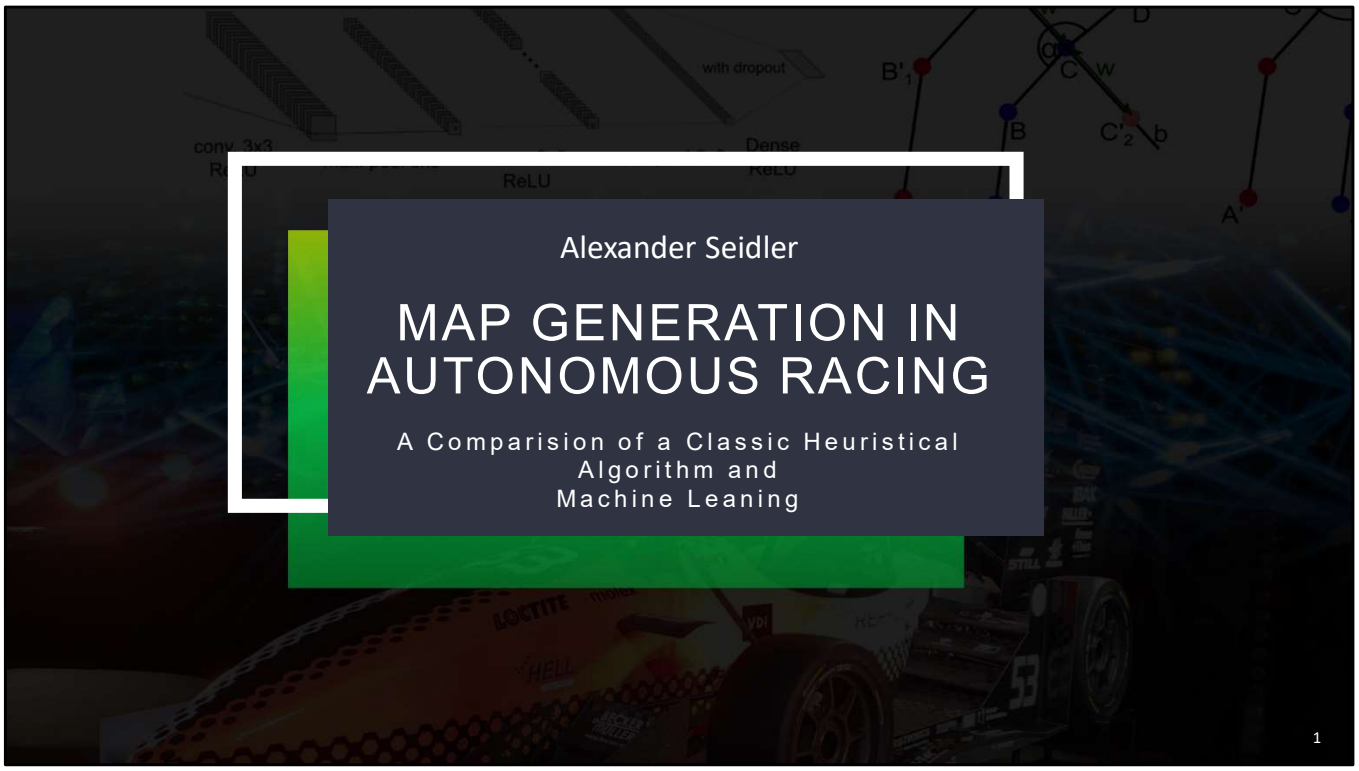
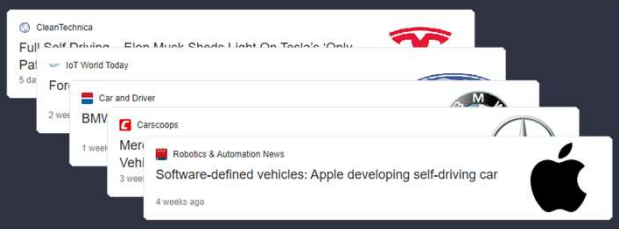


Hello, Like to present to you my bachelors theses: Map Generation in Autonomous Racing - A Comparision of a Classic Heuristical Algorithm and Machine Leaning

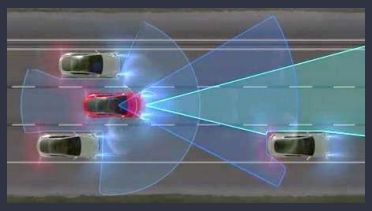
WHY MAP GENERATION?

- Major car brands looking into self driving



- Implement autonomous driving in Raceyard too

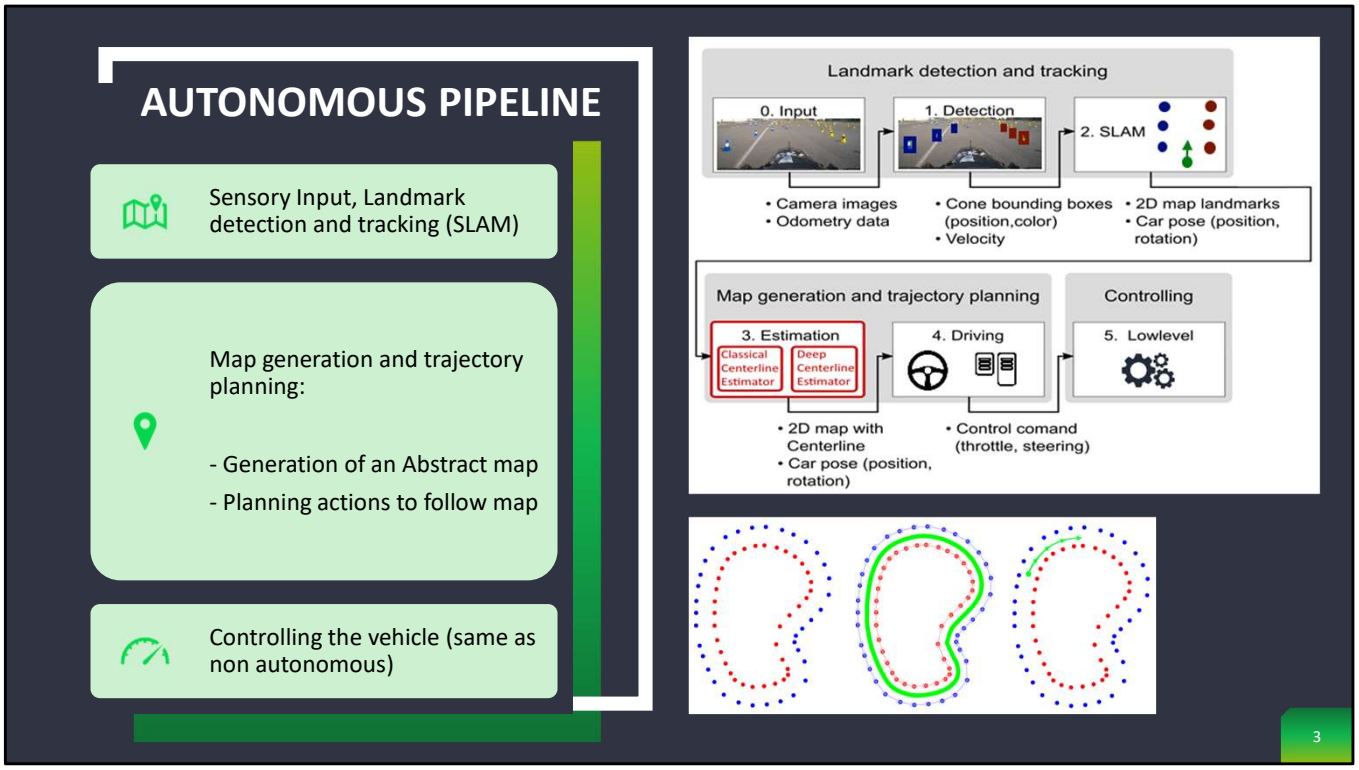
+



2

So why map generation, and what to do with autonomous racing?
Having look at major car manufactors, actively looking into autonomous driving. It just takes a look at the recent news to see that tesla, ford, bmw, mercedes, even apple recently joined efforts to develop fully autonomous vehicle to be used on streets within the next few years.

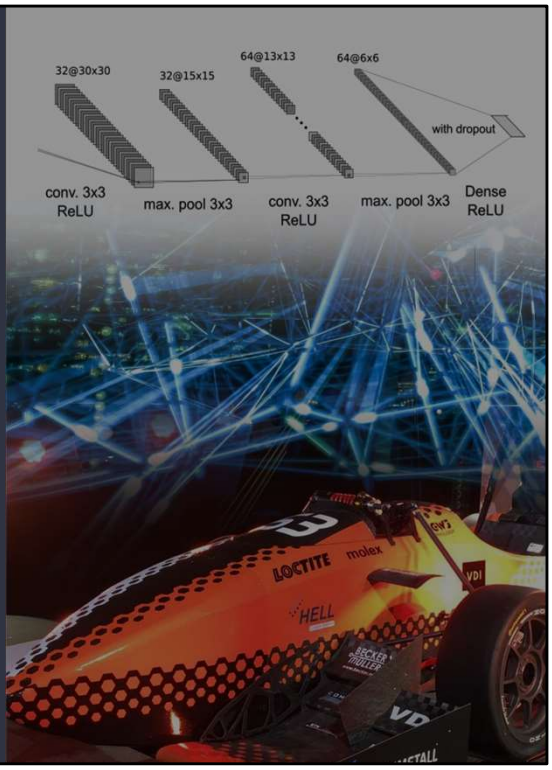
How cool implement this technology at our local formula student team raceyard. To make their race cars race autonomously along their tracks
That's goal raceyard, to compete in driverless discipline in formula student, to make this car [photo], race autonomously without any driver
So, what do need, to make this car fully autonomous?



Can be split into 3 parts, sensory input, the detection [follow onscreen text] [compare to pipeline]
The part marked in red, my contribution, to visualize, basically takes map, classical calculates out centerline, ml curvatures that model course ahead

OUTLINE

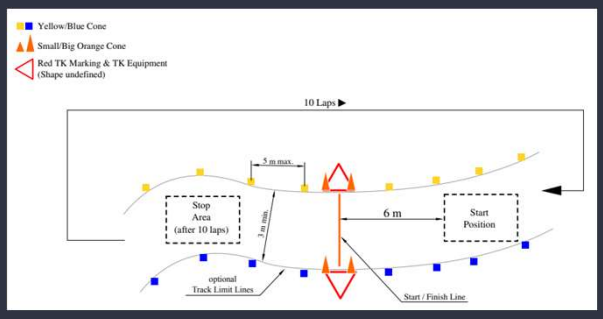
- Goals (Raceyard)
- Foundations
  - Machine Learning
  - Discrete Curvature
  - Development environment
- Methods
  - Classical approach
  - Machine learning approach
- Evaluation
  - Analysis
  - Comparison
- Conclusion

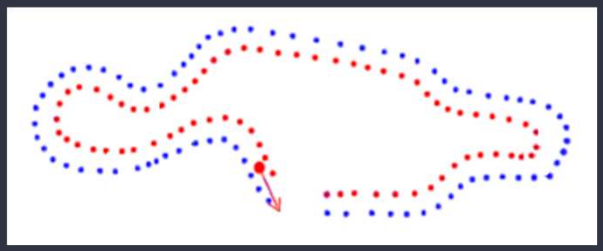


Now that you, hopefully understand that my work good for, lets talk about these to approaches lil more detailed,
details the goals more precisely that this thesis alongasde implementation at raceyard trys to fulfill.
To understand the methods I used for implementing the map generation, I will briefly introduce some foundations, alongaside the development environment used to prototyping the ideas
After weve established that, I will present the two approaches ive taken, whose performance I will analyse and compare to draw a conclusion how each implementation could be used in future

GOALS

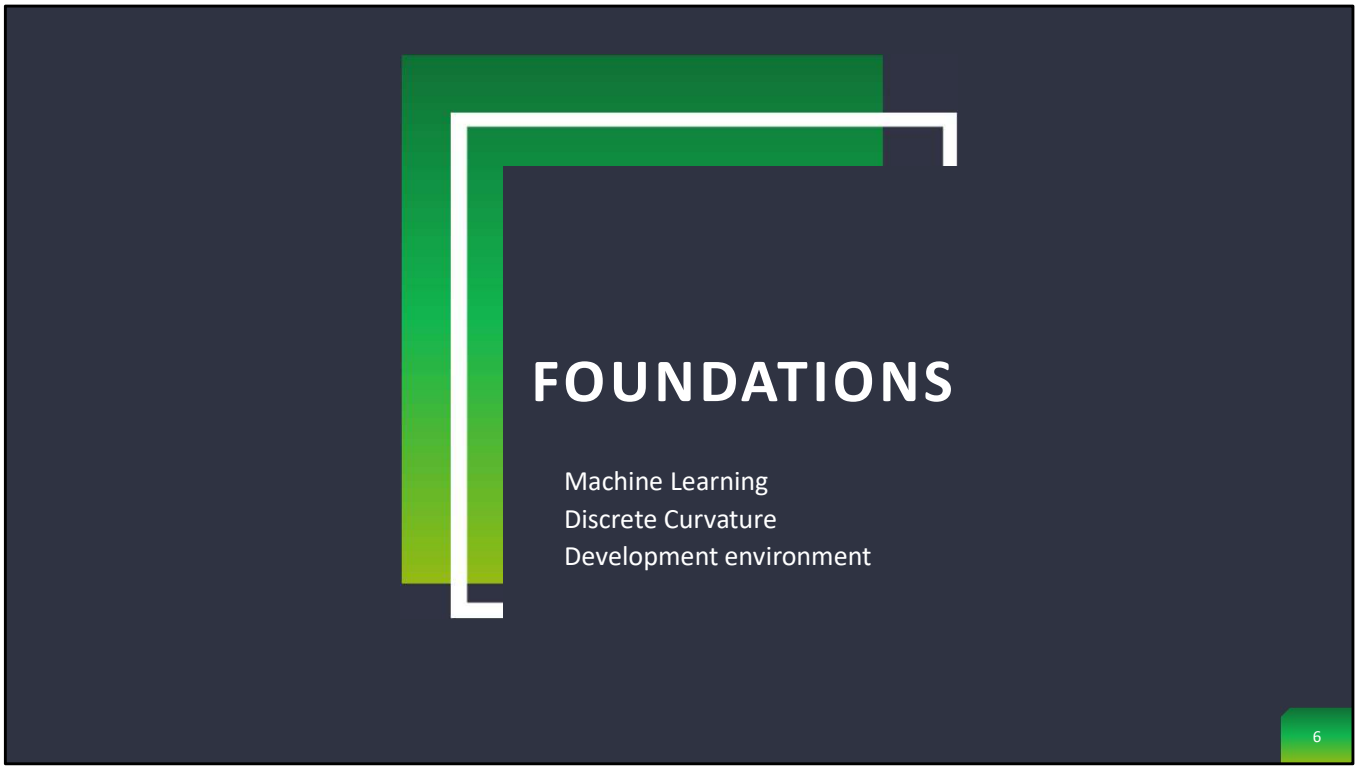
- Solve map generation, necessary to drive autonomously in FSD
- Improve on existing classical algorithm
- Implementation of novel ML algorithm
- Goals:
  - Robustness
    - Misdetection
    - Non-detection
    - Over-detection
  - Better Runtime
  - Better utilization of SLAM Information





5

Main goal established already, we need a map.
Furthermore, improve on existing previously worked classical approach to map generation
But also explore new machine learning aproach
In regards to the implementation, they the improved classical one and ml one, should have better robustness against mistdetection of cones, for exmaple wrong color, non-detection and over detection of cones
Also a better runtime so that the algorithms could be used in real time
And better utilization of all the data the slam provides



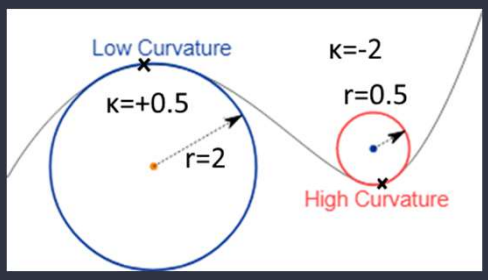
Lets talk about the foundations for my aproaches

MACHINE LEARNING

- Algorithms that can learn
- specifically supervised learning
- Multilayer perceptron
  - neural network with multiple layers
- Deep learning
  - e.g. stochastic gradient descent, using backproagation
- Convolutional Neural Networks (CNN)
  - Primarily used for image input
  - Adds convolution layers, and pooling layers
- [image of cnn]

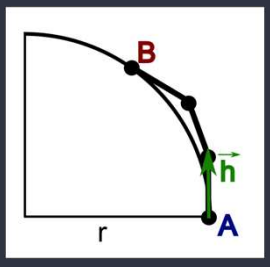
[yeah need to work on this]

CONTINUUS CURVATURE



- 1 / radius of touching circle
- Sign indicates “direction”

DISCRETE CURVATURE



- Defined by two points, A and B and a heading in A
- Used to approximate course of a polyline

One important concept is curvature, curvature at a point on a continuus curve can be described by the circle that touches the curve in that point,
Also the sign indicates the cirection of the curvature, when ist positive ist going right, negative left, when read from left to right
When we have a discrete curve, such as a polyline, we need a more advanced definition, since the curvature alonge the straight parts would be 0 and in the vertices infinite
We can use 2 points plus a heading, with both points being on the circle and the heading being tangent, the heading can be derived by looking at the next point,
, this also gives us a quasi touching circle and by that a curvature for this discrete polyline from point A to B

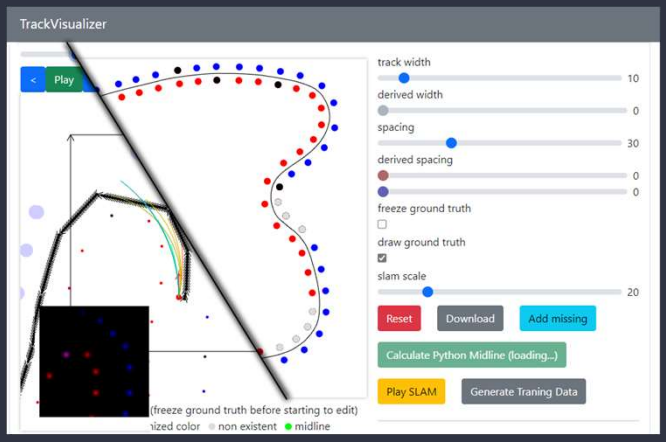
DEVELOPMENT ENVIRONMENT

In Web technologies, HTML/CSS/JavaScript

- Easy prototyping and sharing

Uses pyodide to execute python using WASM

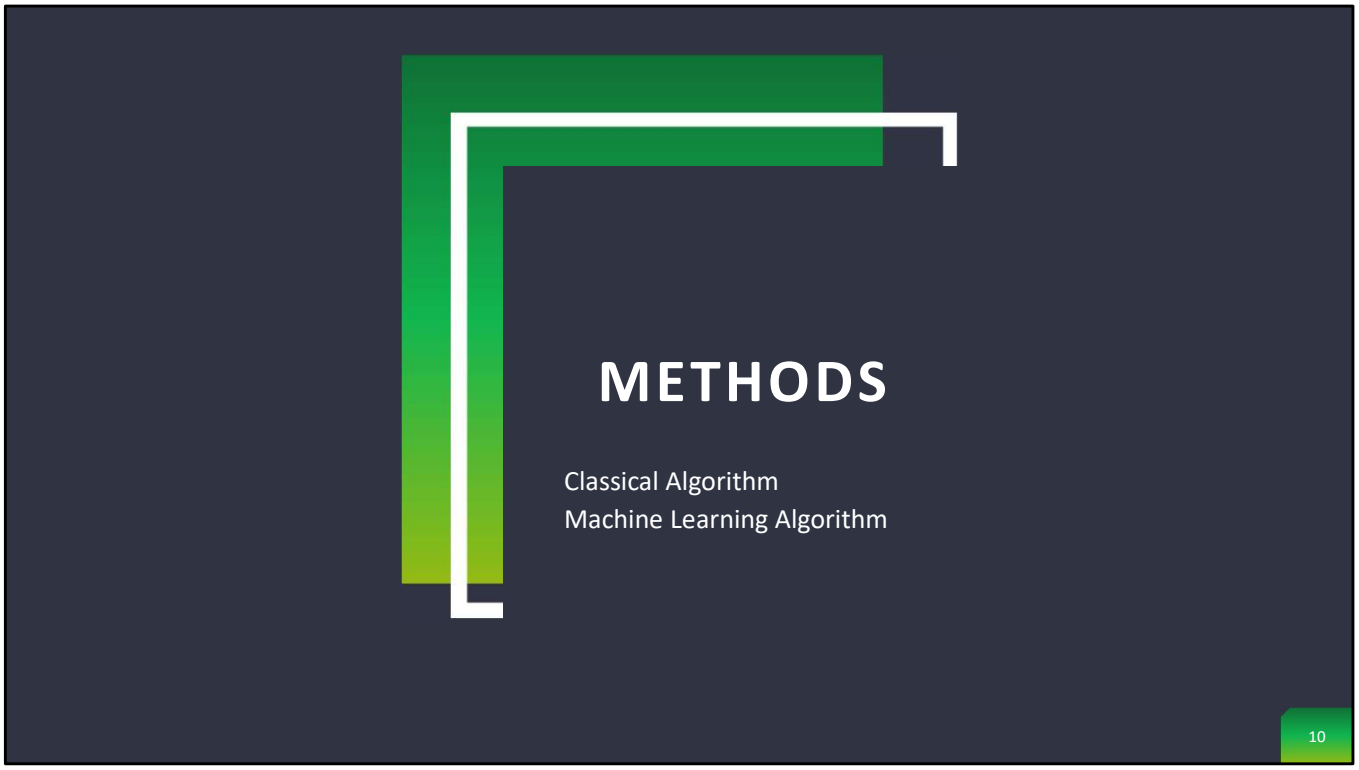
- Completely statically hostable, no backend need e.g.: github pages[1]



[1]https://dsalex1.github.io/BachelorThesisRaceyard/

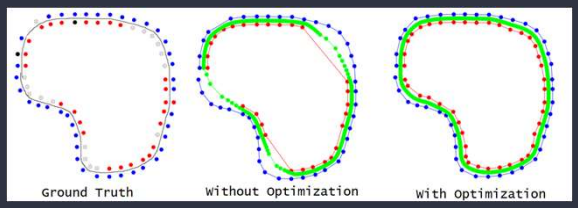
9

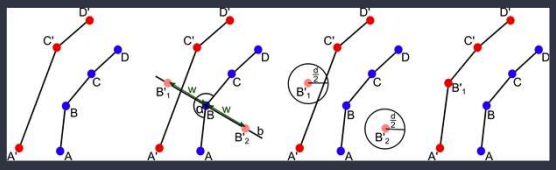
Showcast development environment i programmed to work on this project, because this is where effectively spend most time, prototyping and evaluating ideas, written in web technologies, allows for rapid prototyping and sharing, since pipeline and existing is in python, we use pyodide, which allows python execution in the browser using web assembly, notabily no cross compilation, and having support for many libraries, pandas, scipy – and such statically hostable for example at github pages, this url you can go and try the latest code i used for prototyping

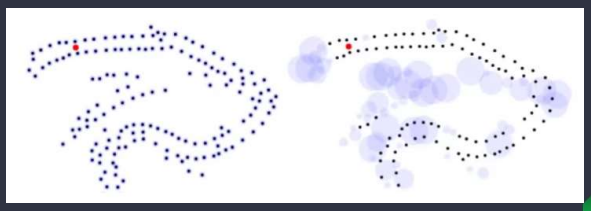

METHODS
Classical Algorithm
Machine Learning Algorithm
10

CLASSICAL ALGORITHM

- Basis: Master Project by Vaishnav/Agrawal
- Given pointcloud from slam calculate centerline
- Improvements
  - Spacial sorting using TSP Approximation
  - Readding of missing points
    - For each cone: guess position of corresponding cone
    - If none there add the guessed one
  - Incoorparaion of covariance

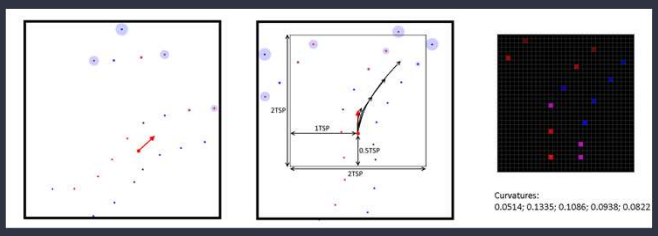




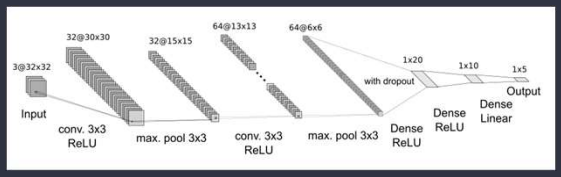


MACHINE LEARNING ALGORITHM

- Input/Output Design
  - Input: Immediate surrounding cones as image
  - Output: curvatures describing the course of track ahead



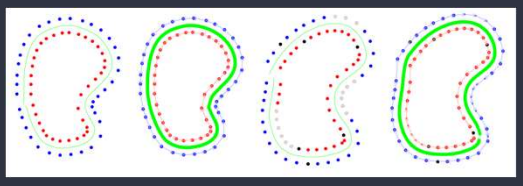
- Standard CNN architecture



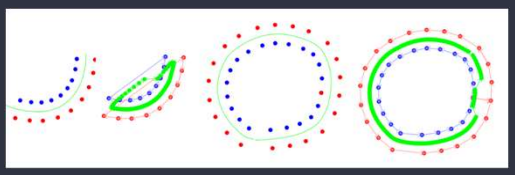

EVALUATION
Analysis
Comparison
13

ANALYSIS – CLASSICAL

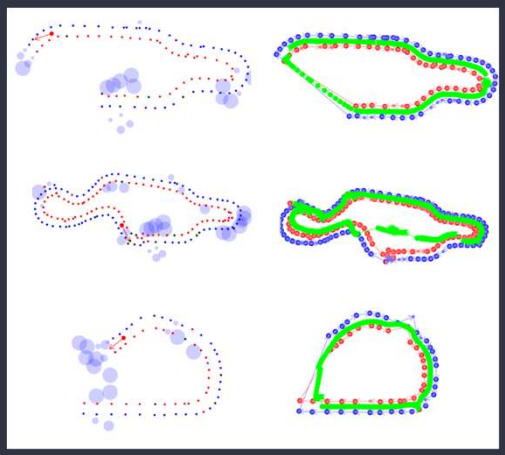
- Works well for simple cases:
  - non-detections
  - arbitrary cones sorting



- Fails for more erroneous data
  - Incomplete tracks
  - misidentified colors



- With Simulated data
  - Sometimes good/sometimes not at all

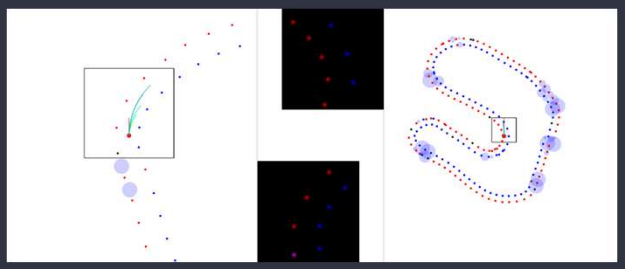


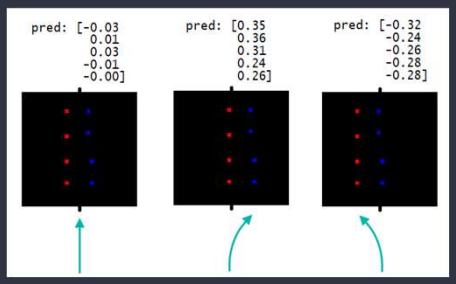
14

All in all, less and less usable output increasing errors in the input, but if input very precise

ANALYSIS – ML

- Neural Network successfully drives around track
- Very error resilient





- Deviation from track center automatically corrected
- Centerline of whole track cant be easily determined

15

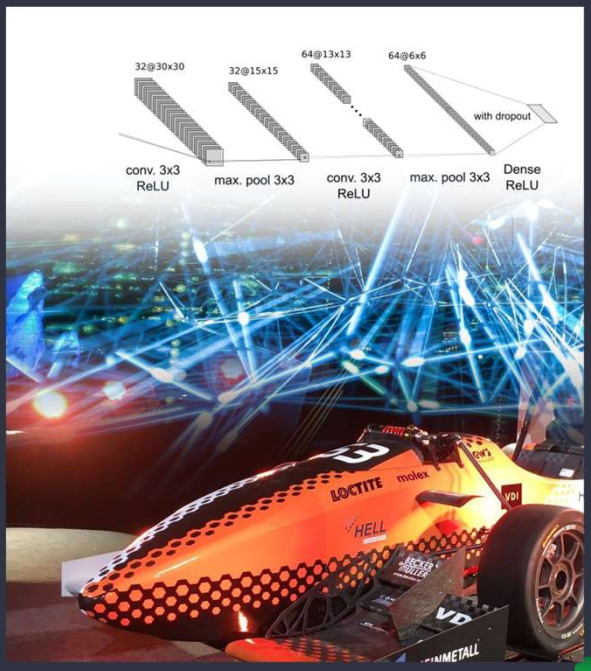
Classical algorithm:
All in all, less and less usable output increasing errors in the input, but if input very precise

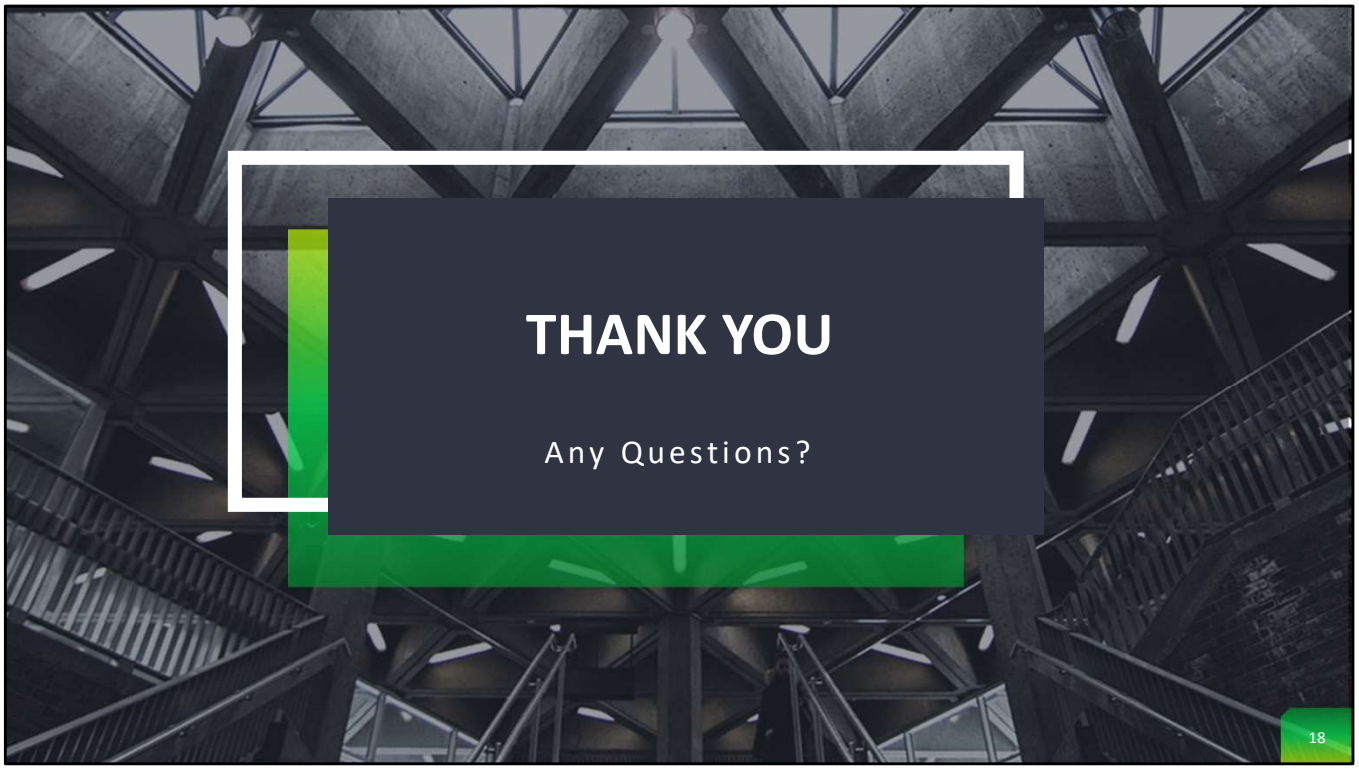
COMPARISON

Machine Learning	Classical
• High noise tolerance	• Very brittle
• Produces approximative local centerline	• Produces complete accurate centerline or unuseable output
• Runtime yields ~16FPS, consistent	• Runtime yields ~1.6FPS, depending on number of cones

CONCLUSION

- Improvements made classical approach robust
- ML useable for real life tests, runtime suitable for realtime, can use in 1st round
- Classical map generation once 1st round completed





[show track 1 with 26 mb]