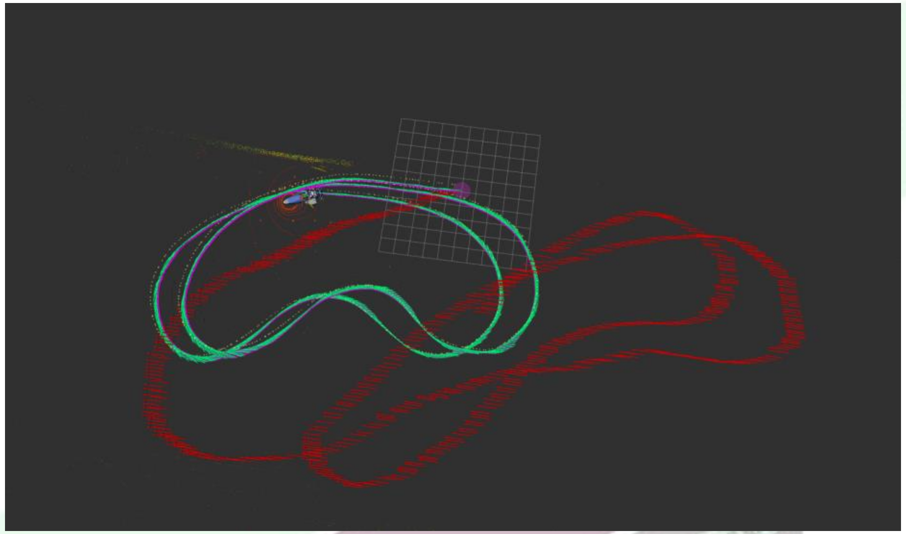


Localisation and Mapping

- Localisation - Have to locate car in the map in order to plan
- Effector noise eg. friction
- Odometry sensor noise
- Better localisation estimates by fusing IMU, GPS, odometry etc.
- But still error prone
- Solution: relocalise by matching sensor data to that expected based on map
- Probabilistic framework formalises uncertainty and belief states



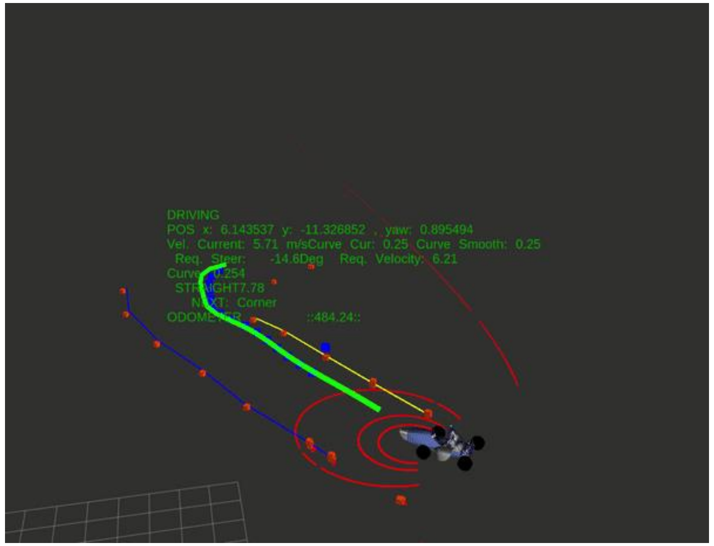




Path planning and Control

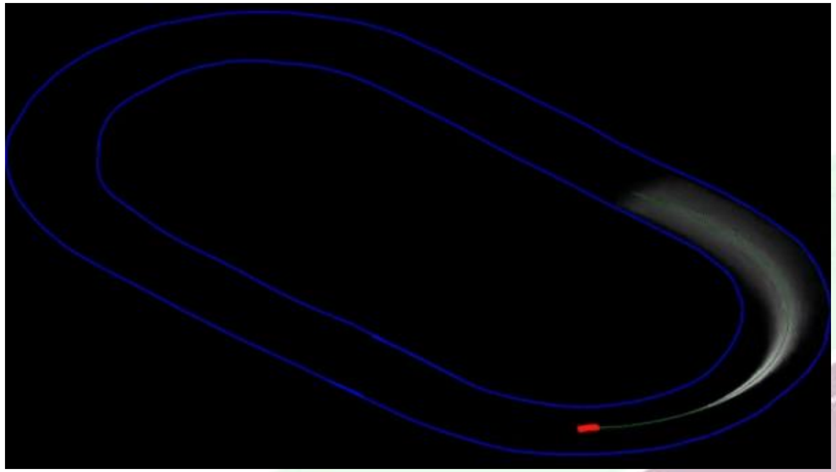
Local planning

Plan a trajectory and follow it only on immediate input data



Global planning

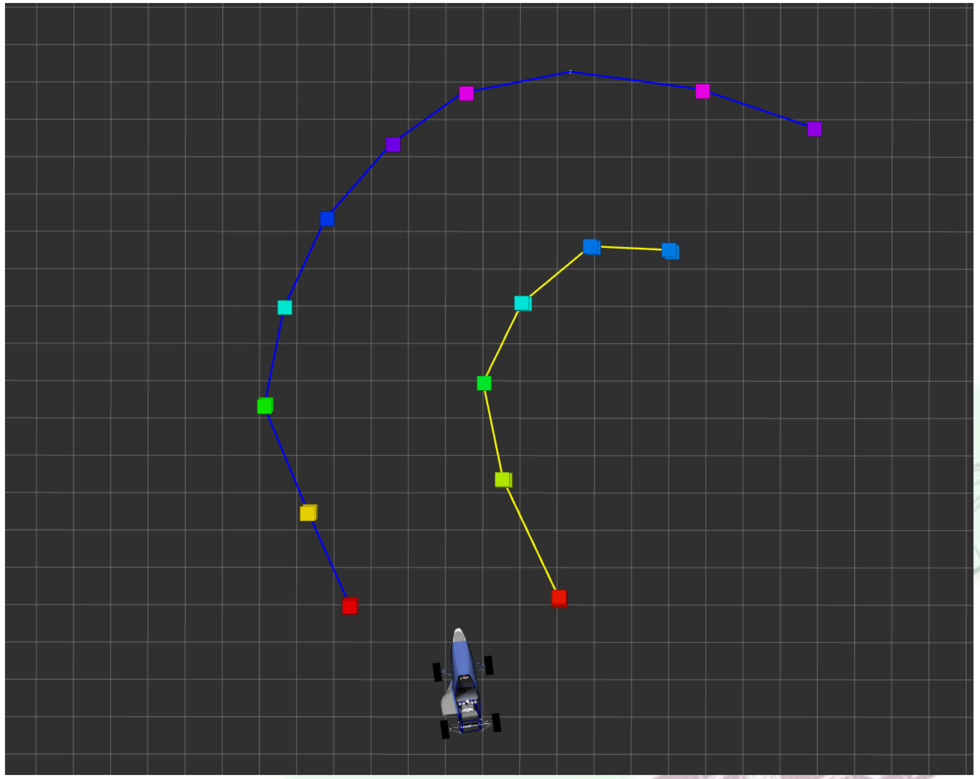
Once you have a full map of the track, can you optimise to go faster?



https://autorally.github.io/

Local planning

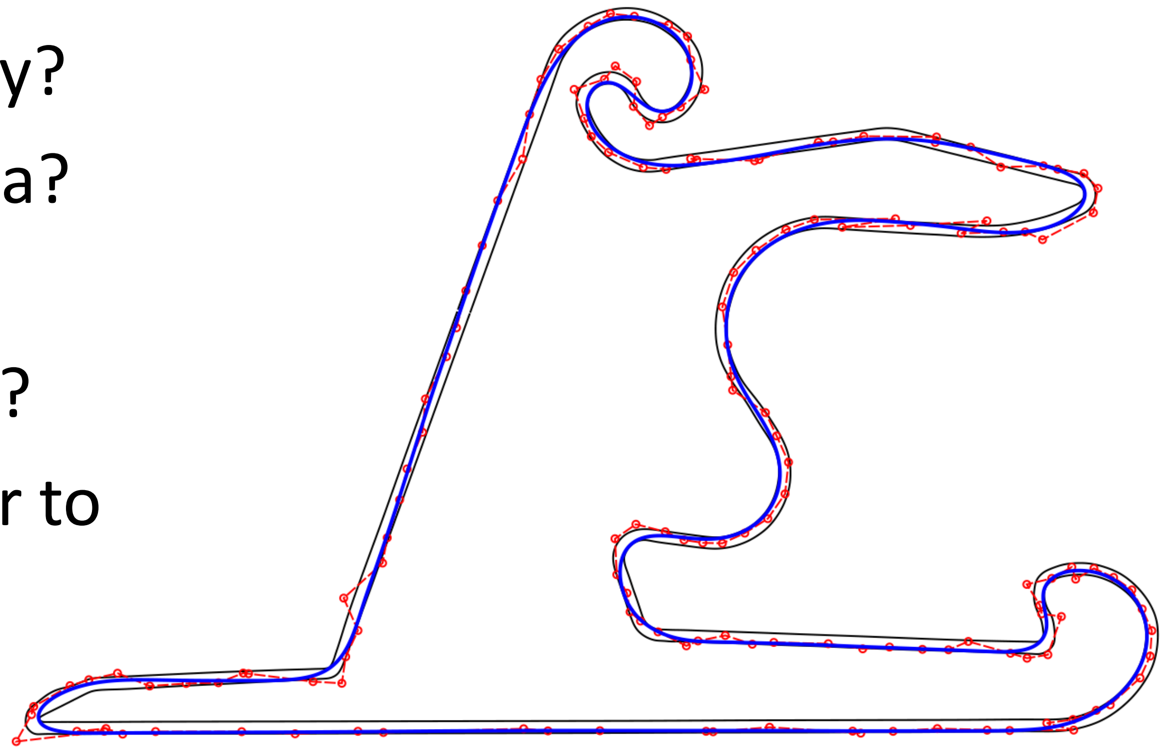
- No view of the whole track – must be conservative
- Can use traditional robotics map planners
  - Based on map
  - Usually safe and non-dynamic
  - Eg – A*
- Constrained environment – can make a specialised local planner
  - Eg. - midpoints of track



Global planning

Now that you have a map:

- Can you optimise a better trajectory?
- How can you process your map data?
- Hit all apexes?
- Take into account vehicle dynamics?
- How feasible is that path for the car to follow?



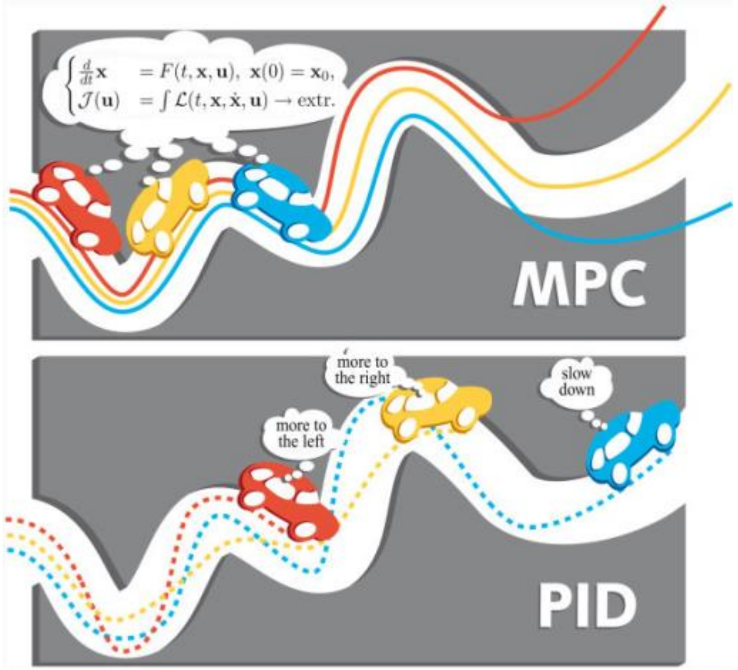
Control

- Once you have a trajectory, how can the car follow it?
- Software engineers can assume the car is a black box:
  - Steering
  - Speed
  - Torque
- Hardware engineers should:
  - Simplify control
  - Make it reliable

Control algorithms

- PID Controller
  - Trivial
  - Doesn't forward sample
  - Doesn't take into account dynamics model

- MPC Controller
  - Forward samples and optimises control
  - Based on costing
  - Requires knowledge of the dynamics of the car



https://openi.nlm.nih.gov/detailedresult.php?img=PMC2784347_cc8023-1&req=4

FORMULA STUDENT

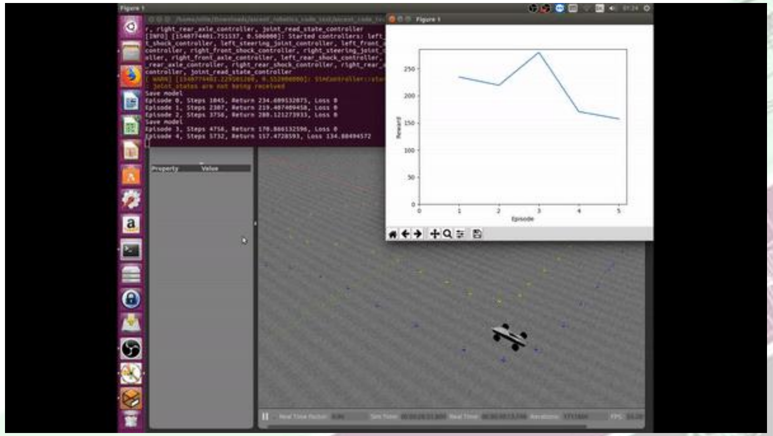
AI and Machine Learning

- Use breakthroughs in deep and reinforcement learning to control the car
- End to end – camera image input directly to control outputs
- Imitation learning
  - Record a human driving, model learns to copy the expert
- Reinforcement learning
  - Model has a go and given feedback on it's performance
  - eg. How far it got round track

Agent (the bit that learns)

Speed control

Steering control



Software tools, development and testing

- Middleware software eg. ROS
  - Message-passing between processes
  - Implementation of commonly used functionality
  - Package management
  - "Nodes" (programs) in most programming languages – C++, Python, MATLAB etc.
- Repo management eg. GitLab
  - Code reviews
  - Software testing – continuous integration
  - Version control

Software tools, development and testing

- Simulation for developing algorithms / ideas eg. Gazebo
  - Ours (with FSUK DDT car model) is open sourced:
  - https://github.com/eufsa/eufs_sim
  - https://github.com/Microsoft/AirSim/wiki/technion
- You can develop a small-scale testing platform that can be tested anywhere(eg. a wheelchair)
- Datasets for verifying and evaluation on more realistic data:
  - Ours on wheelchair and from 2018 FSUK competition is shared:
  - https://github.com/eufsa/datasets
  - https://github.com/AMZ-Driverless/fsd-resources

To prepare for the competition

- There are a lot of cool things you can do but always justify why you are doing them!
- Always think of how you can generalise your solution! 90% of the things you do on the car is also being done in industry right now
- Have clearly defined tasks for all members attending
- Have a cool Plan A
  - But if that fails be prepared with Plan B

Problems

- It's FS – problems are always around the corner
- Good idea to already have redundancy
- Draw up a list of things that can go wrong – they probably will!

Sponsors & QA