From Data to Action

Data & Telemetry
BOOTCAMP

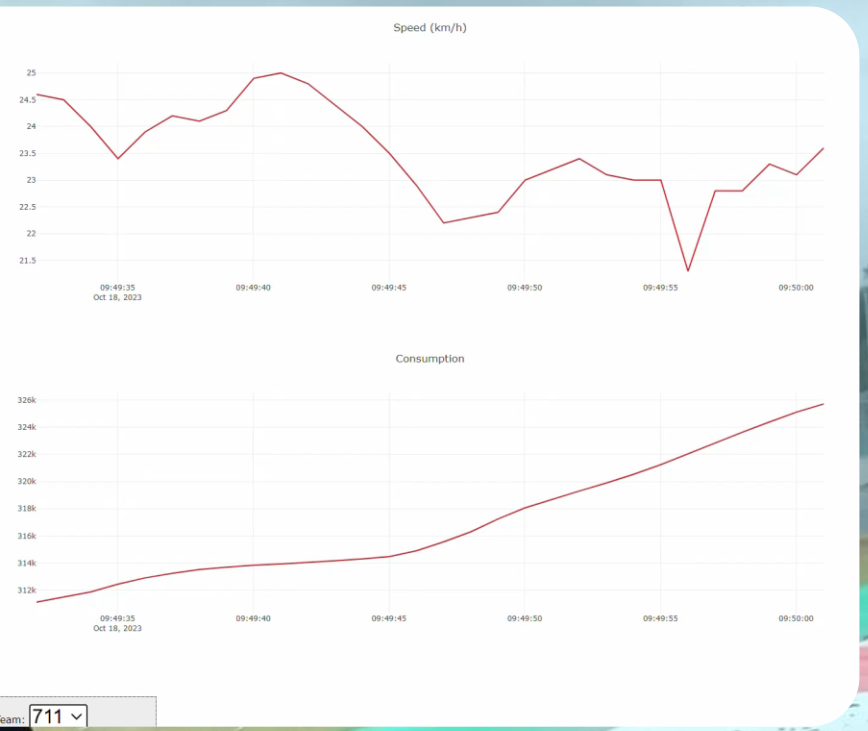
Marco Schmid
CEO and Head of R&D
Schmid Elektronik

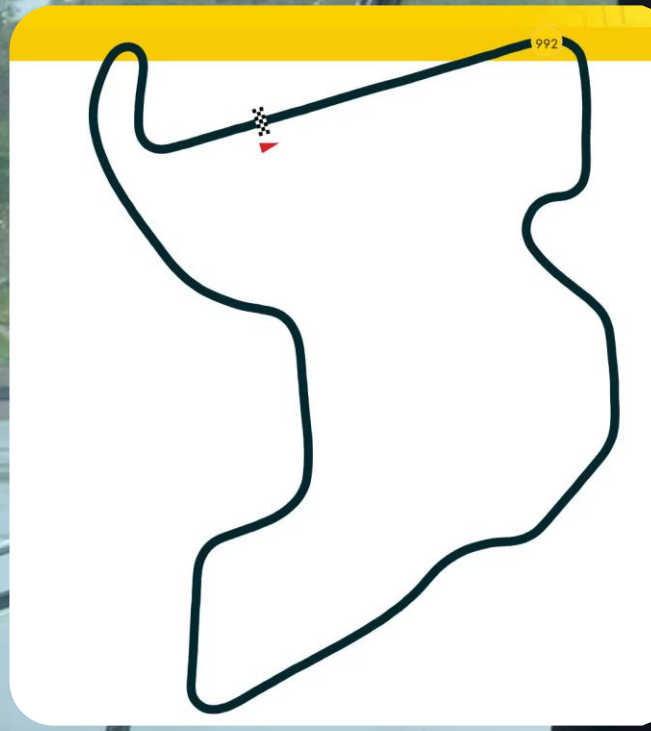
SCHMID Elektronik

Shell Eco-marathon Partner

26.6.2026, SEM Silesia Ring, Poland

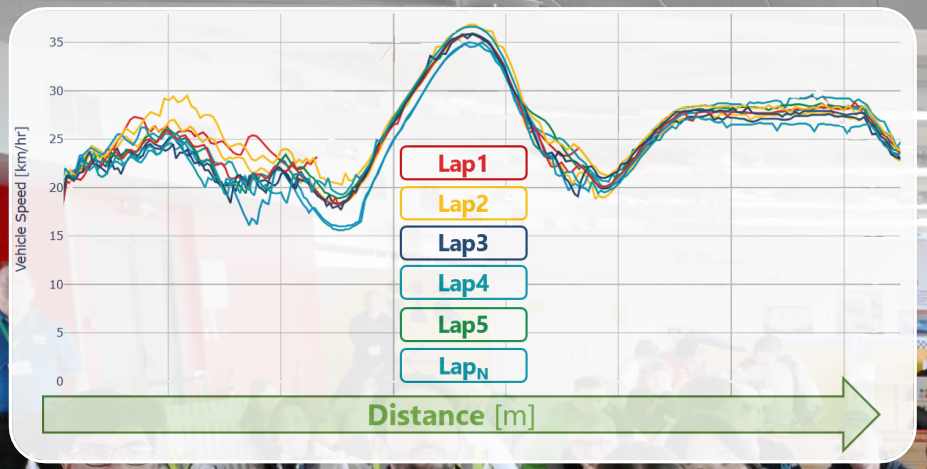
From the Drivers World Chamionship...





… to the Mileage Challenge

Marco Schmid



Schmid Elektronik is **Shell Eco-marathon Partner**

About Shell Eco-marathon

For over 40 years, since 1985, the programme has consistently brought to life Shell's mission of Powering Progress together by providing more and cleaner energy solutions.

Watch the NEW film to learn more about Shell Eco-marathon'

Read more →

2026 Programme

The season opens a new chapter that reflects the changing face of STEM and introduces a comprehensive competition experience, leading to a truly Global Championship.

Learn more about the 2026 Programme here

Read more →

Partners

Our partners are essential to Shell Eco-marathon. Not only do we work together, but we combine resources and share ideas and innovations to help build a lower-carbon future.

SCHMID Elektronik | Shell

Shell Eco-marathon Partner

Read more →

NEW 2026

Shell Eco-marathon **Telemetry System**

NEW 2026

Logistics Nightmare

10 Years → OBC NXG

~~One Size fits all~~?

Shell Eco-marathon **Telemetry System**

Changes of Telemetry Focus: all Teams equal

All Energy Types

- ICE
- H2
- BE

For both Vehicle Categories

Data & Telemetry **Off-Track Award**: Developing a Strategy that:

#1 Accounts for the straights, slopes, crests and corners of a track

#2 Balances maximum energy efficiency with competitive lap times

#3 Ensures driver safety in all conditions, including rain and wind.

A **Six**th OTA-Question

NEW 2026

1. **Data Strategy** to Achieve the Three Goals
2. Capturing Data with a **Telemetry** System
3. Gaining **Knowledge** from Race Data
4. Data-driven **Race Strategy** Development
5. **Driver** performance on Track
6. Qualitative and Quantitative **Results Improvement**

Publishing the OTA-Winners

NEW 2026

Data and Telemetry Award sponsored by Schmid Elektronik

The Data and Telemetry Award celebrates teams that harness the power of data to optimise vehicle performance and enhance overall strategy.

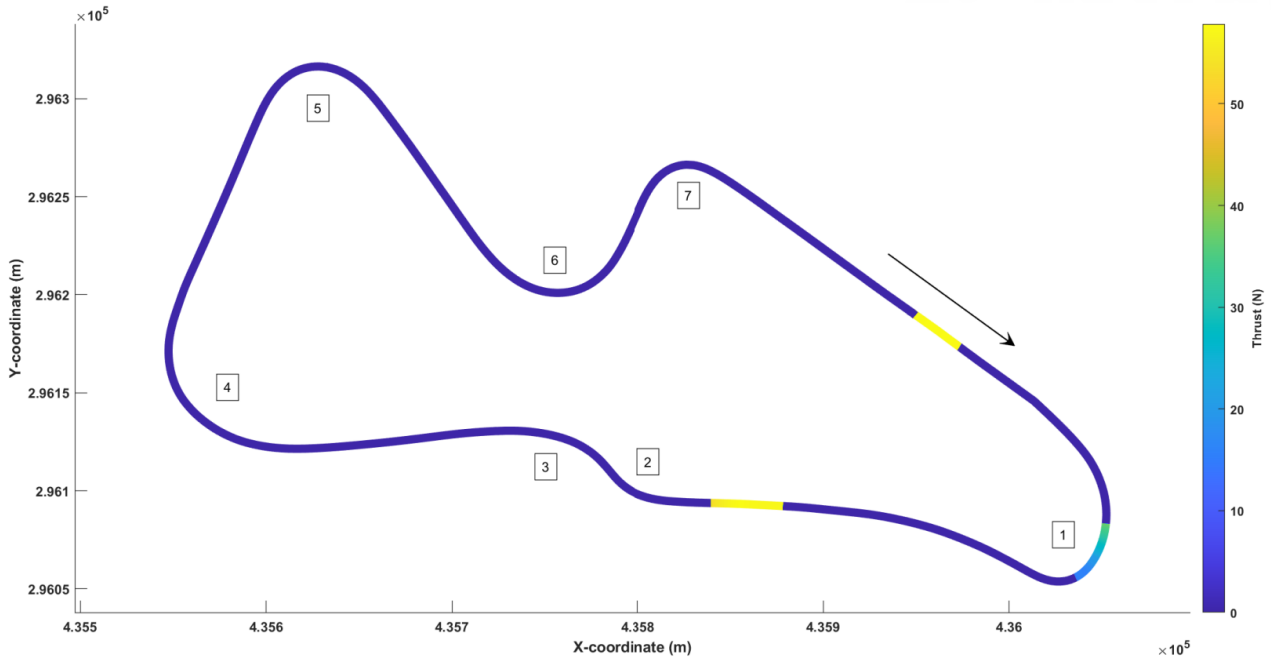
Winner: Imperial Eco-Marathon, Imperial College London, United Kingdom

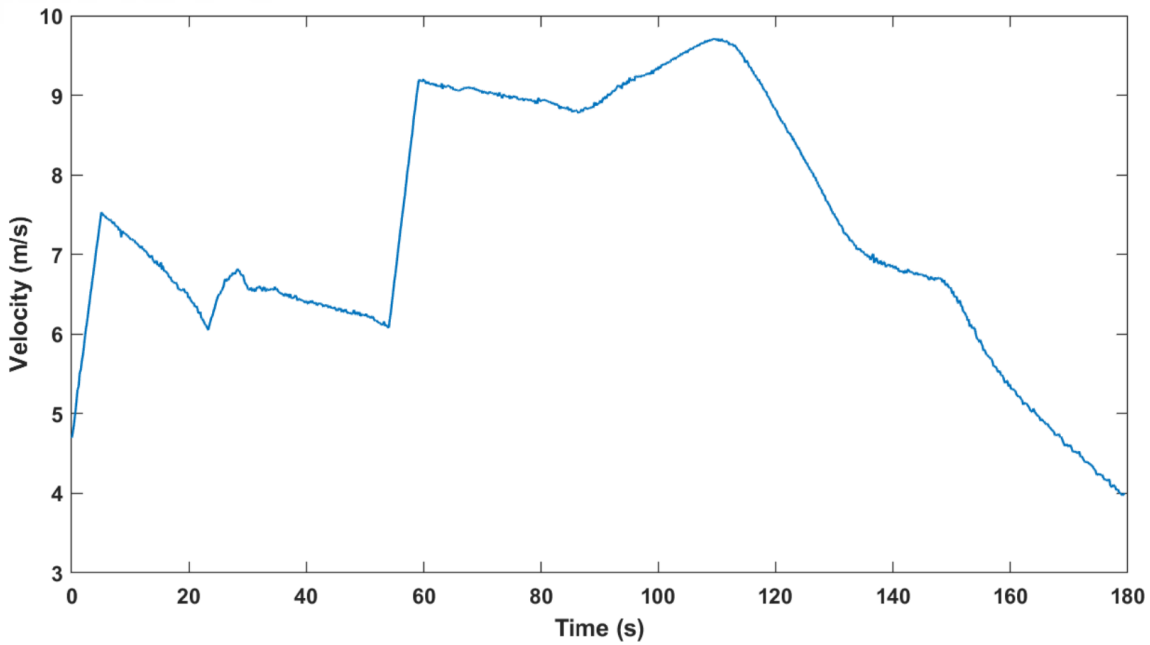
Judges' Remarks: Rigorously engineered telemetry and simulation underpin this entry. Instead of chasing a single perfect lap, the team optimizes the entire multi-lap attempt, preventing over-tuning of a single lap. A predictive throttle map and speed profile give the driver foresight, while live telemetry refines tactics in real time— a holistic, adaptive and data-driven masterclass!

Runner-Up: PROMETHEUS ECO RACING NTUA, National Technical University of Athens, Greece

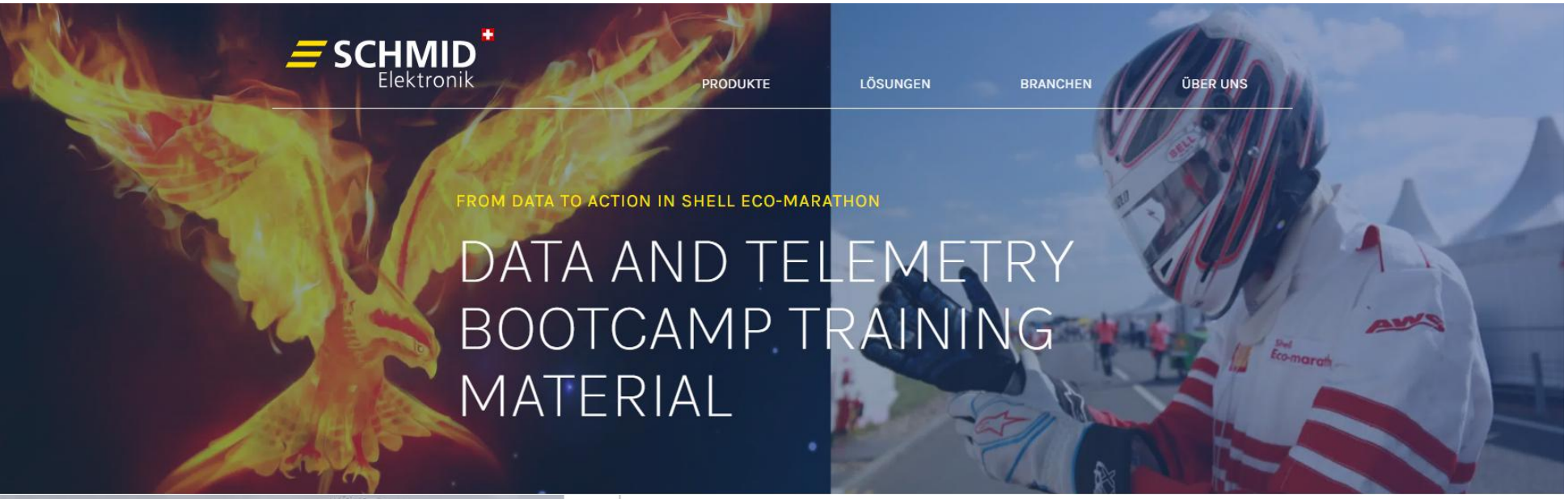
Read Imperial Eco-Marathon's winning entry (PDF, 4 MB)

Judges' Remarks: Rigorously engineered telemetry and simulation underpin this entry. Instead of chasing a single perfect lap, the team optimizes the entire multi-lap attempt, preventing over-tuning of a single lap. A predictive throttle map and speed profile give the driver foresight, while live telemetry refines tactics in real time - a holistic, adaptive and data-driven masterclass!





https://schmid-elektronik.ch/bootcamp-sem26/



Bootcamp Structure

YOU PROGRESS FROM LEVEL TO LEVEL, JUST LIKE IN A GAME.

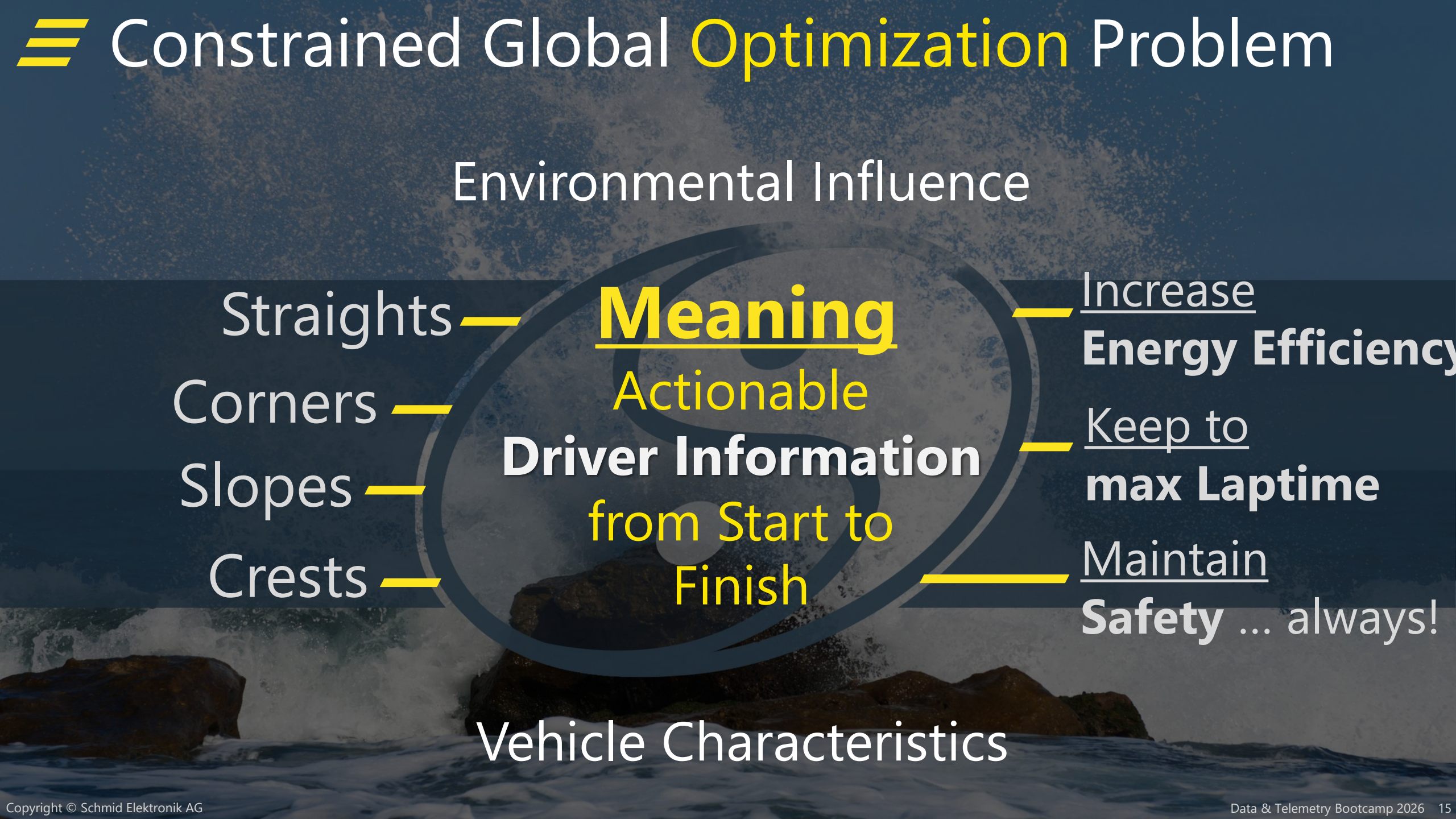
In five steps, you will learn how data turns into knowledge and knowledge turns into action.

Level 1 | Telemetry: How to Create relevant Race Data on Track?
Level 2 | Data Analytics: Recognize Correlations and Patterns
Level 3 | Live Data: Send Data from the Track to the IoT
Level 4 | Modelling: Maintain Physics Based Digital Twin
Level 5 | Holistic: Advanced Data-Driven Racing

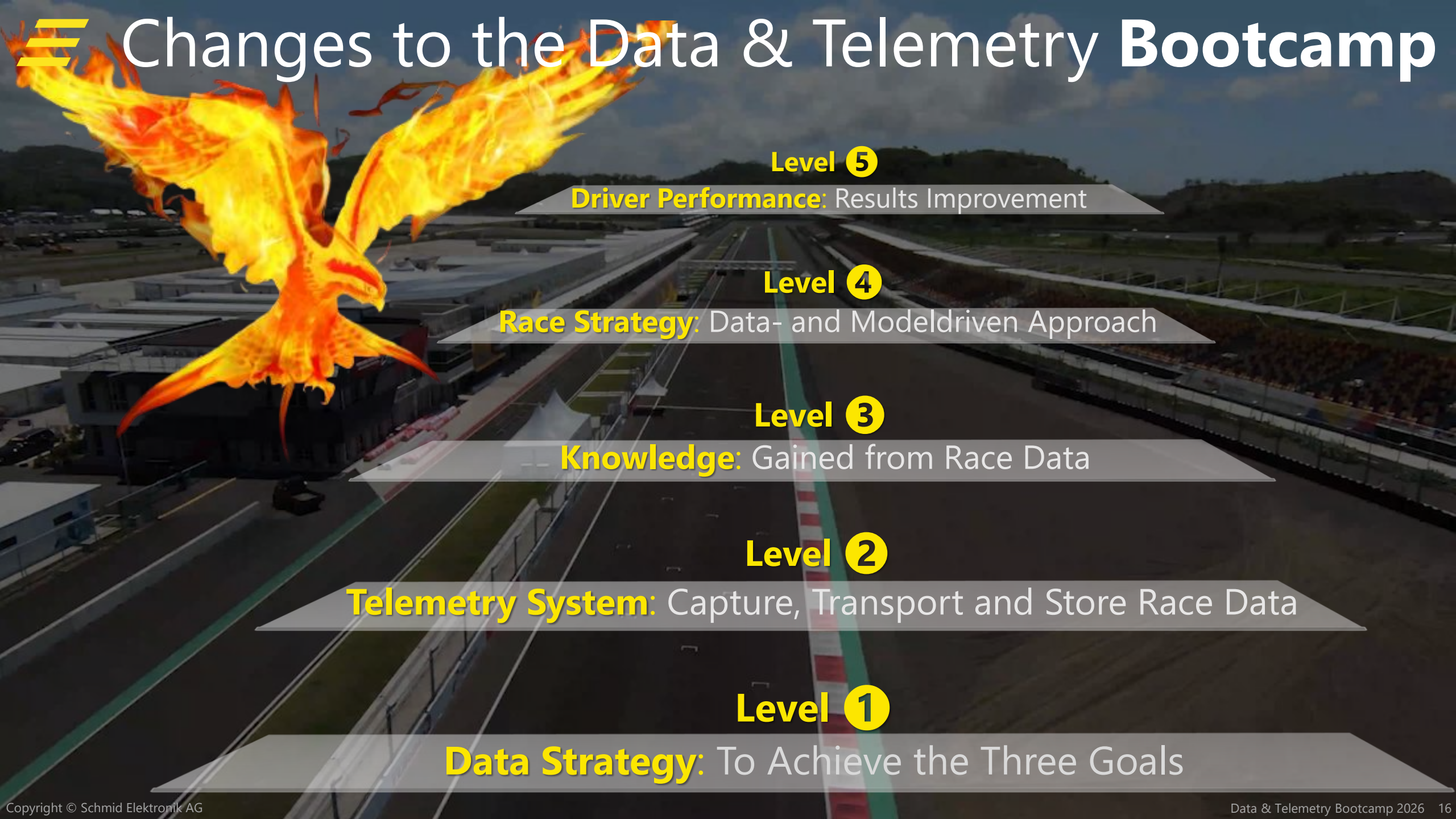
Constrained Global Optimization Problem

Why move from intuitive to data-driven Racing?

Constrained Global Optimization Problem



Changes to the Data & Telemetry **Bootcamp**



Level 5

Driver Performance: Results Improvement

Level 4

Race Strategy: Data- and Modeldriven Approach

Level 3

Knowledge: Gained from Race Data

Level 2

Telemetry System: Capture, Transport and Store Race Data

Level 1

Data Strategy: To Achieve the Three Goals

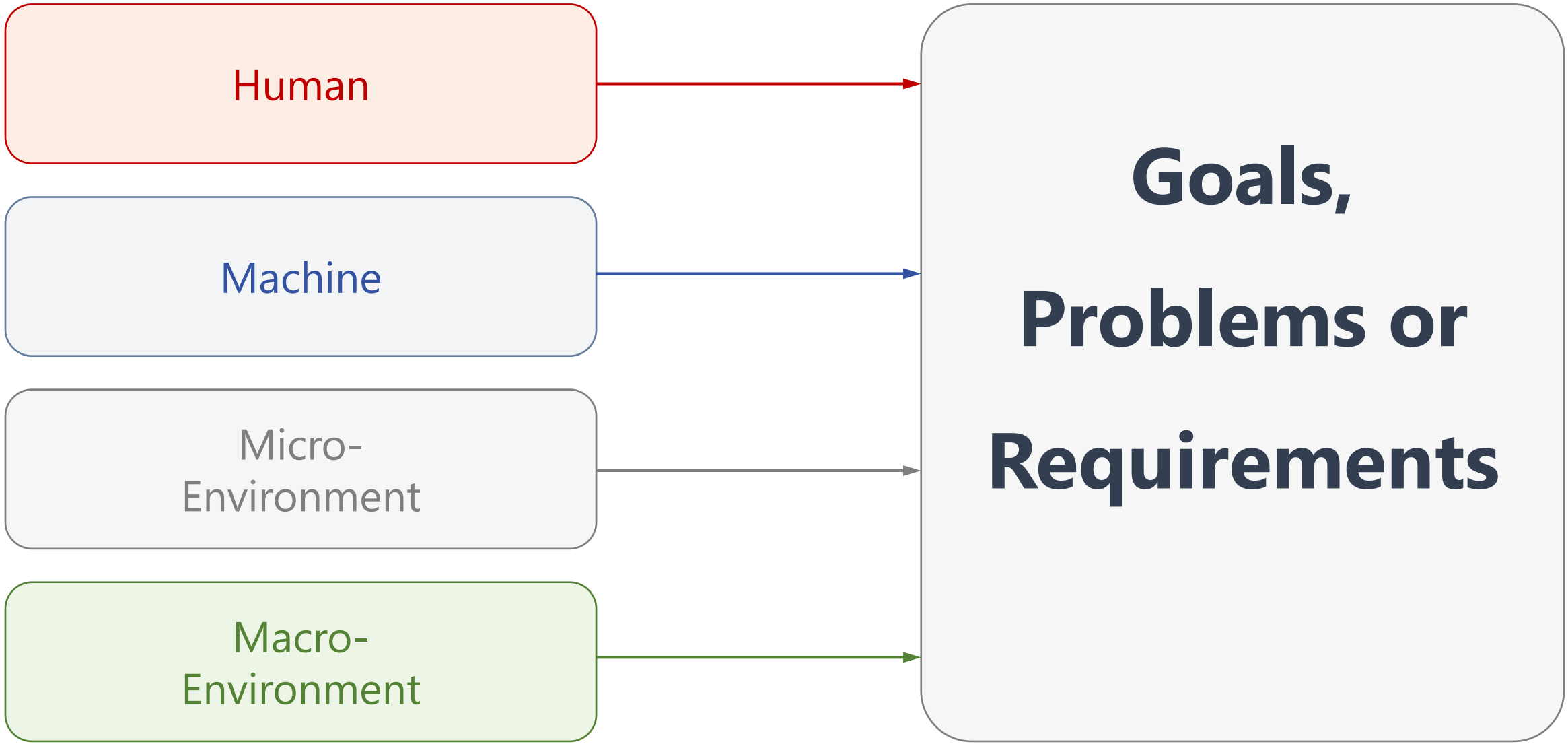
Data Strategy

to achieve the 3 Goals:

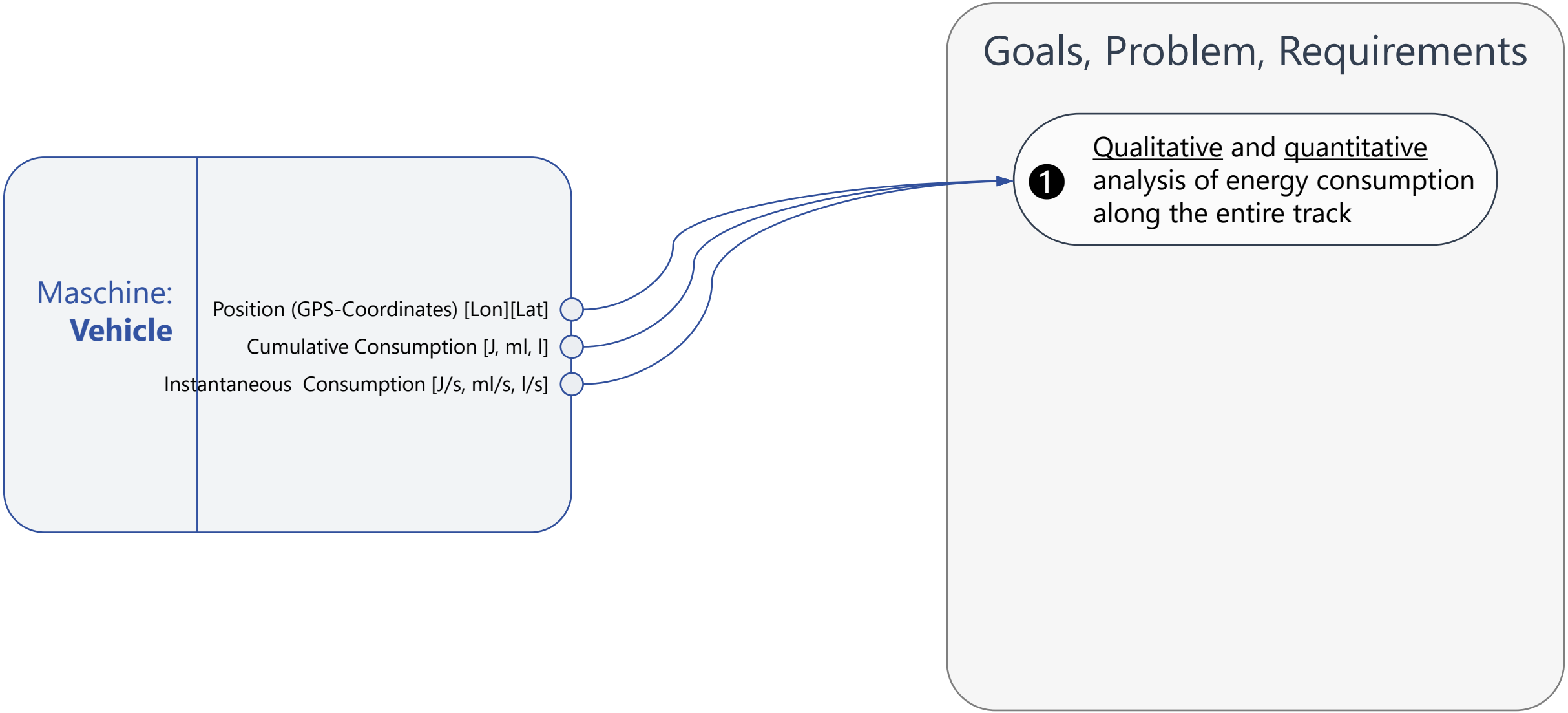
1. Account for straights, slopes, crests and corners
2. Balance maximium energy efficiency with minimum lap times
3. Ensure driver safety in all conditions, including rain & wind

Level 1

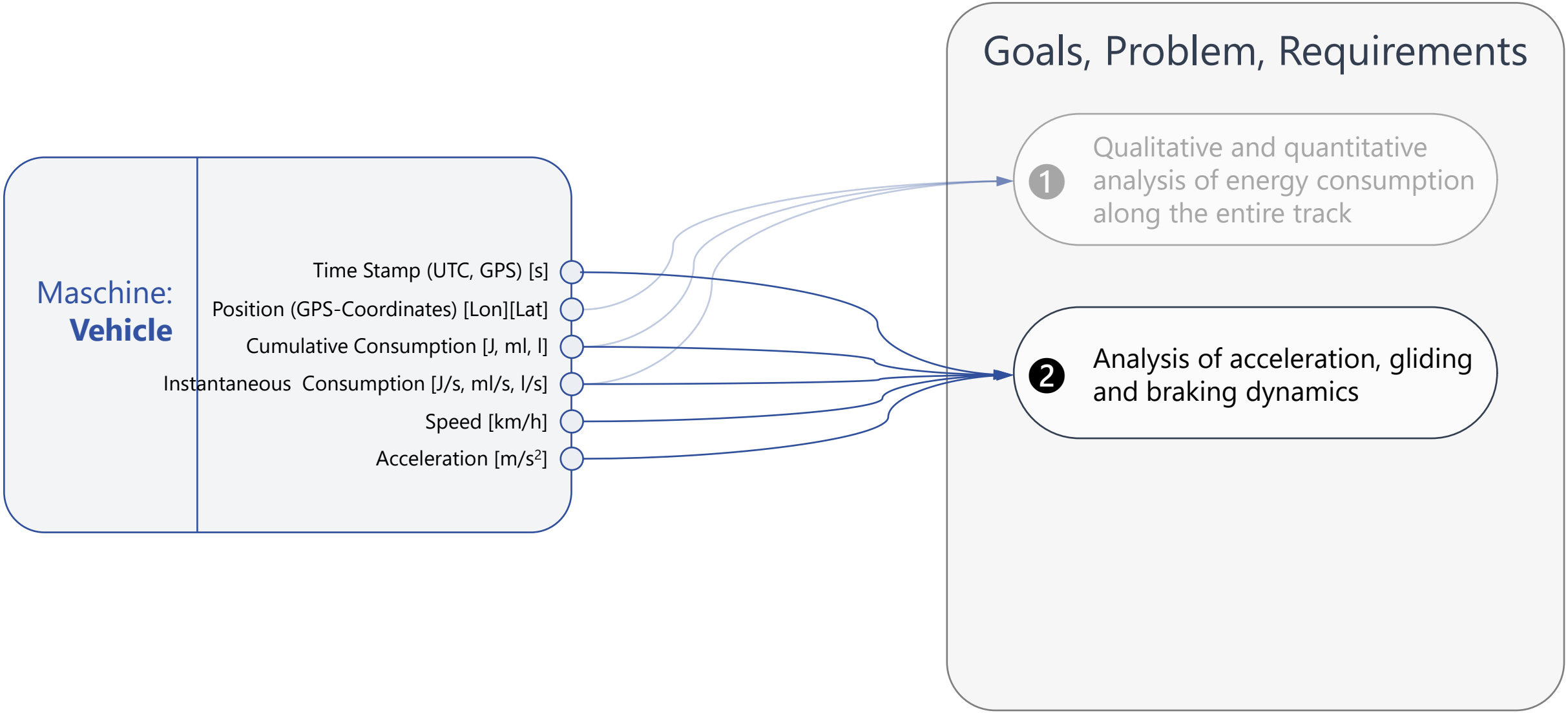
What is your **Data Strategy** + relevant Data?



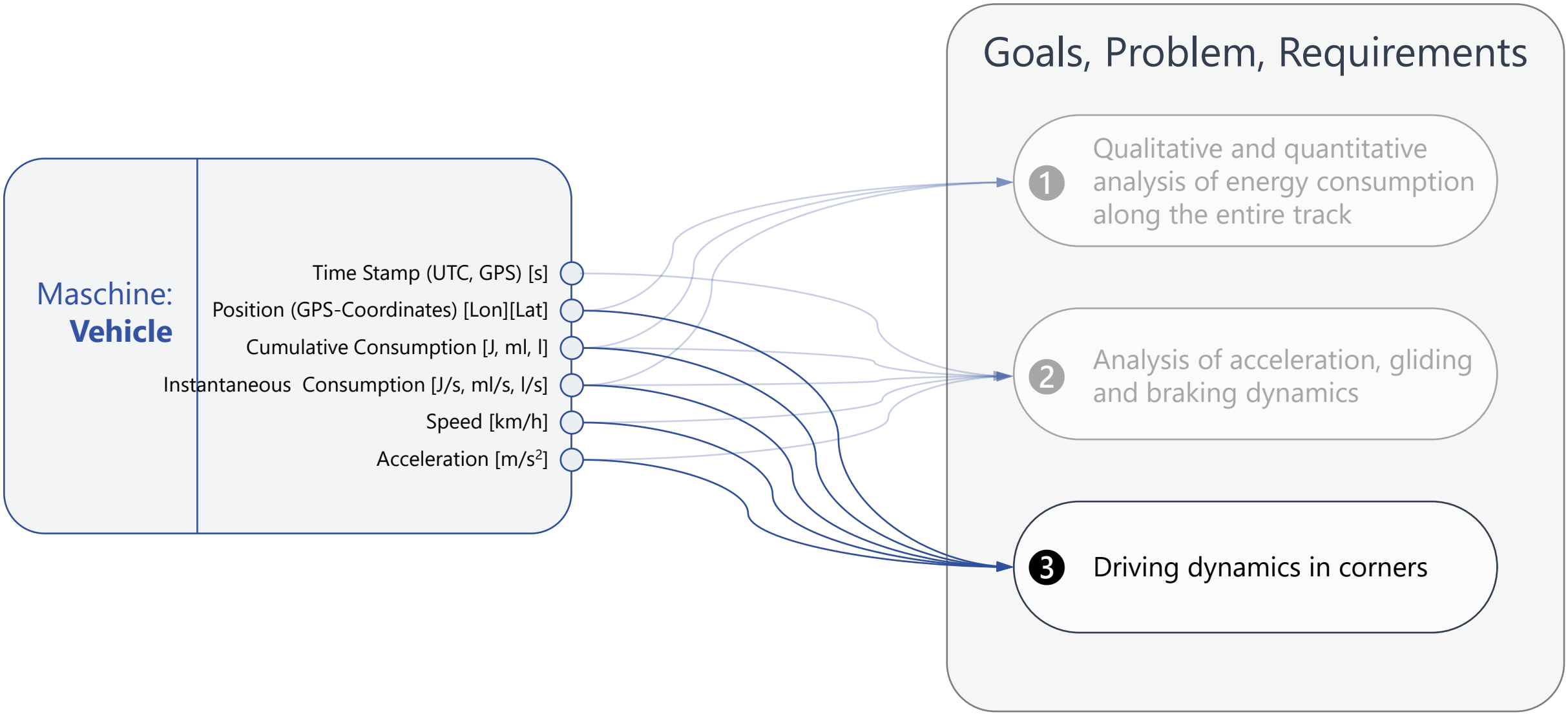
What is your **Data Strategy** + relevant Data?

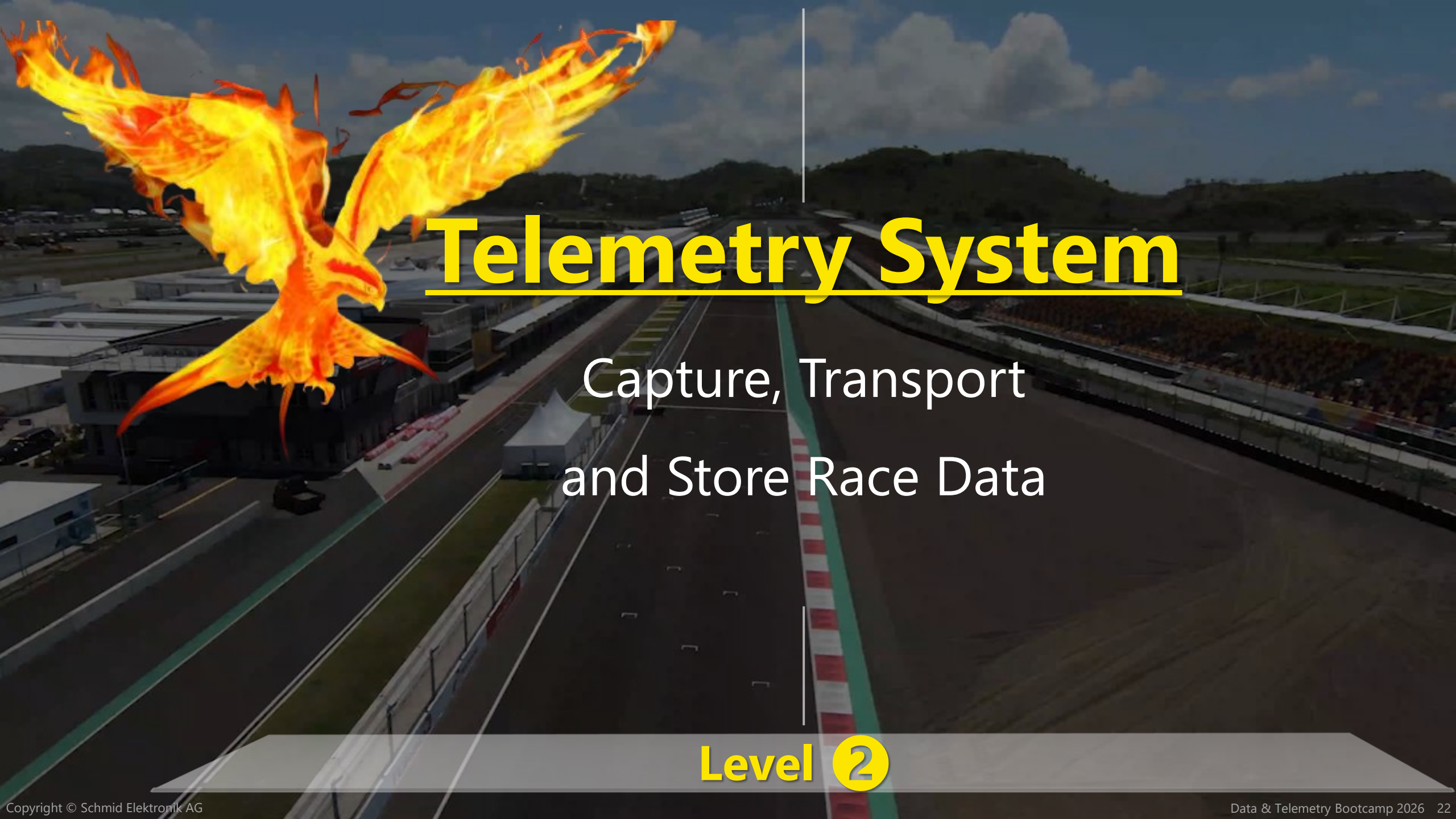


What is your **Data Strategy** + relevant Data?



What is your **Data Strategy** + relevant Data?





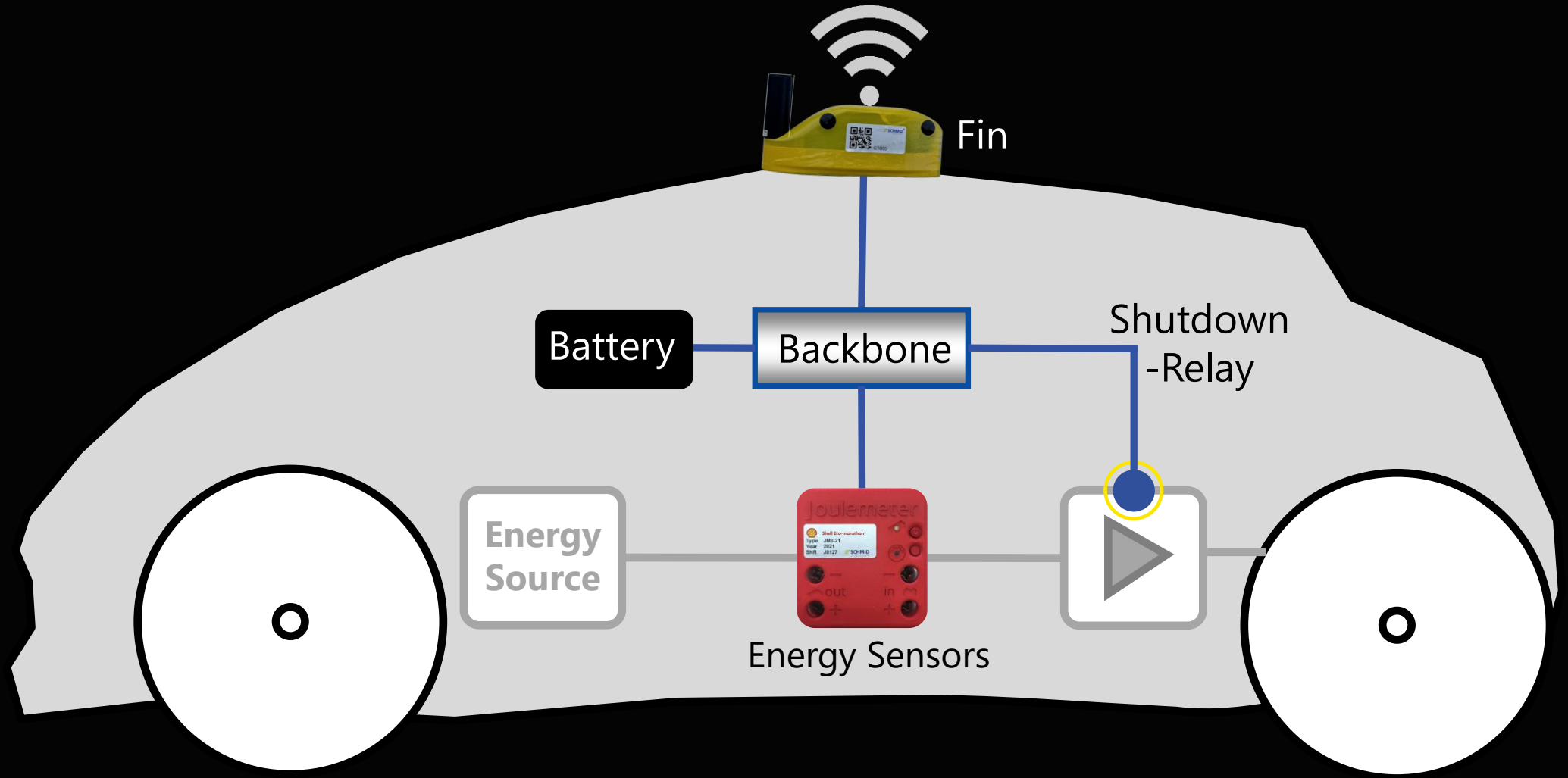
Telemetry System

Capture, Transport

and Store Race Data

Level 2

```
imx6@obc:~$ mosquitto -c /etc/mosquitto/mosquitto.conf -d
imx6@obc:~$ mosquitto_sub -h localhost -v -t 'device/mqtt/data/506_data'
```



Data & Telemetry WIKI 2026

Shell
Eco-marathon Data & Telemetry Portal

Search

You are here: Home » **Student Documentation**

student_documentation:home

Home

Student Documentation

Track Coordinates

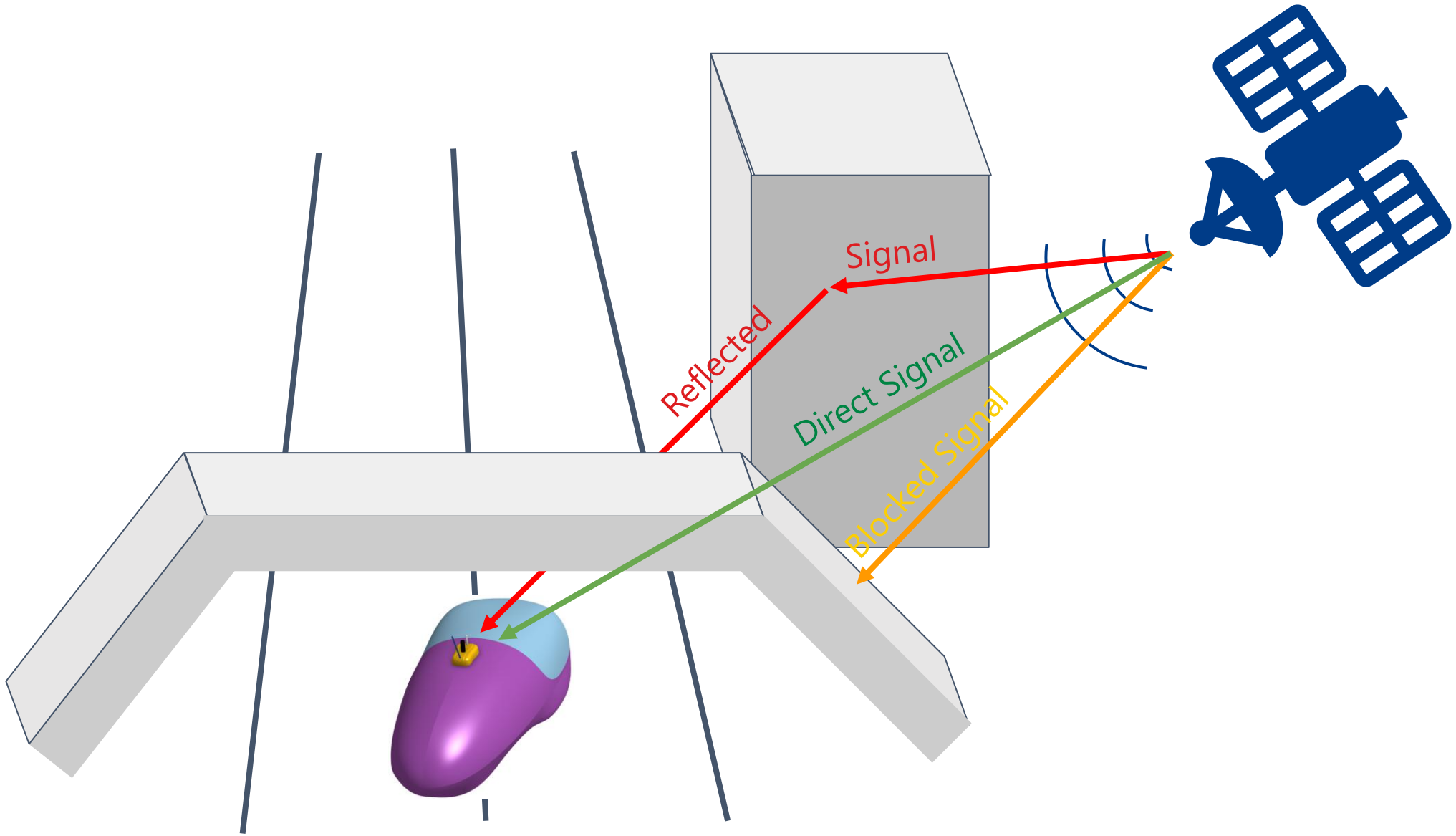
Contact

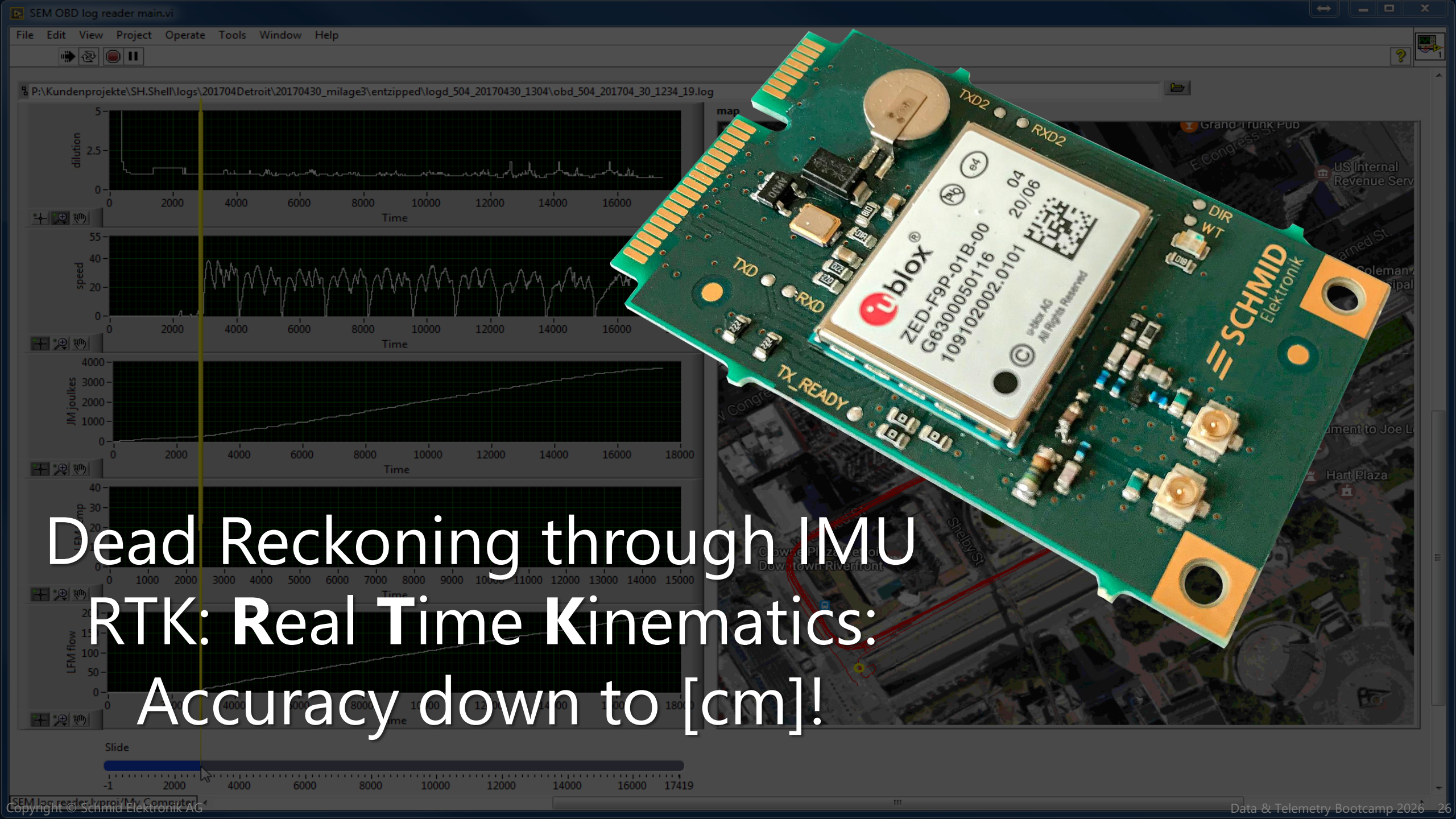
Student Documentation

Select your category/class from the following:

		Prototypes P	Urban Concepts UC
	ICE	Prototype Internal Combustion Engine	Urban Concept Internal Combustion Engine
Hybrid	ICE	Prototype Internal Combustion Hybrid Engine	Urban Concept Internal Combustion Hybrid Engine
	BE	Prototype Battery Electric	Urban Concept Battery Electric
	H2	Prototype Hydrogen without Supercapacitor	Urban Concept Hydrogen without Supercapacitor
Supercap	H2	Prototype Hydrogen with Supercapacitor	Urban Concept Hydrogen with Supercapacitor

GPS Data and it’s Challenge



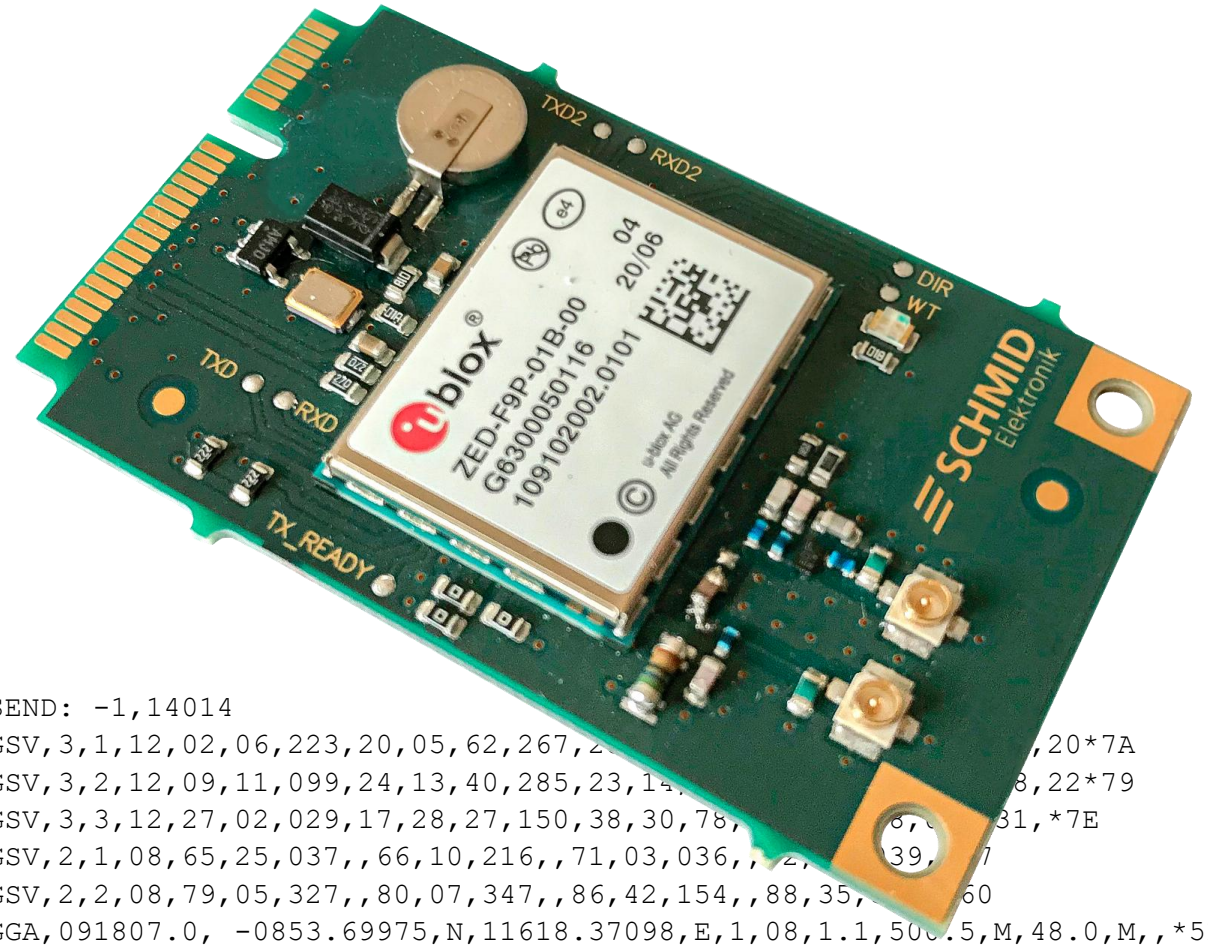


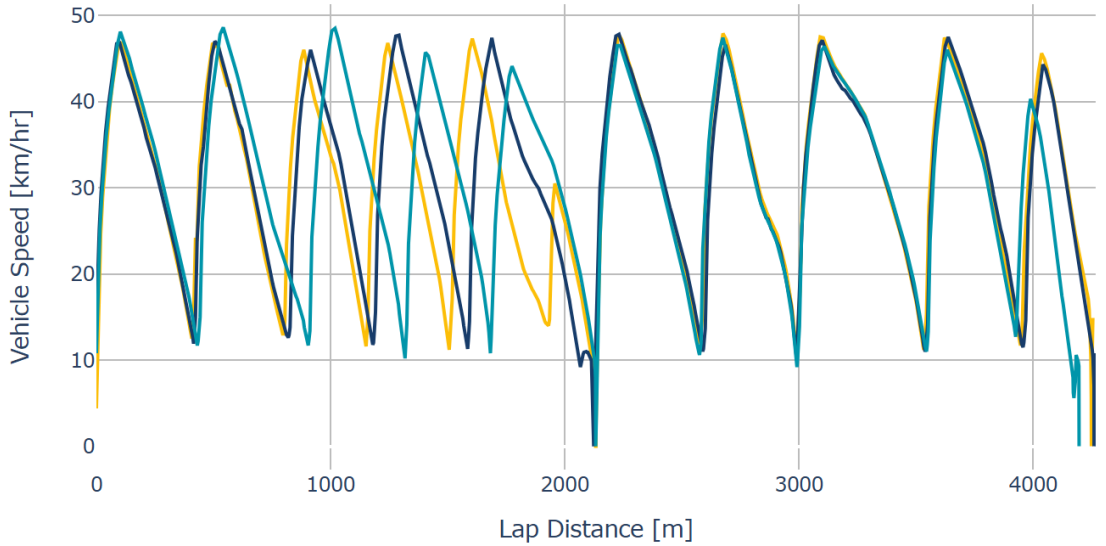
Passive Helix-Antenna and GPS RTK

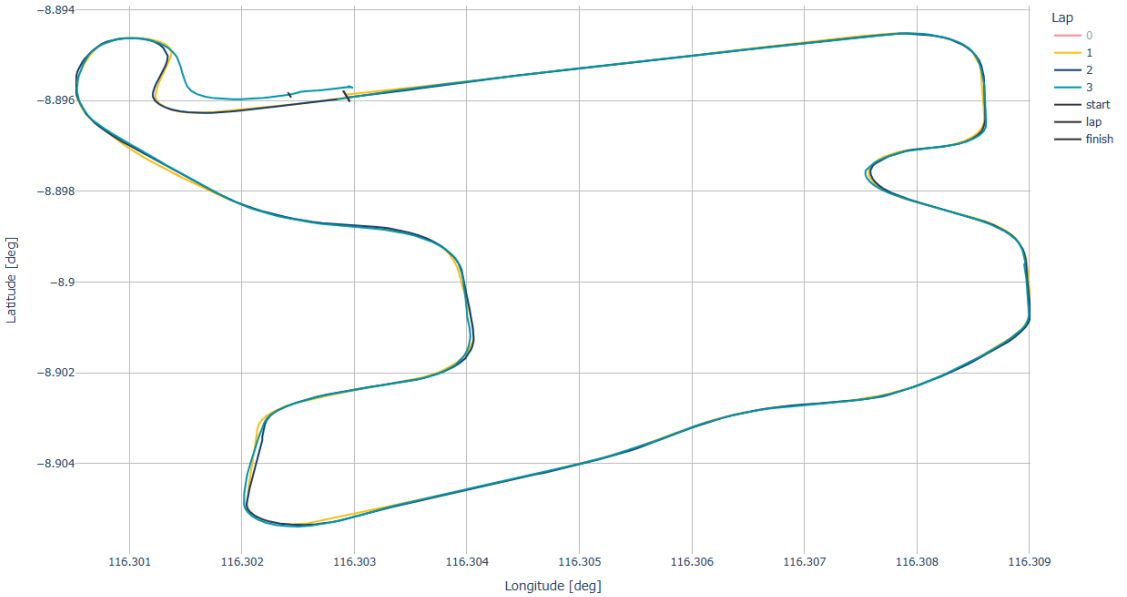
Maxtena

M1621HCT-P-SMA

GPS-Sensor for **Position** and **Speed**

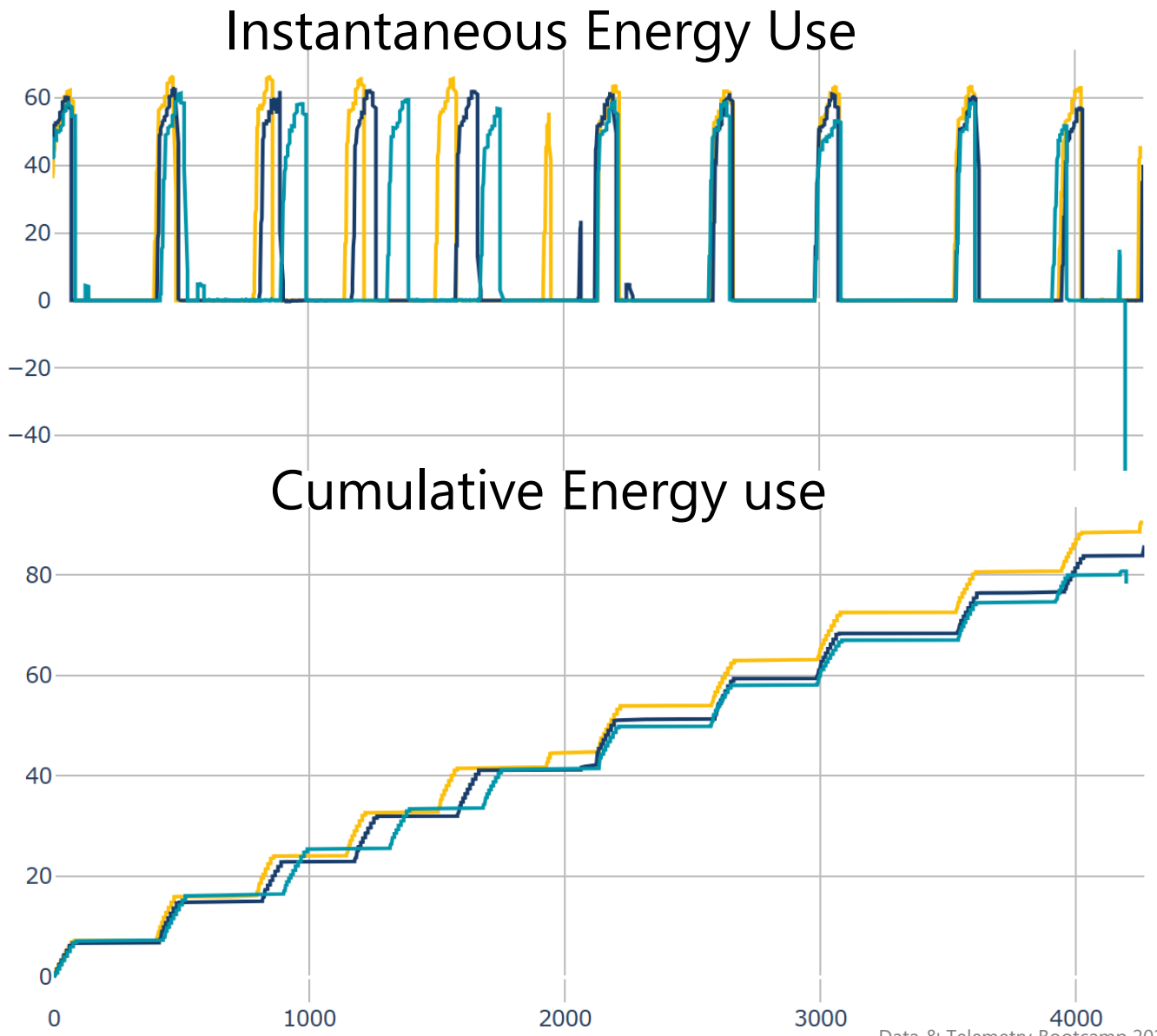




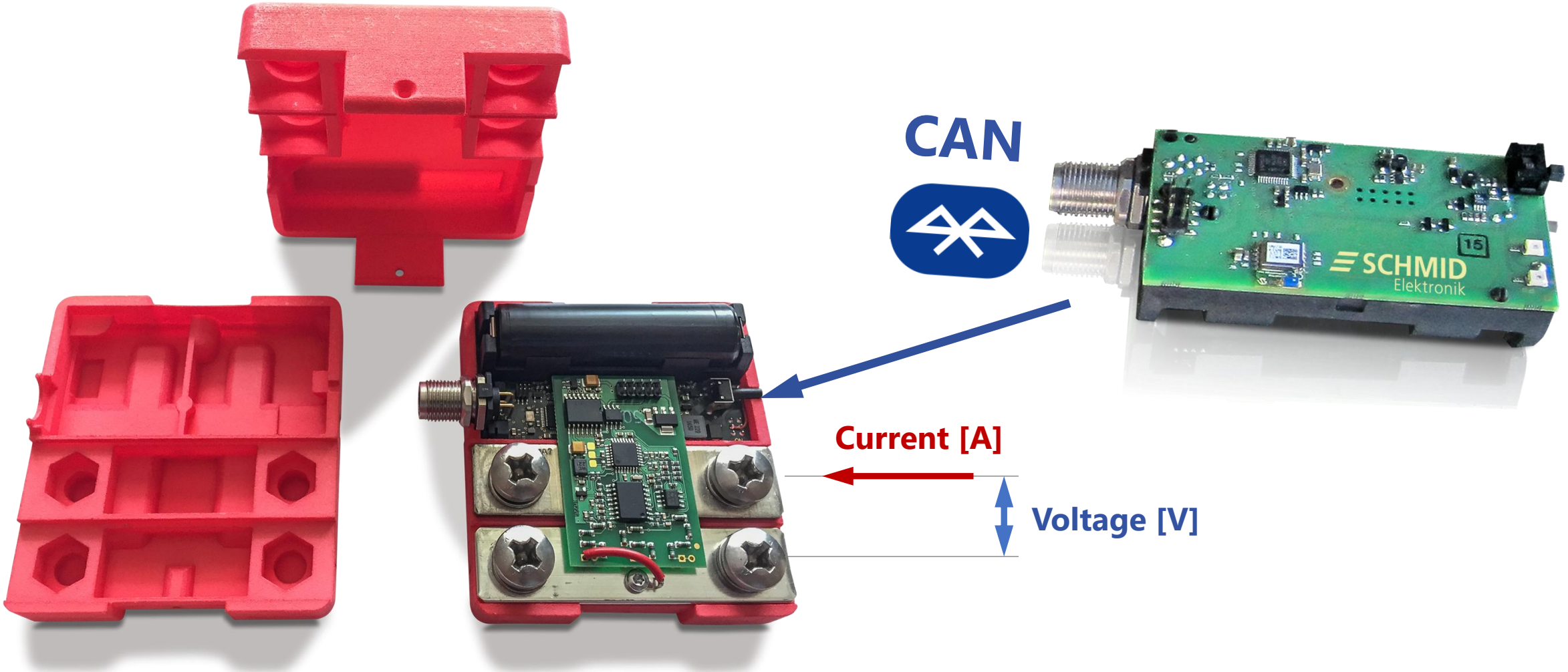



```
^POSEND: -1,14014
$GPGSV,3,1,12,02,06,223,20,05,62,267,2                ,20*7A
$GPGSV,3,2,12,09,11,099,24,13,40,285,23,14            8,22*79
$GPGSV,3,3,12,27,02,029,17,28,27,150,38,30,78,        31,*7E
$GLGSV,2,1,08,65,25,037,,66,10,216,,71,03,036,,       39,*7
$GLGSV,2,2,08,79,05,327,,80,07,347,,86,42,154,,88,35,       60
$GPGGA,091807.0, -0853.69975,N,11618.37098,E,1,08,1.1,500.5,M,48.0,M,,*5D
$GNGNS,091807.0,-0853.69975,N,11618.37098,E,AN,08,1.1,500.5,48.0,,*66
$GPVTG,144.6,T,144.8,M,0.0,N,0.0,K,A*2D
$GPRMC,091807.0,A, -0853.69975,N,11618.37098,E,0.0,144.6,110601,0.2,W,A*10
$GPGSA,A,2,02,05,07,09,13,14,28,30,,,,,1.4,1.1,0.8*39
$GNGSA,A,2,02,05,07,09,13,14,28,30,,,,,1.4,1.1,0.8*27
$GNGSA,A,2,,,,,,,,,,,,,1.4,1.1,0.8*20
$GPHWBIAS,13*3F
```

Joulemeter for **Electrical Energy**



Electrical Energy: **Sensorfusion**



Measures electrical Energy in Joules [J]

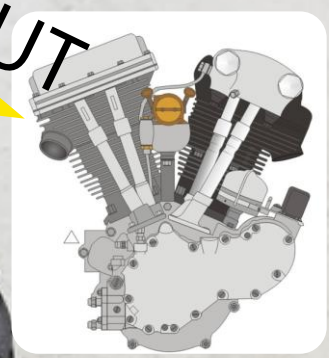
1 Joule [J] = 1 Wattsecond [Ws] = 1 **VA**s = 1 N m = 1 kg m^2 s^{-2}

NEW 2026

Liquids
in milli-liters [ml]
e.g 12.5ml

IN

OUT



Gasflow Meter for **Hydrogen**

Hydrogen in

normalized Liters [l], e.g 0.534l

CAN

GPS Receiver

4G/5G Modem

WIFI

Digital I/O

9x IMU

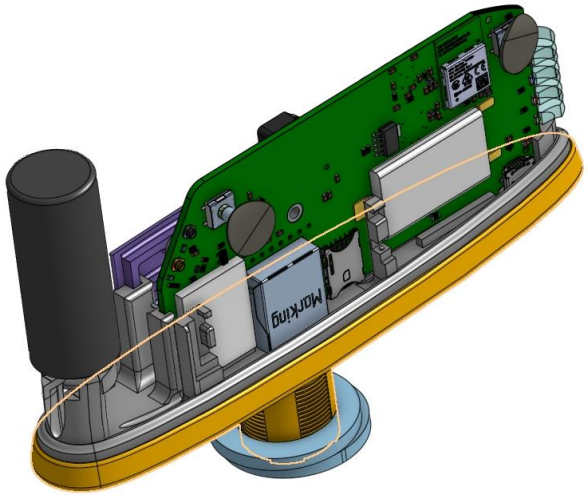
Battery Mgmt

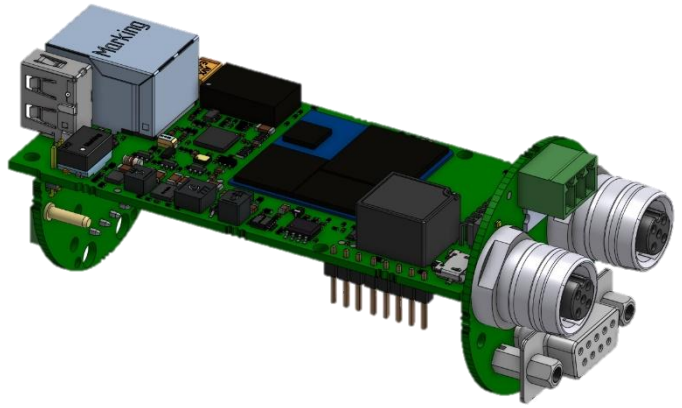
SIM Card

µSD

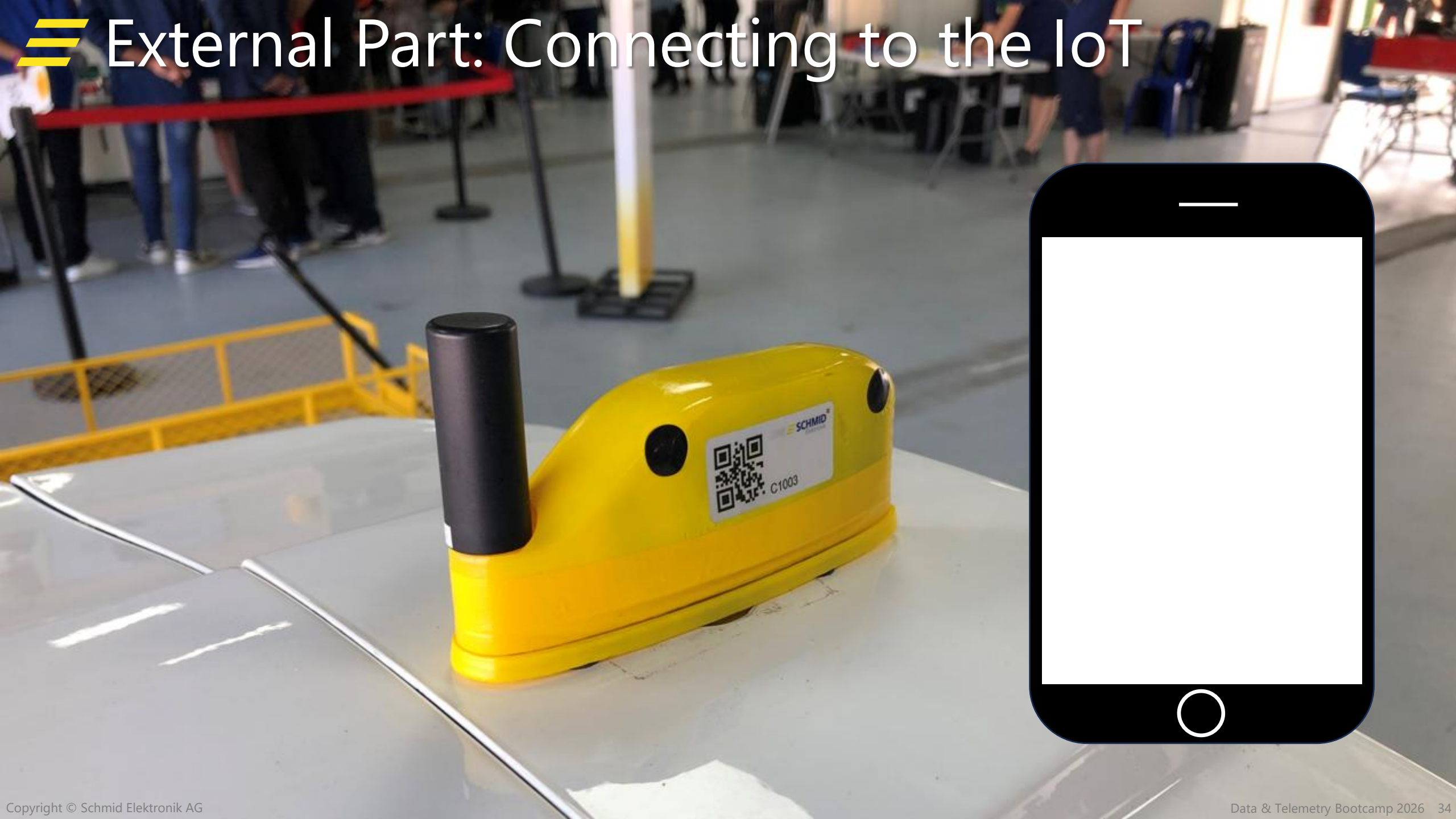
Blue Tooth

1x ARM **Cortex-A7** Cores and 1x **Cortex-M4** Core

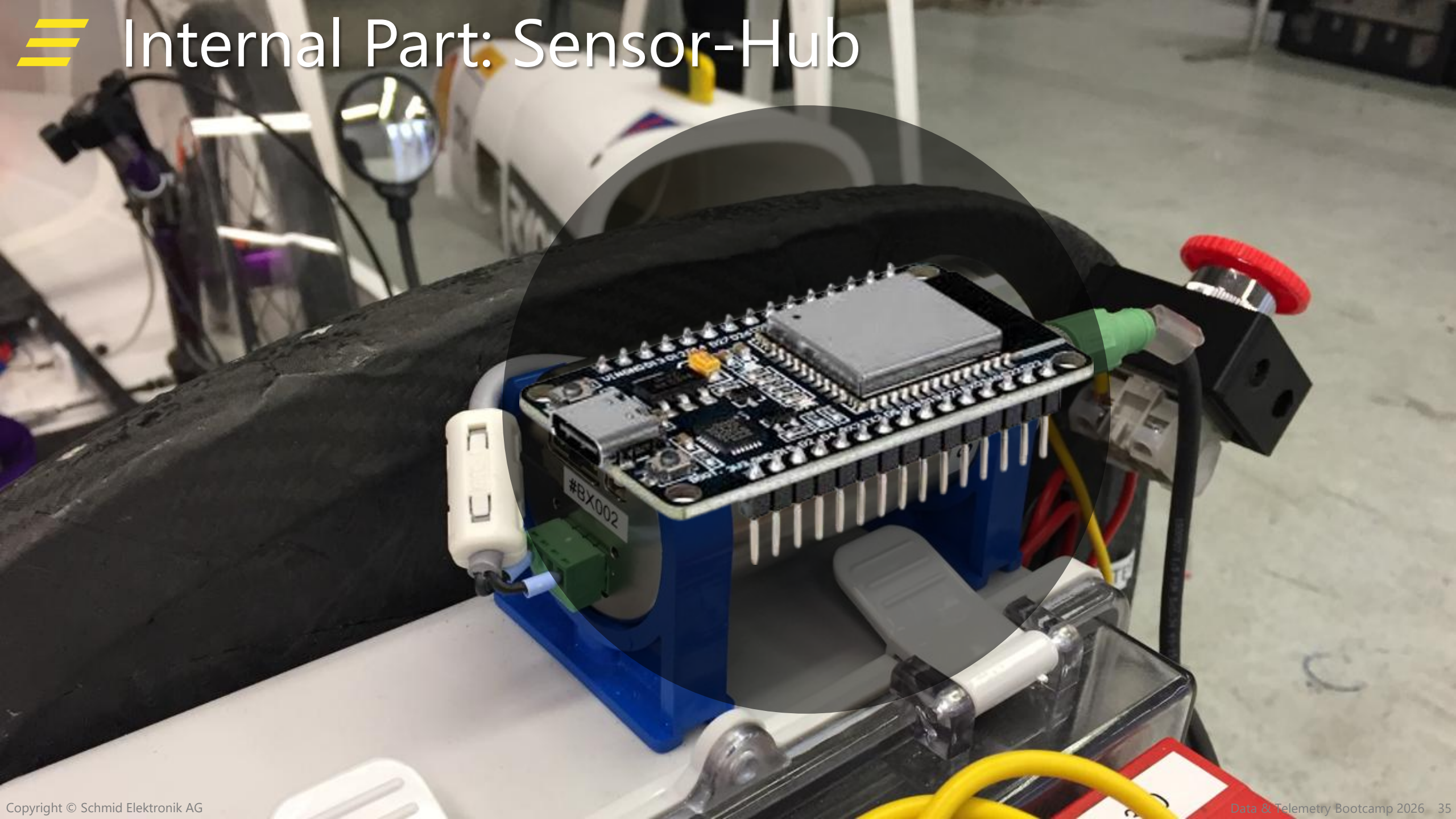




External Part: Connecting to the IoT



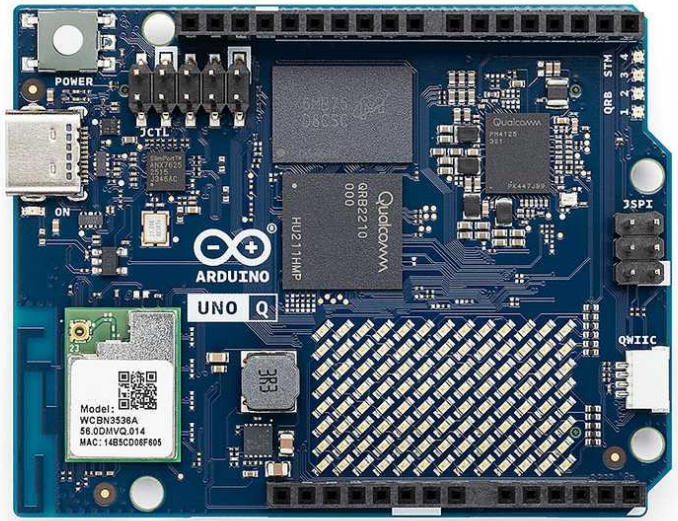
Internal Part: Sensor-Hub



Alternatives that I have seen in OTA-Papers

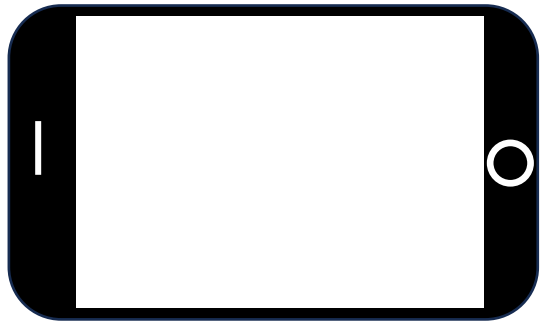
ESP32

Arduino

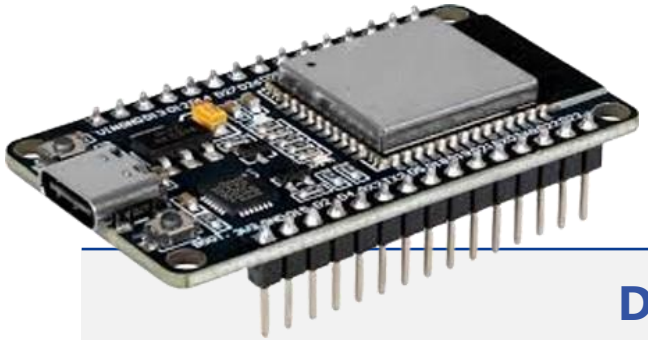


Raspi

Your Phone



Empowering you to create **your own Telemetry**



Data-API (Application Programming Interface)

CAN

CAN

CAN

Joulemeter

Liquid Flowmeter

Gas Flowmeter

Live Dataflow in the IoT-Server

Grafana

Node-RED

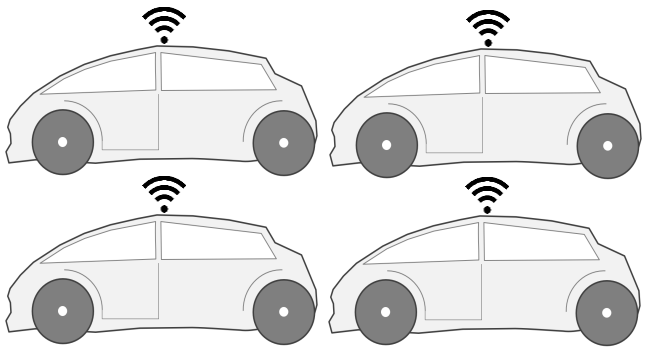
MQTT Broker

Kinetic Service

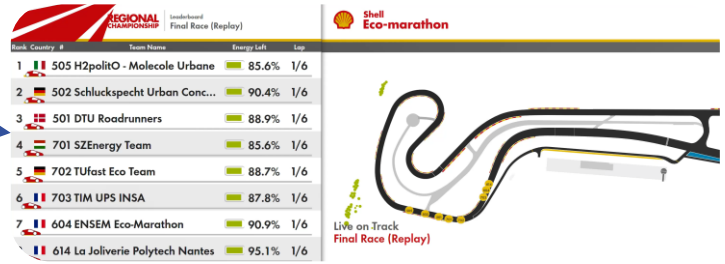
Data Validation
Boundary Check
Outlier Detection
Smooth GPS-Data

Database for Static Data (SD-Card)

Telemetry Monitoring



Onboard Computers use **Publish/Subscribe** Messaging



Live Data Visualization

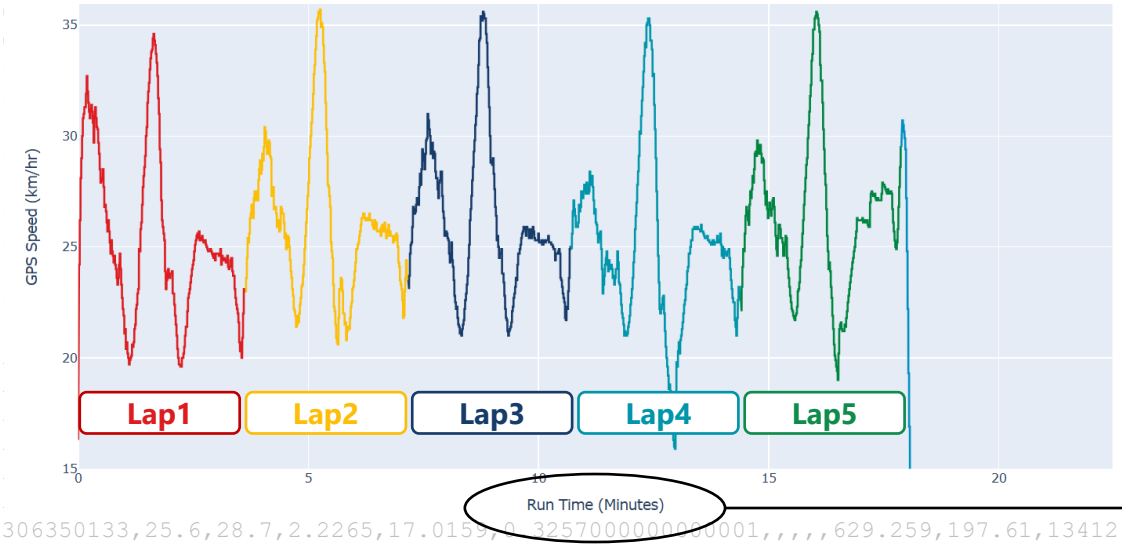
Knowledge

gained from Race Data

Level 3

Time-Based «Chunk» to distance based Lap-to-Lap Data

NEW 2026

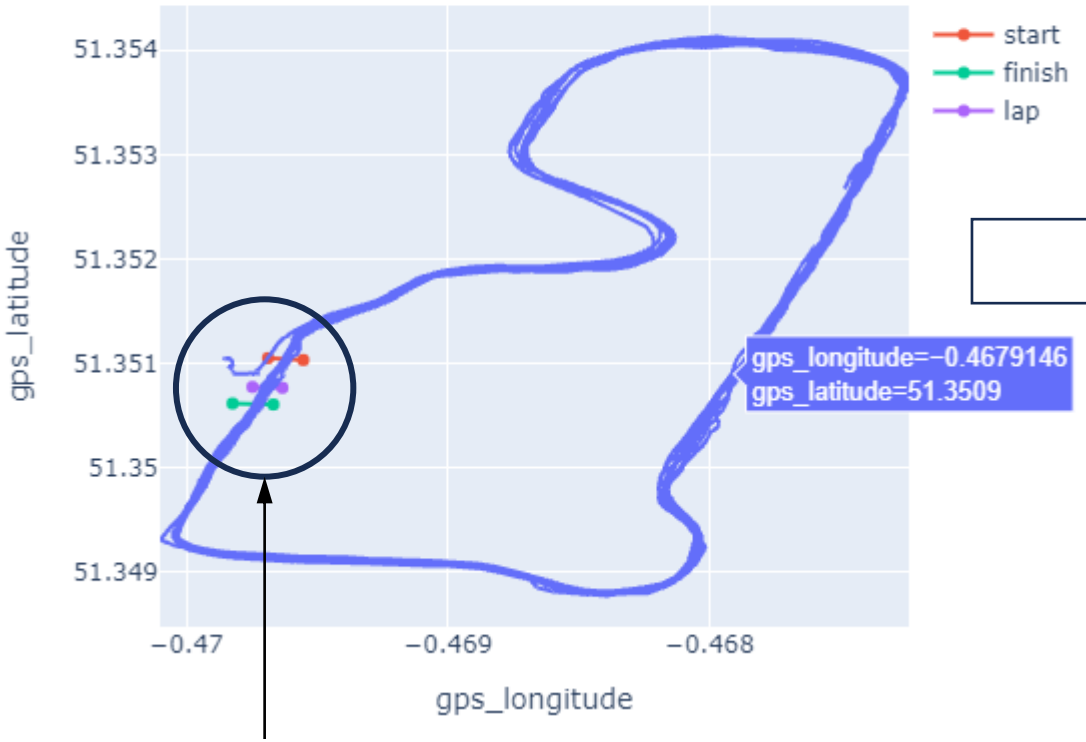


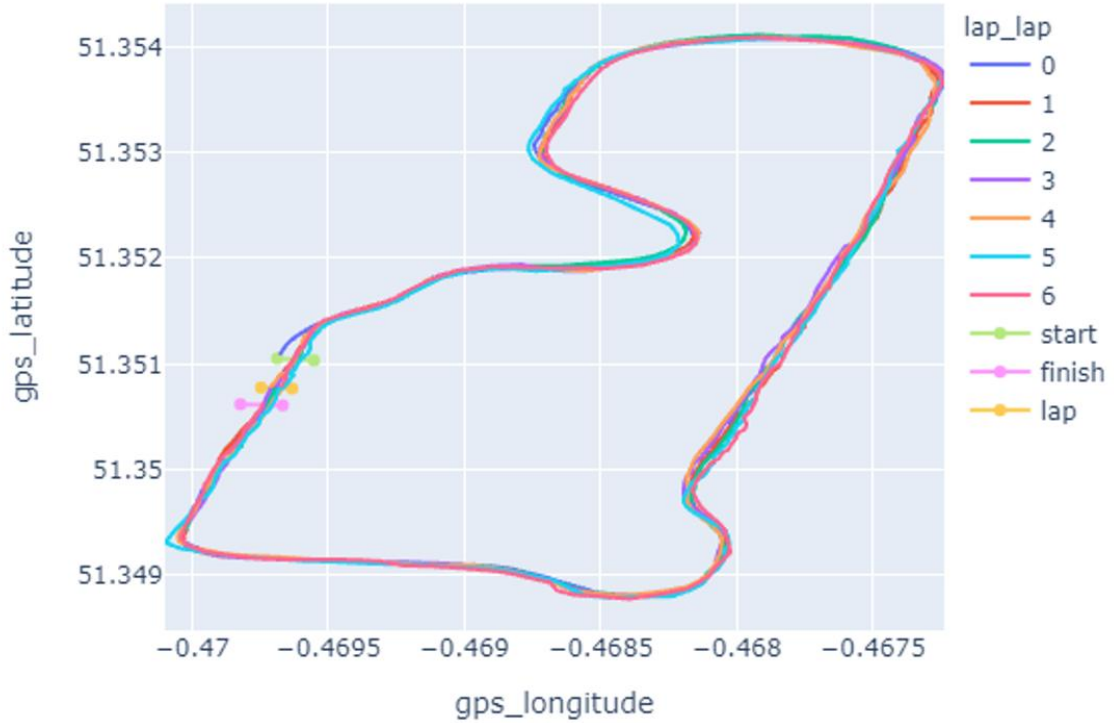
t=0 min

t=17 min

Time-Based «Chunk» to distance based Lap-to-Lap Data

```
fig = df.plot(x='gps_longitude', y='gps_latitude')
for name, points in lines.items():
    fig.add_trace(go.Scatter(x=points[0], y=points[1], name=name))
fig.show()
```






```
lines = {
    'start': ((-0.469553367, -0.4696881), (51.35103562, 51.3510535)),
    'finish': ((-0.469667683, -0.469823533), (51.35060957, 51.35061913)),
    'lap': ((-0.469633117, -0.46974765), (51.35076913, 51.35077756))
}
```

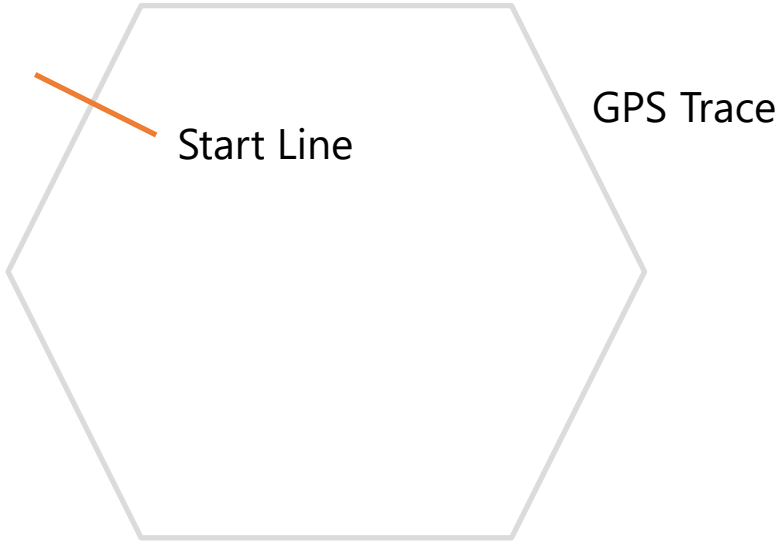
Time-Based «Chunk» to distance based Lap-to-Lap Data

```python
for line, points in lines.items():
    x1 = points[0][0]
    y1 = points[1][0]
    x2 = points[0][1]
    y2 = points[1][1]
    x3 = df.gps_longitude
    y3 = df.gps_latitude
    x4 = df.gps_longitude.shift(-1)
    y4 = df.gps_latitude.shift(-1)

    denom = ((x1 - x2) * (y3 - y4)) - ((y1 - y2) * (x3 - x4))
    t = round((((x1 - x3) * (y3 - y4)) - ((y1 - y3) * (x3 - x4))) / denom, 1)
    u = round((((x1 - x2) * (y1 - y3)) - ((y1 - y2) * (x1 - x3))) / denom, 1)

    df[f'{line}_cross'] = (t >= 0) & (t <= 1) & (u >= 0) & (u <= 1)
    df[f'{line}_lap'] = df[f'{line}_cross'].cumsum()
```

- Efficiently calculate when a vehicle has crossed one of the three lines entered previously: Line-segment intersection




```python
df.drop(index=df.loc[df.start_lap < 1].index, inplace=True)
df.drop(
    index=df.loc[df.finish_lap >= df.finish_lap.max()].index,
    inplace=True
)
```

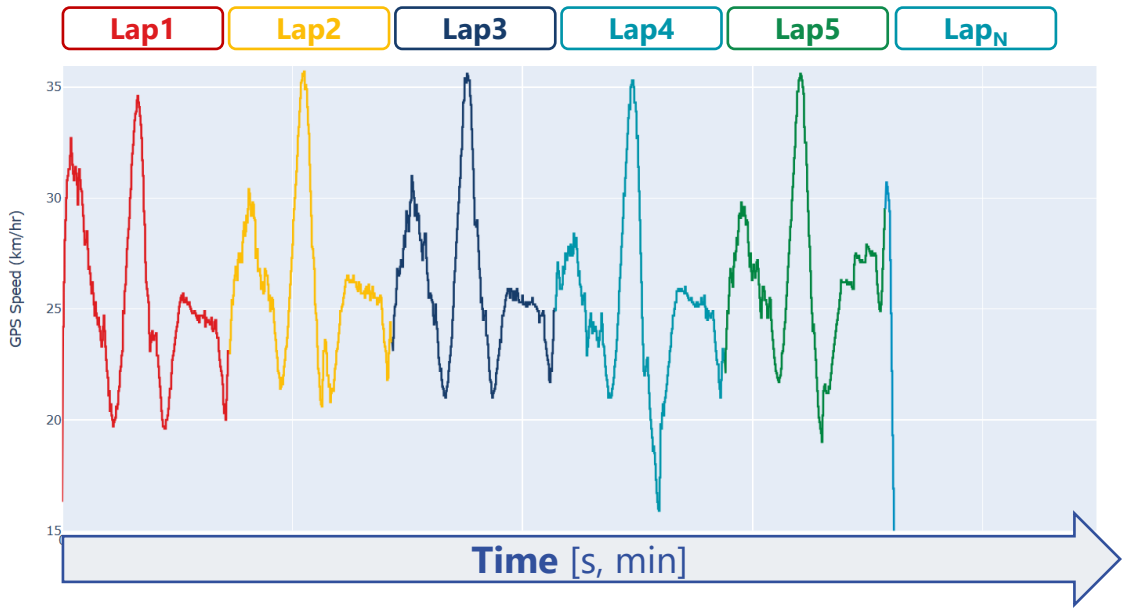
- Data at beginning and end of logfile is dropped

Formula: WIKIPEDIA „**Line-Line-Intersection**“

$$t = \frac{\begin{vmatrix} x_1 - x_3 & x_3 - x_4 \\ y_1 - y_3 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

$$u = \frac{\begin{vmatrix} x_1 - x_2 & x_1 - x_3 \\ y_1 - y_2 & y_1 - y_3 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = -\frac{(x_1 - x_2)(y_1 - y_3) - (y_1 - y_2)(x_1 - x_3)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)},$$

Time-Based «Chunk» to distance based Lap-to-Lap Data



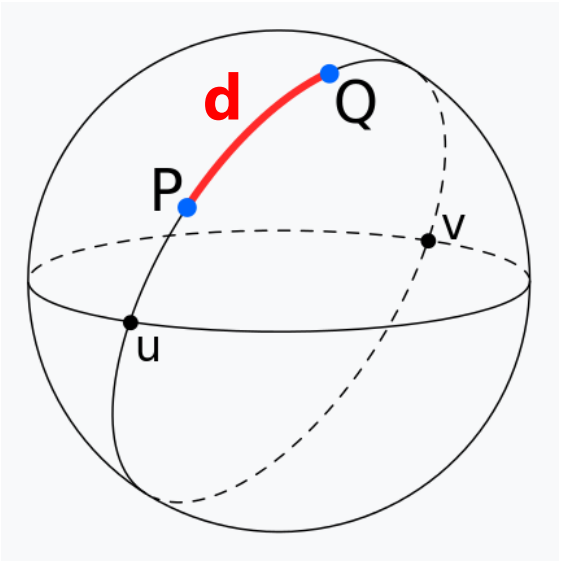
Time-Based «Chunk» to distance based Lap-to-Lap Data

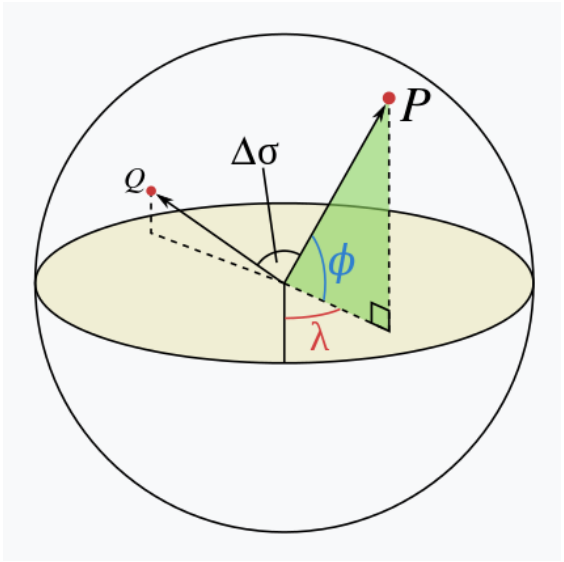
The Haversine Formula:

en.wikipedia.org/wiki/Haversine_formula

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

φ_1, φ_2 are the latitude of point 1 and latitude of point 2,
λ_1, λ_2 are the longitude of point 1 and longitude of point 2.



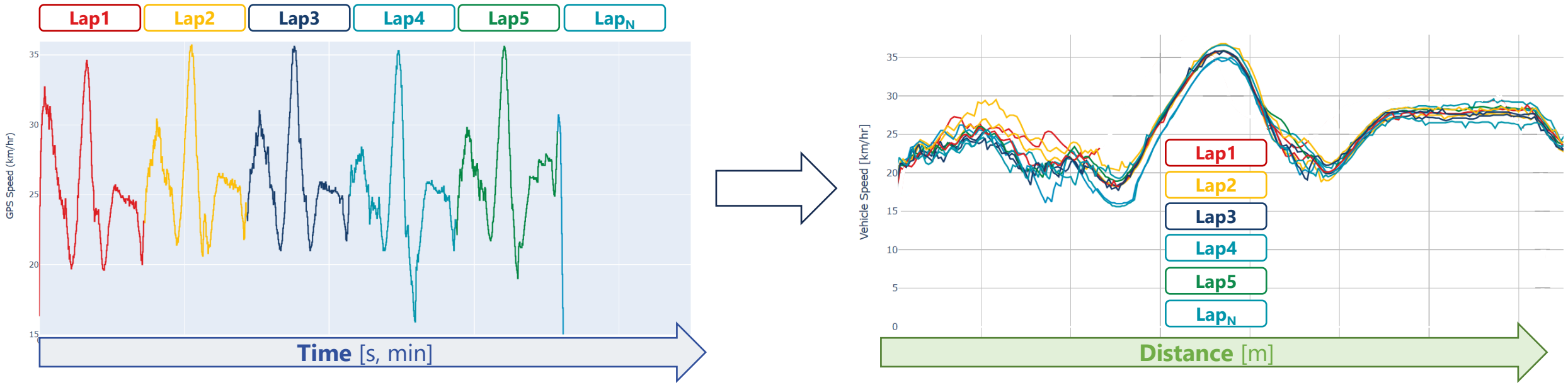



```python
lon1 = df.gps_longitude / 180 * np.pi
lat1 = df.gps_latitude / 180 * np.pi
lon2 = df.gps_longitude.shift(1) / 180 * np.pi
lat2 = df.gps_latitude.shift(1) / 180 * np.pi

r = 6371000  Earth Radius in [km]

df['dist'] = (
    2
    * r
    * np.arcsin(
        (
            np.sin((lat2 - lat1) / 2) **2
            + np.cos(lat1) * np.cos(lat2) * np.sin((lon2 - lon1) / 2) ** 2
        )
        ** 0.5
    )
)
df['dist'] = df.dist.cumsum()
```

Time-Based «Chunk» to distance based Lap-to-Lap Data



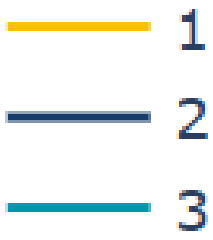
Python Code for this Transformation

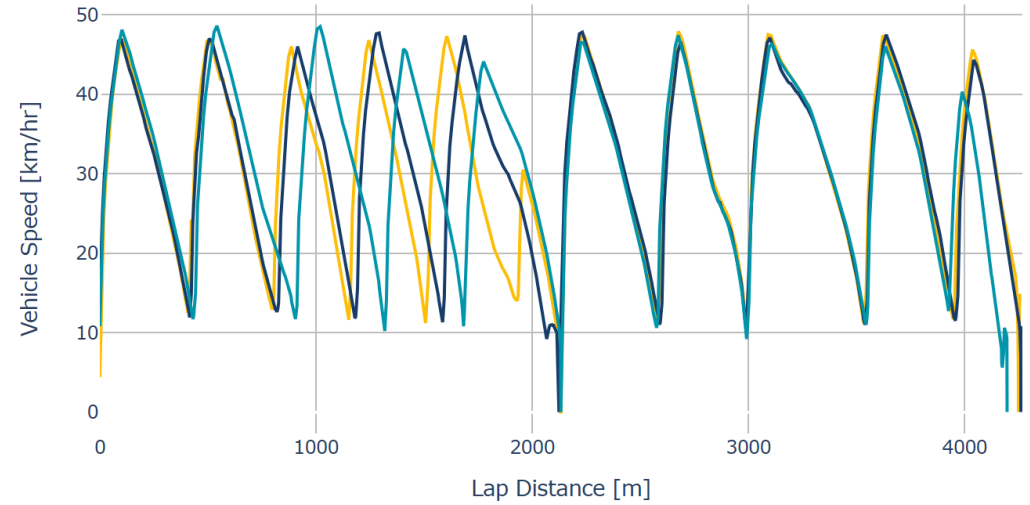
OPTIONAL: DOWNLOAD PYTHON-PACKAGES FOR THE ANACONDA-SUITE

This step is optional. The three sandboxes can be conveniently run directly in your own browser (Binder) via the three links above. If you want to use these two packages (Jupyter Lab), download the **Anaconda 2.7.0** suite here at anaconda.org.

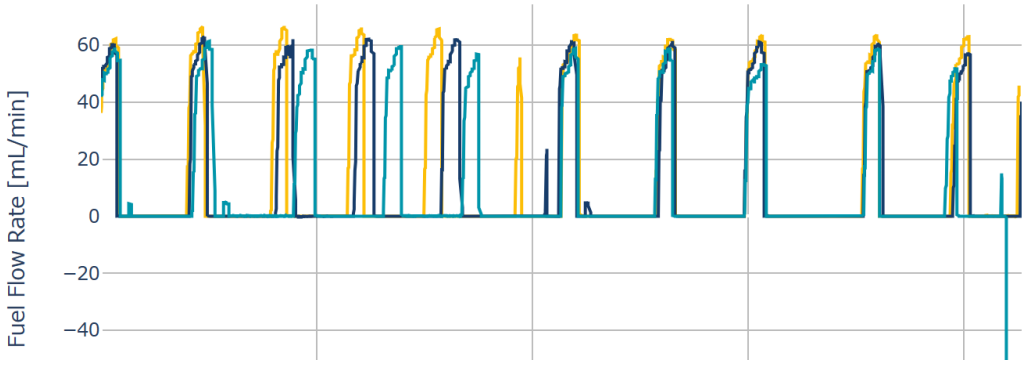
Phoenix Bootcamp Python Code Space-Time Transformation	DOWNLOAD
Phoenix-Bootcamp Sandbox #1 Data Analysis	DOWNLOAD
Phoenix Bootcamp Sandbox #2 and #3 Knowledge Graph and AI	DOWNLOAD

Data Correlation

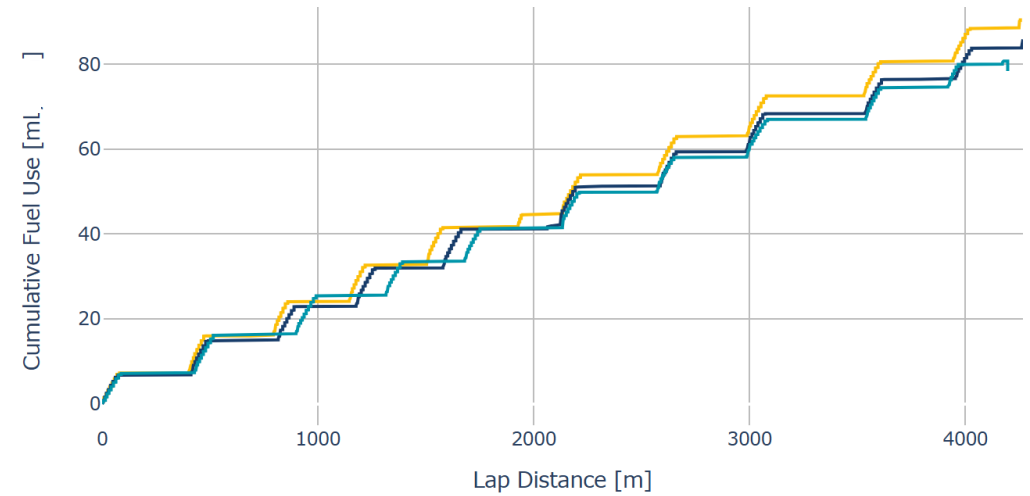




Speed [km/h]



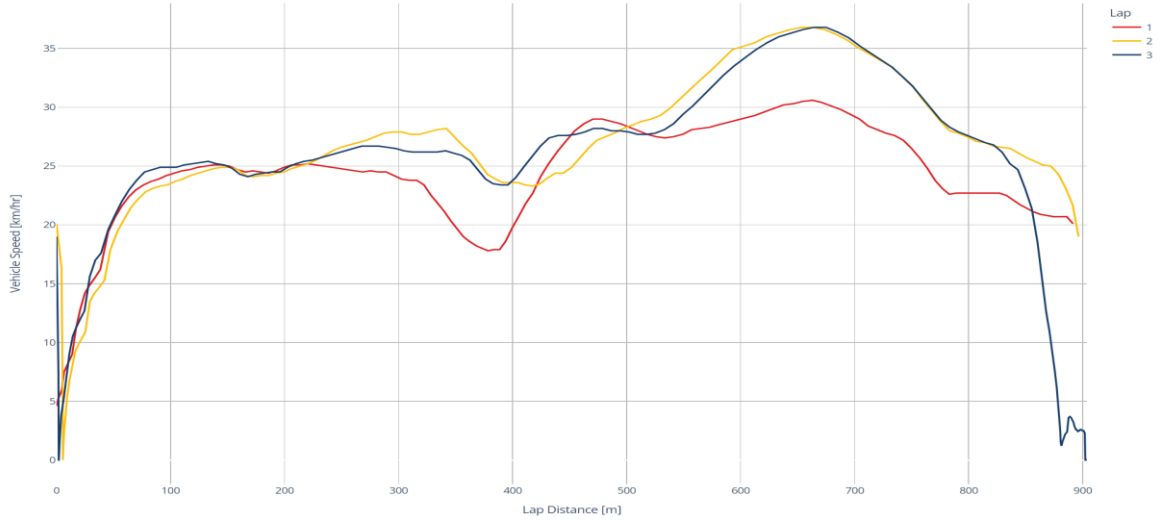
<u>Instantaneous</u> Energy Consumption [ml/min]



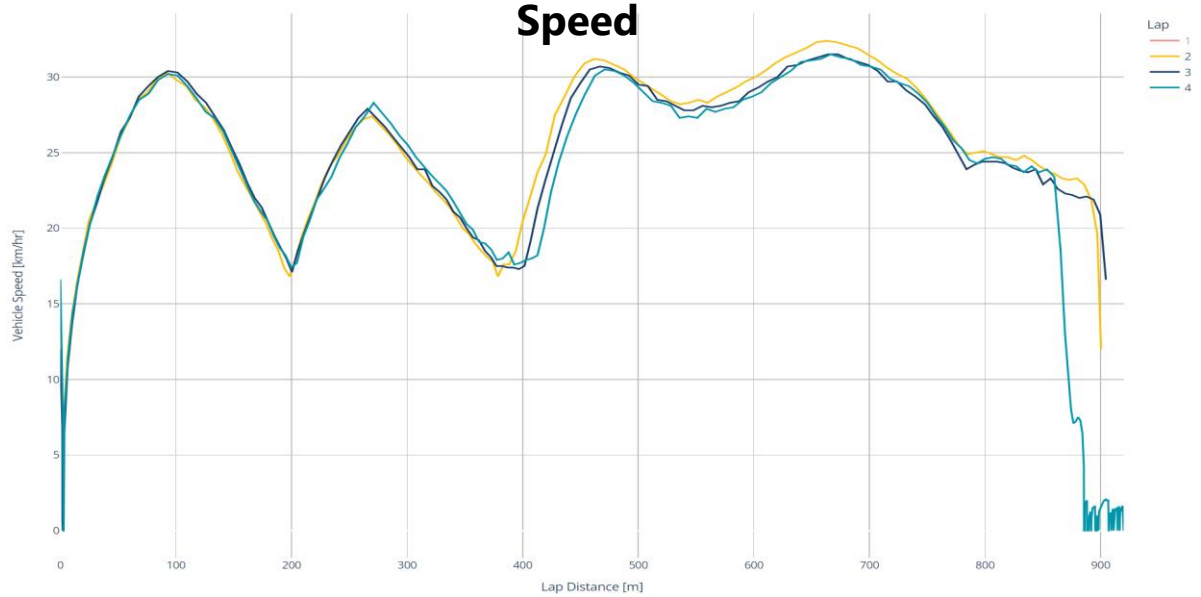
<u>Cumulative</u> Energy Consumption [ml]

Driving Consistency!

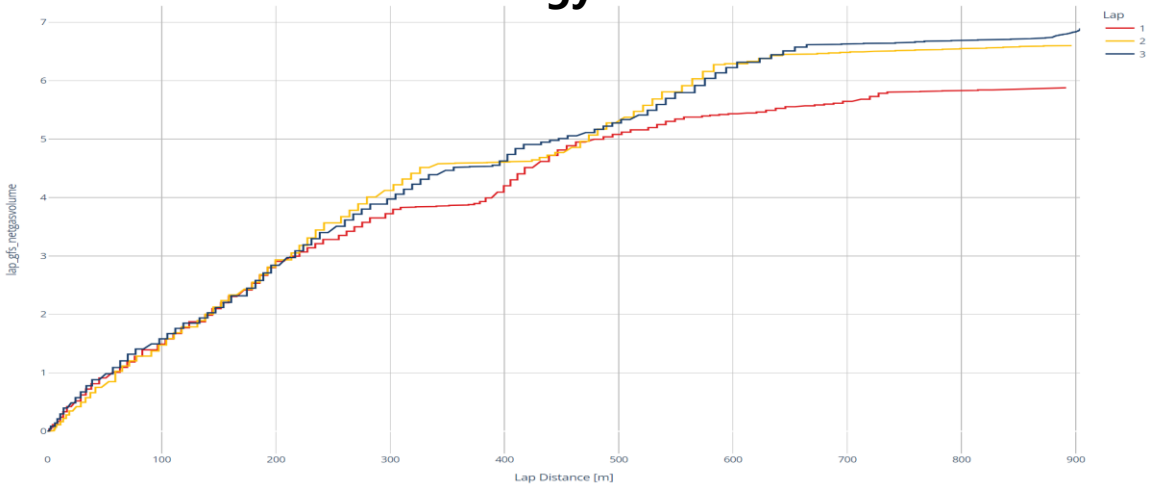
Speed



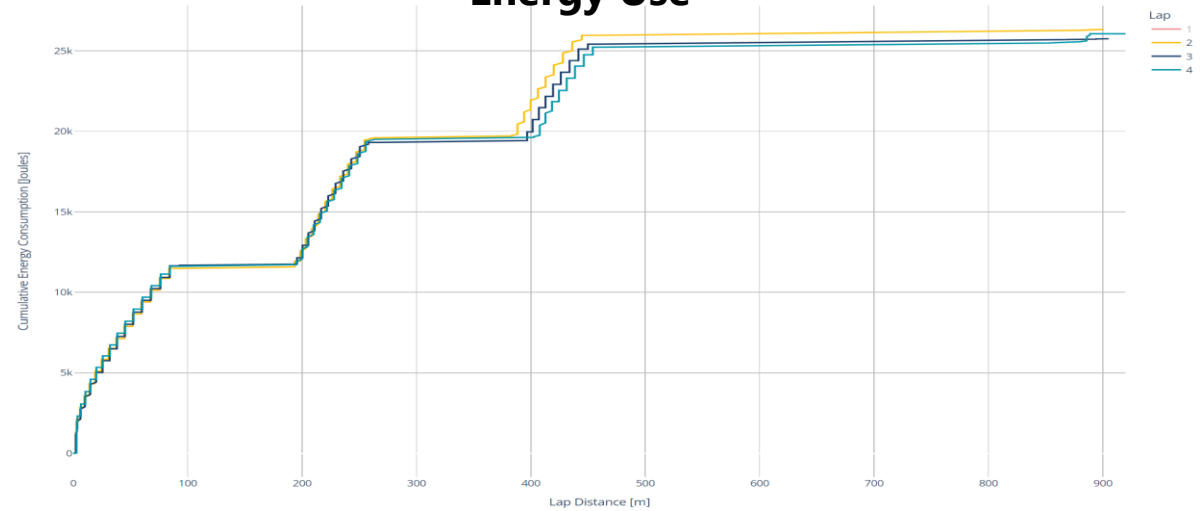
Speed



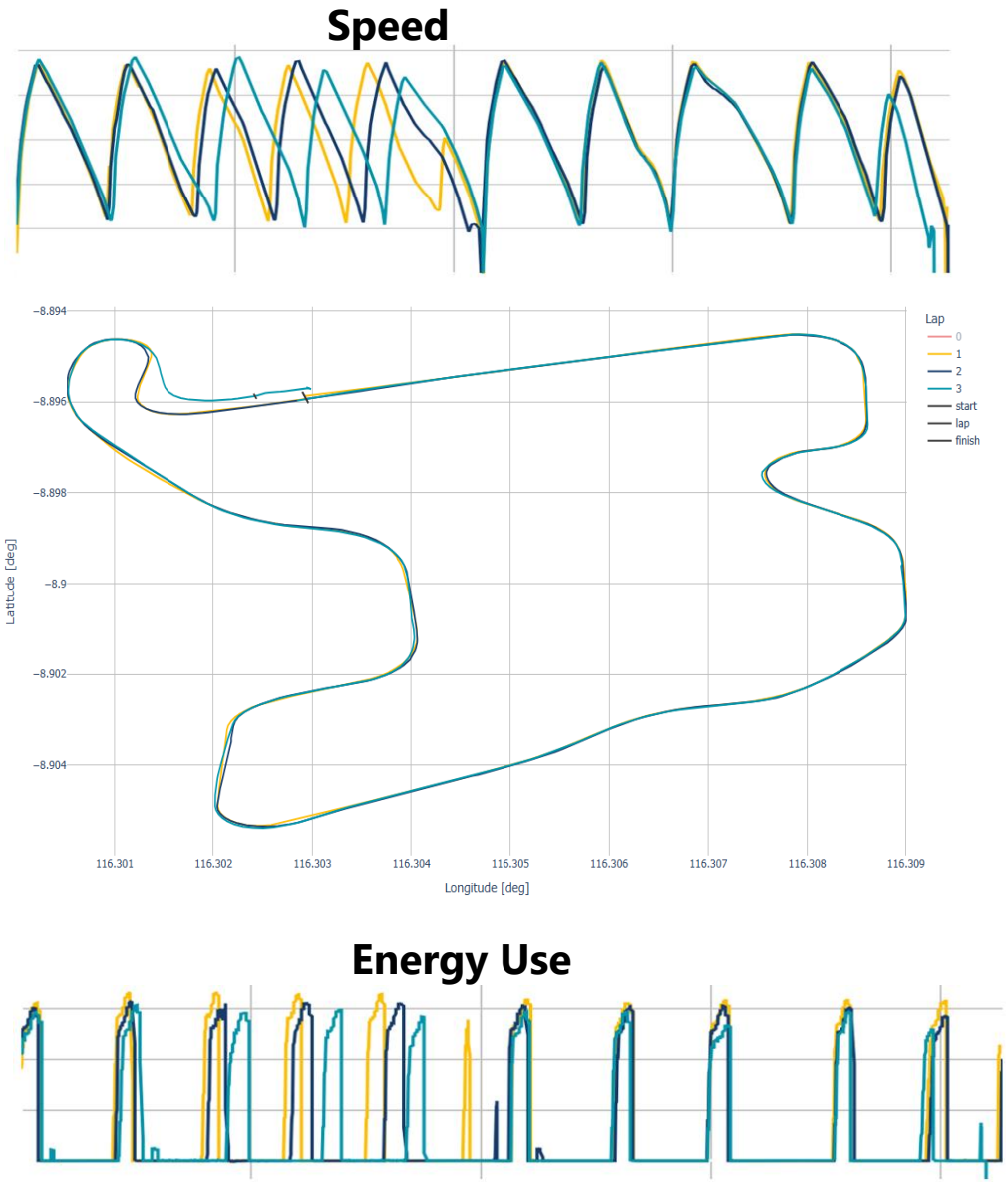
Energy Use



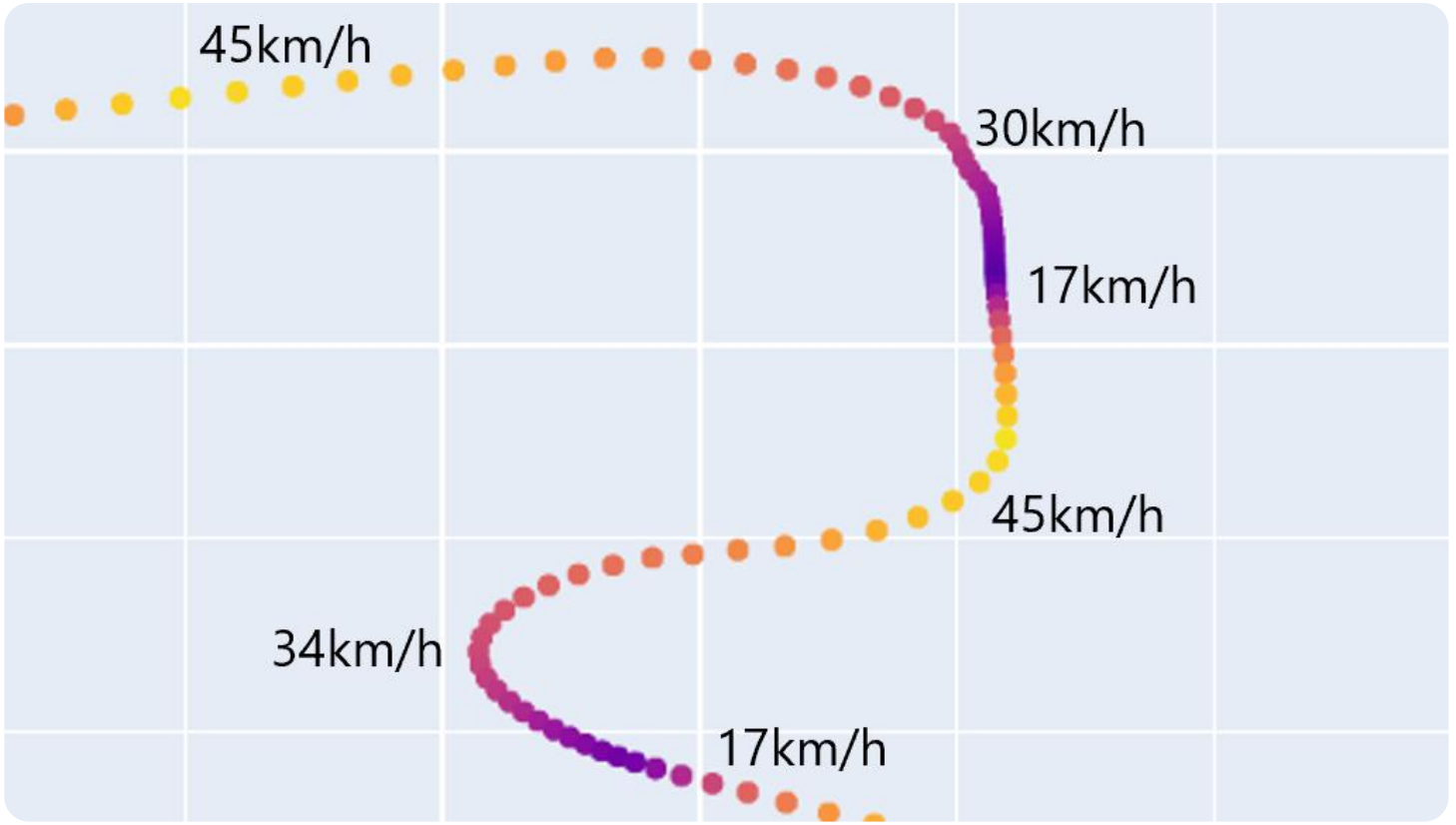
Energy Use



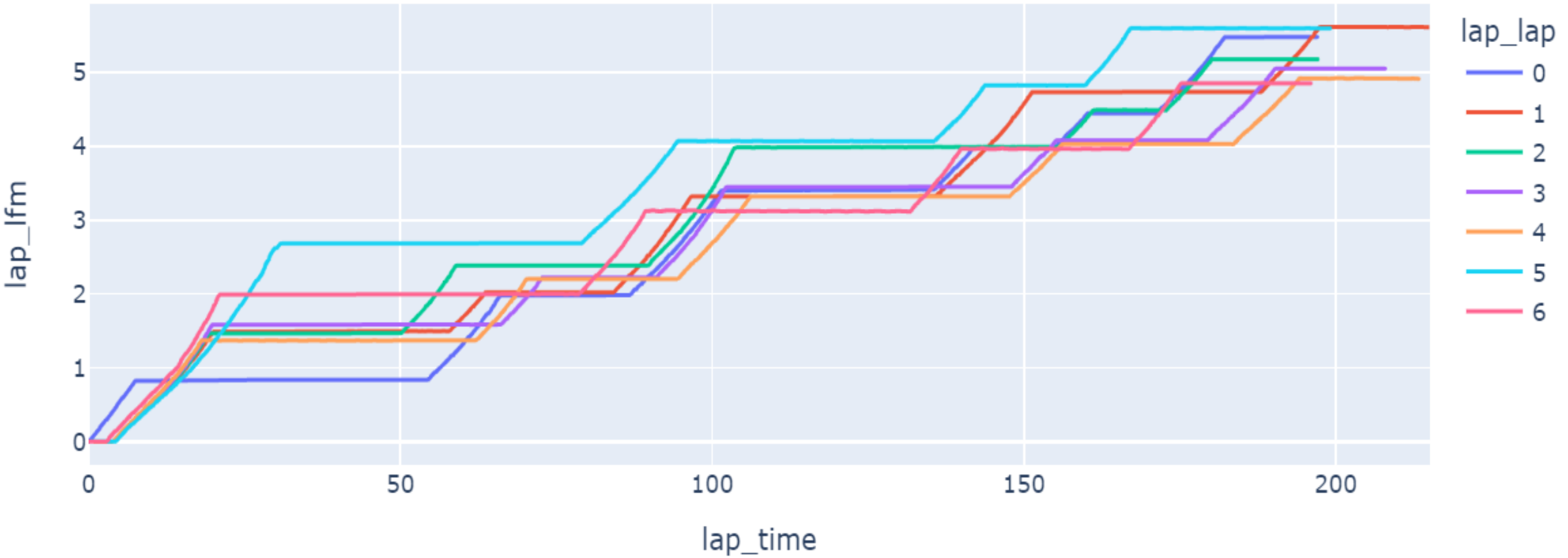
From **Pattern** to **Meaning**



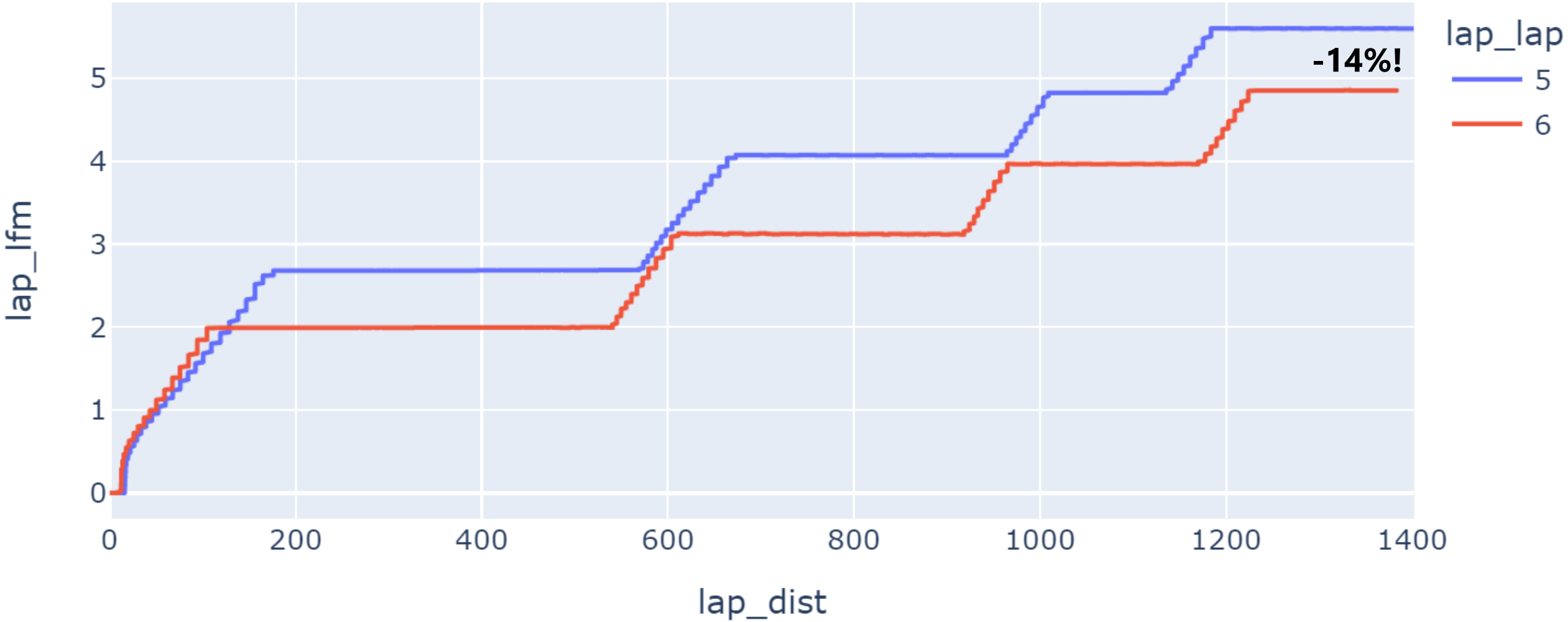
Pattern and Meaning



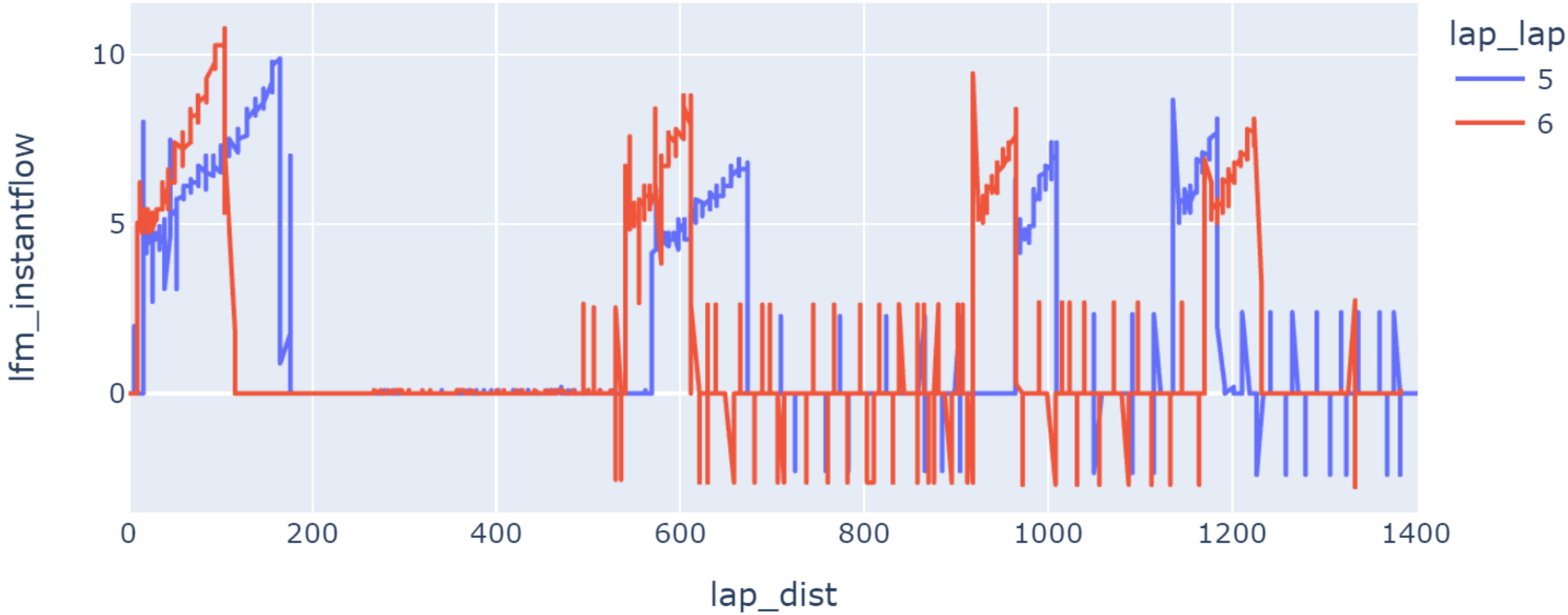
Quantitative Analysis: **Cumulative** Consumption



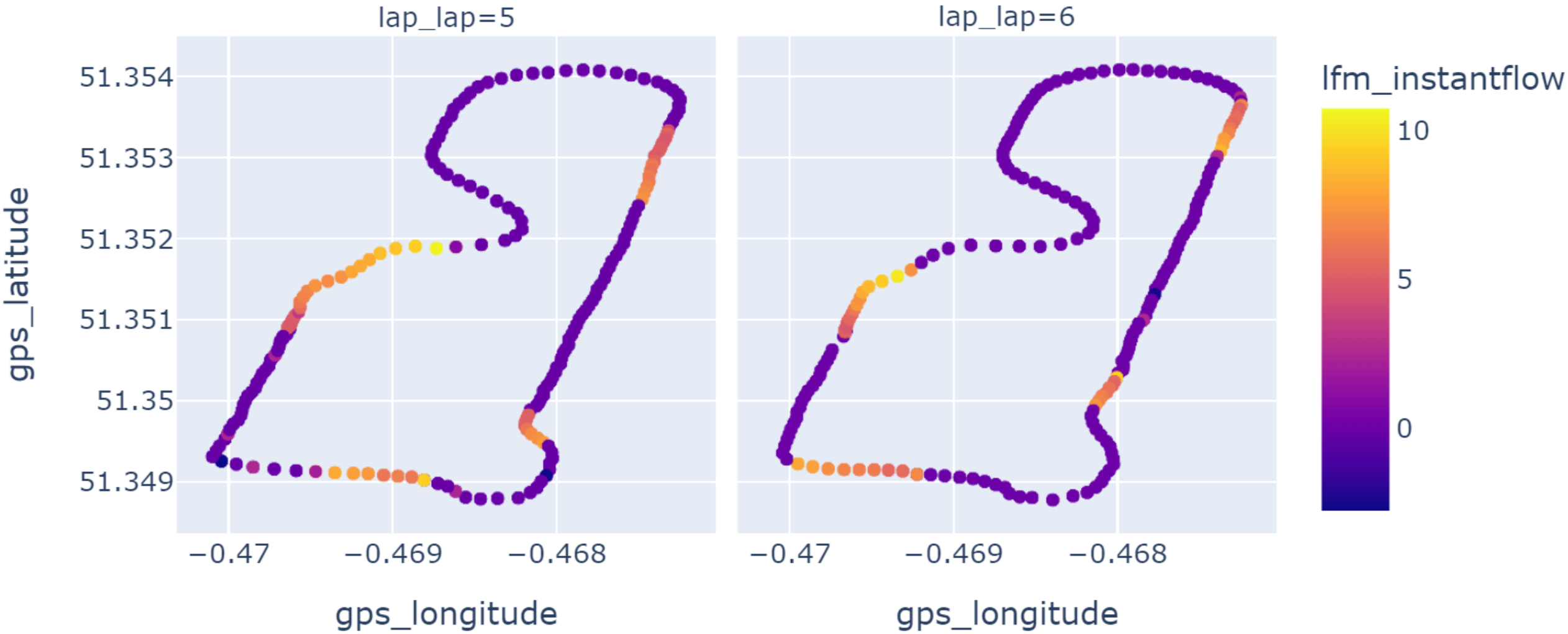
Quantitative Analysis: **Cumulative** Consumption



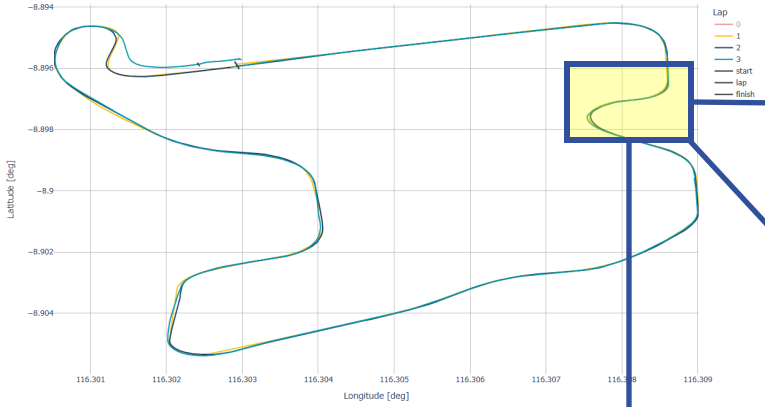
Quantitative Analysis: **Instantaneous** Consumption



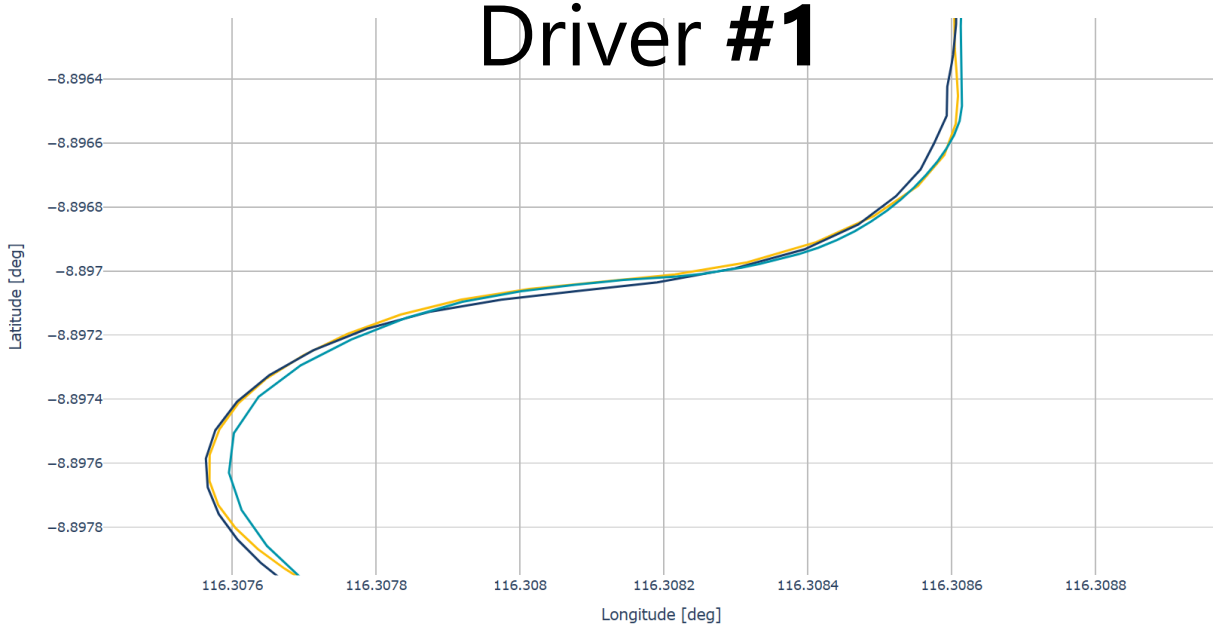
Qualitative Analysis: **Burn & Coast Heatmap**



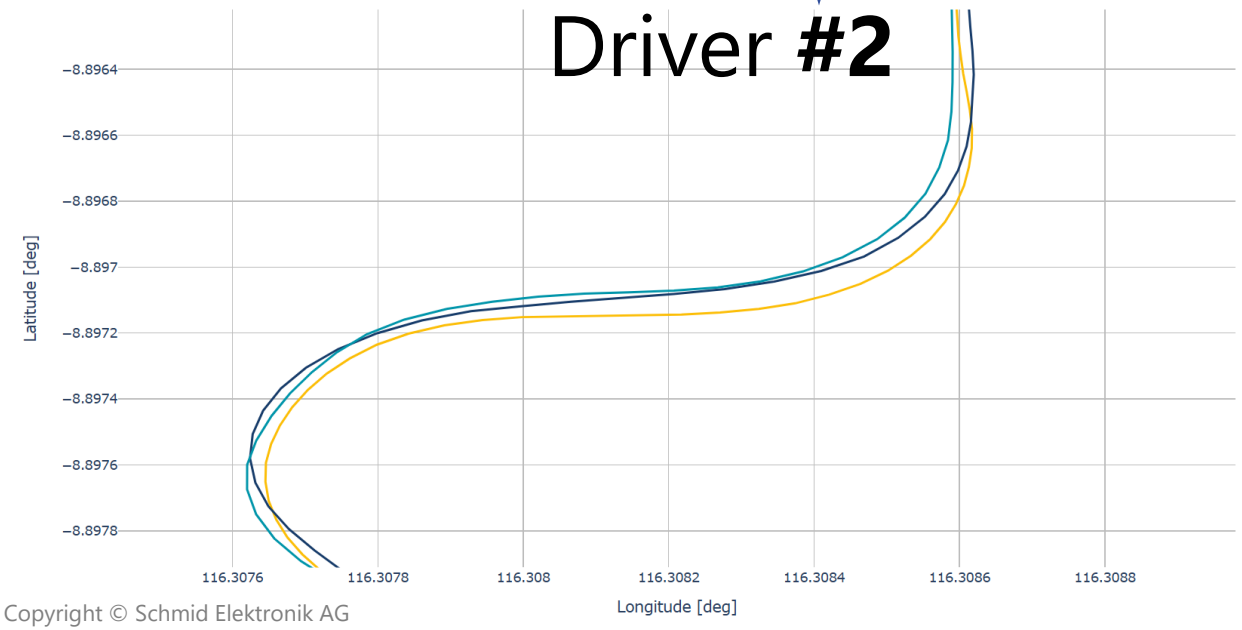
Race Lines in **Corners**



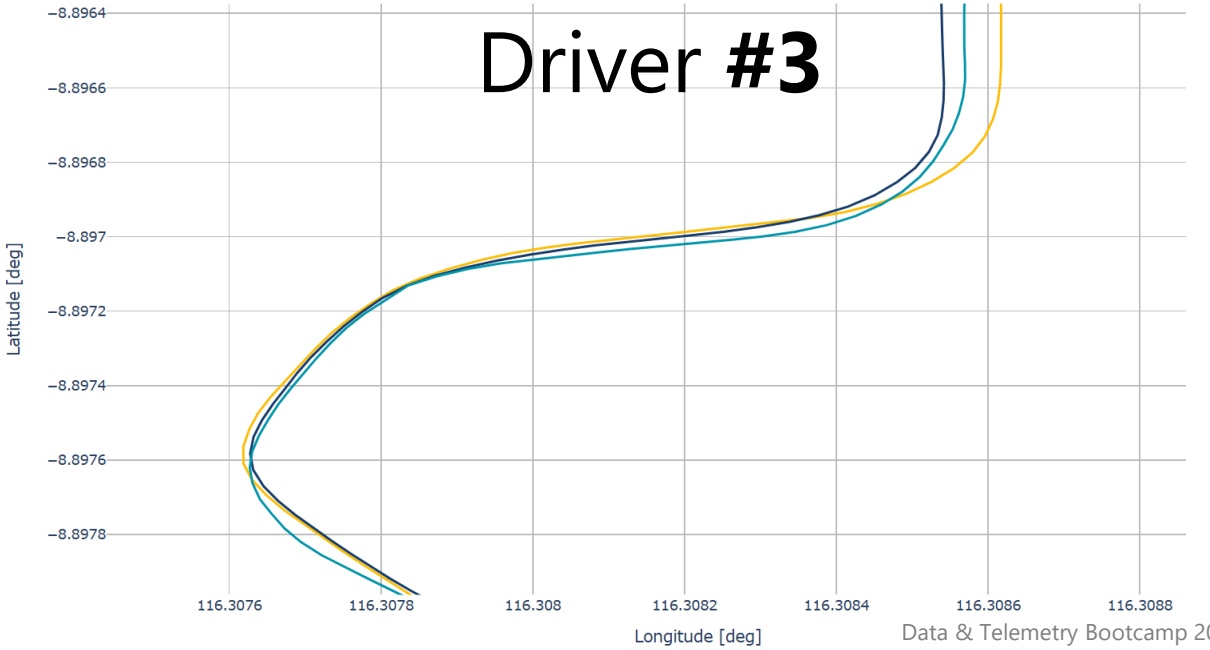
Driver **#1**



Driver **#2**

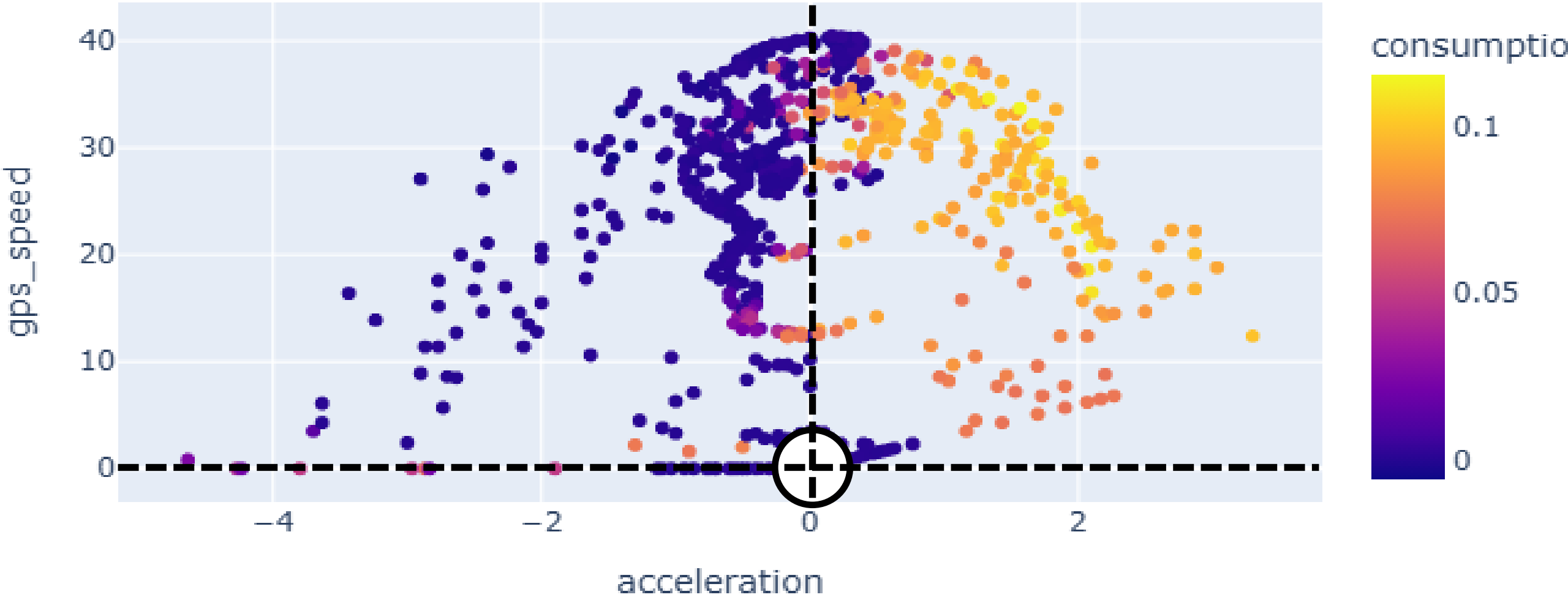


Driver **#3**

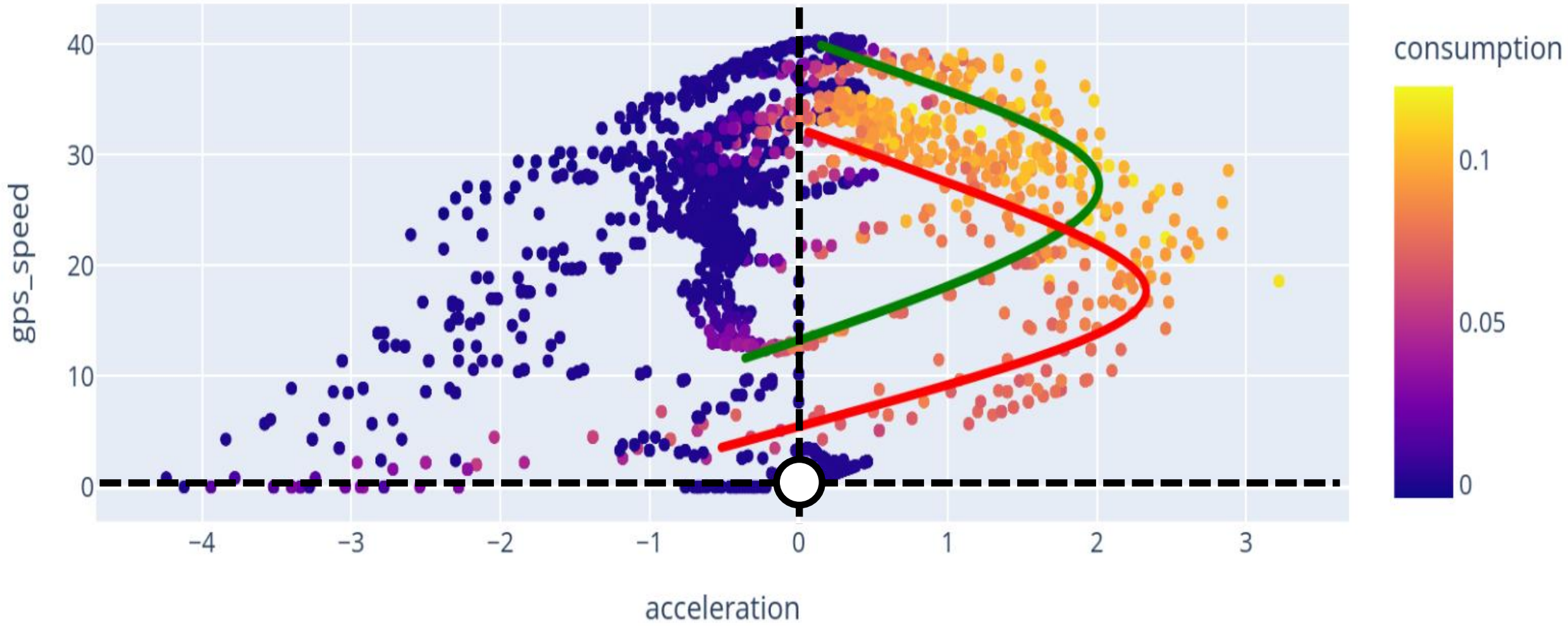


Data as a storyteller

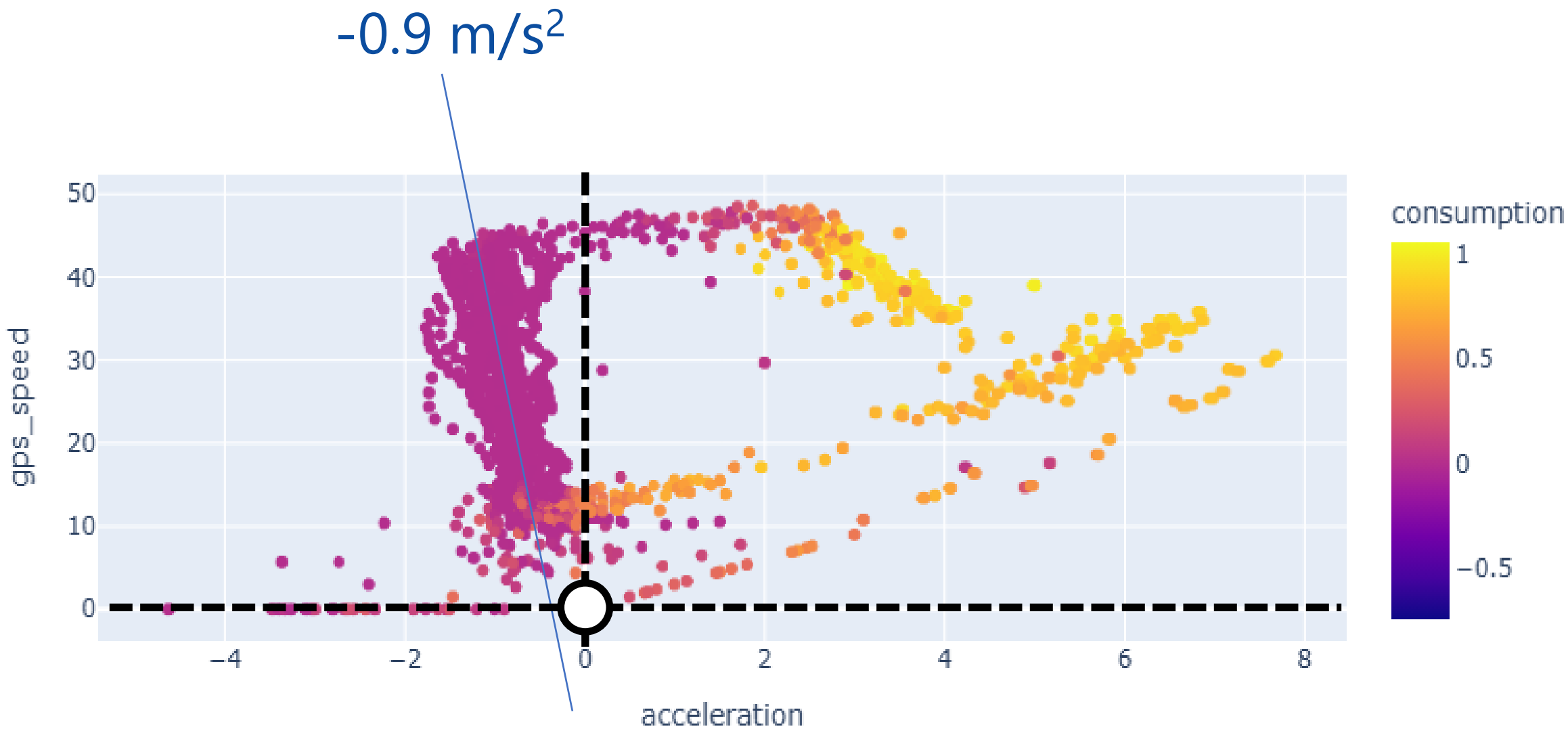
Label	Unit	Description
acceleration	m/s^2	How much the speed changes per second
consumption	W/s	How much energy is used per second



Data as a storyteller



Data as a storyteller: e.g. Coastdown-Testing



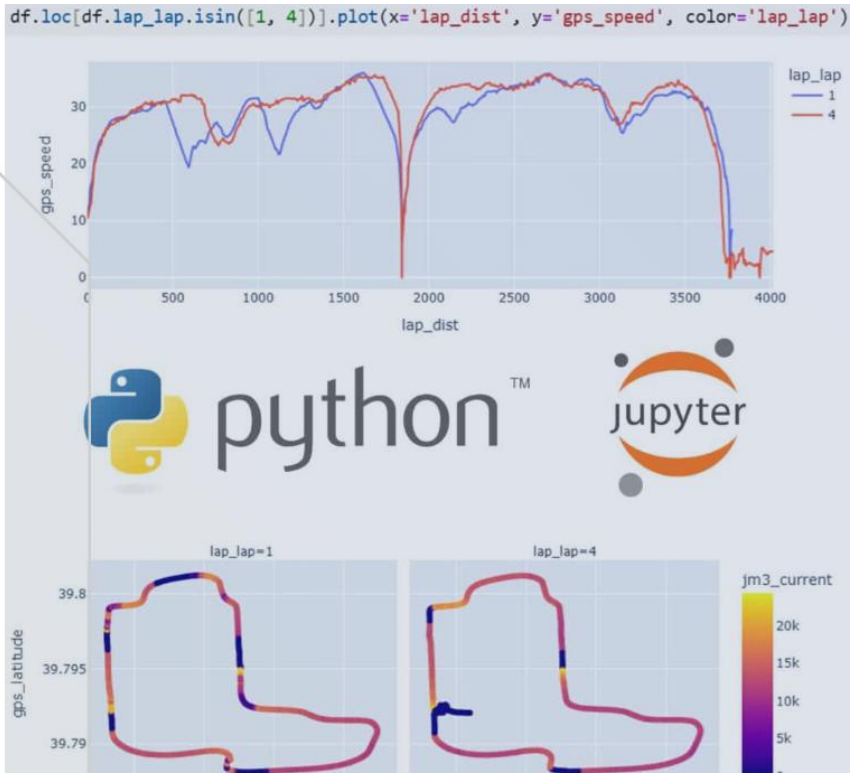
Python **Sandbox #1** to understand Race Data

Programming Sandbox #1

JUPYTER NOTEBOOK UND PYTHONCODE FOR LEVEL-3

Start Jupyter Notebook in your browser (Binder, ~1–2 min.) and play with Python. Start with real race data (speed, GPS) and calculate accelerations and energy consumption. Save the source code locally, because when you close the browser, the environment will be gone.

START SANDBOX #1 IN YOUR BROWSER ▷



pandas

plotly

NumPy

Race Strategy

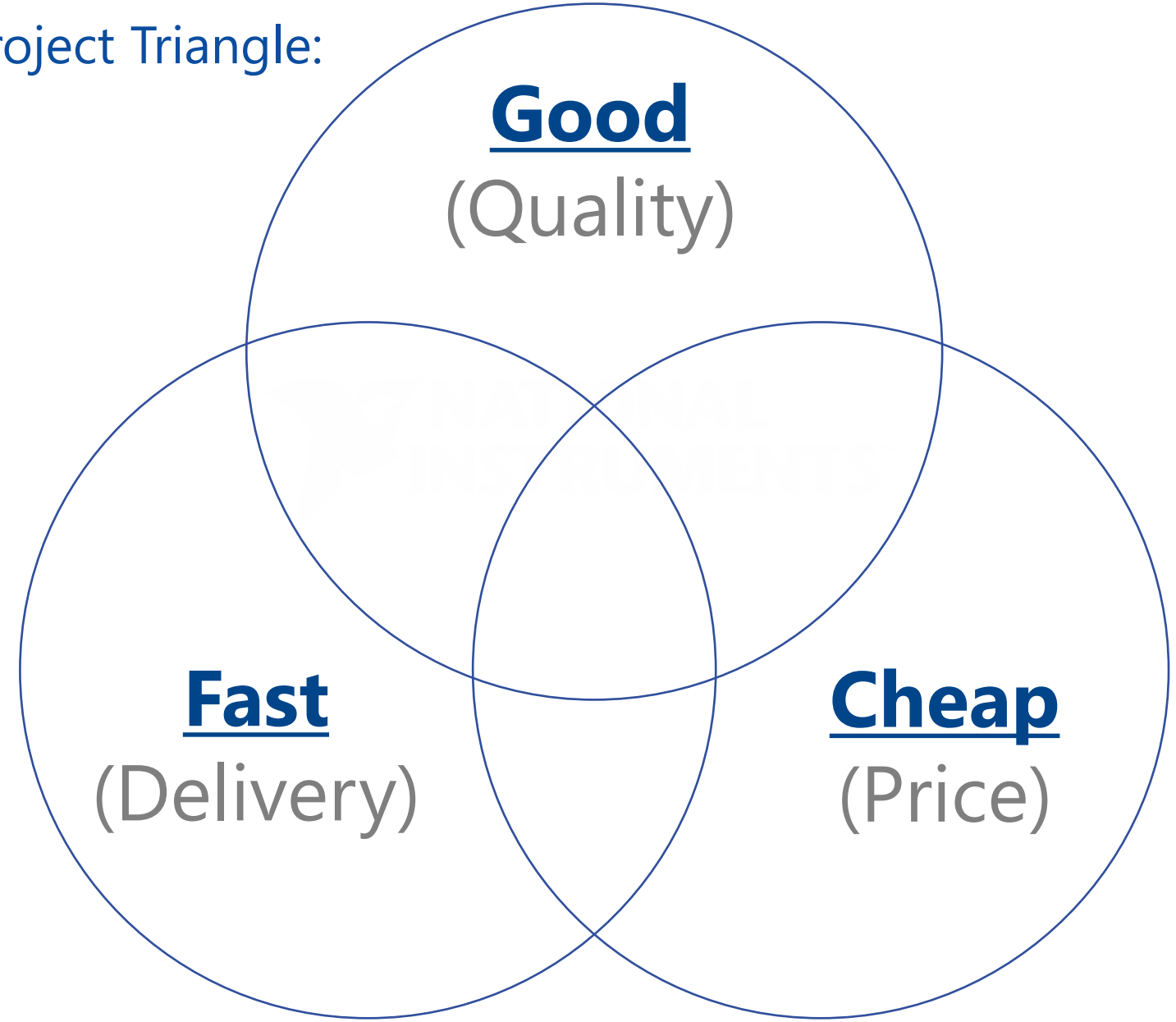
Data- and Modeldriven Approach

Level 4

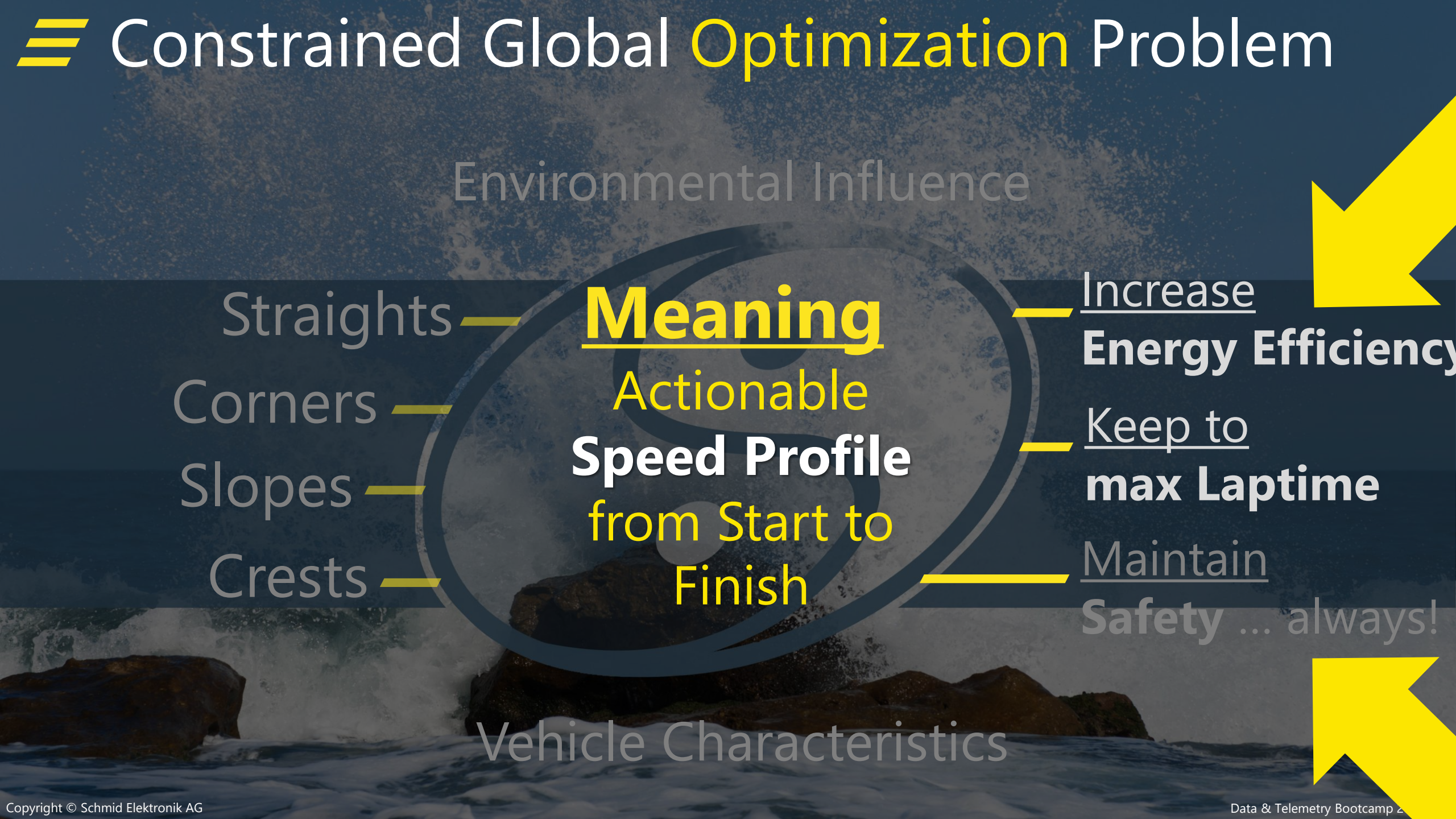
What are Conflicting Goals?

Conflicting Goals in Industrial Projects

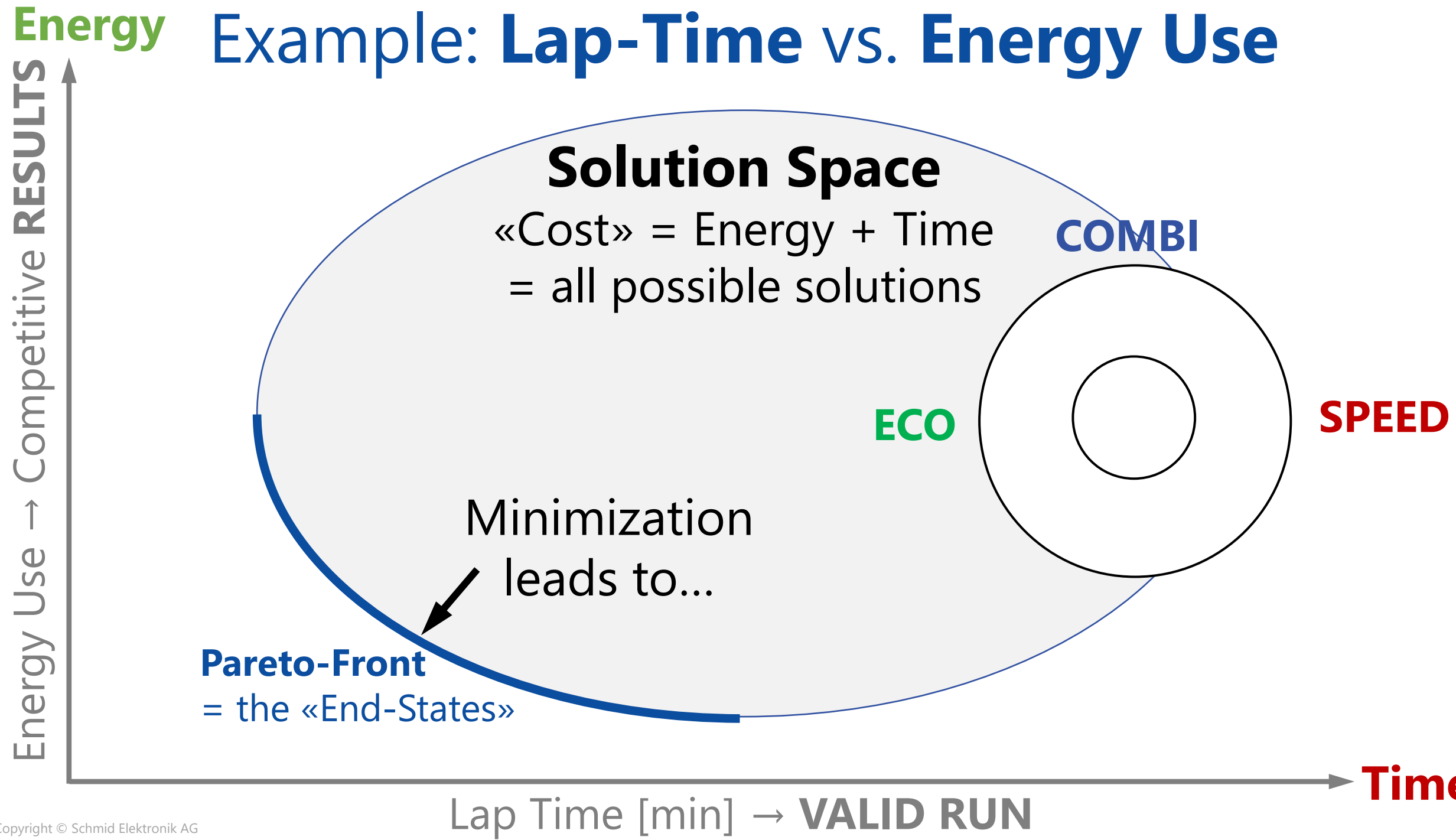
The Magic Project Triangle:



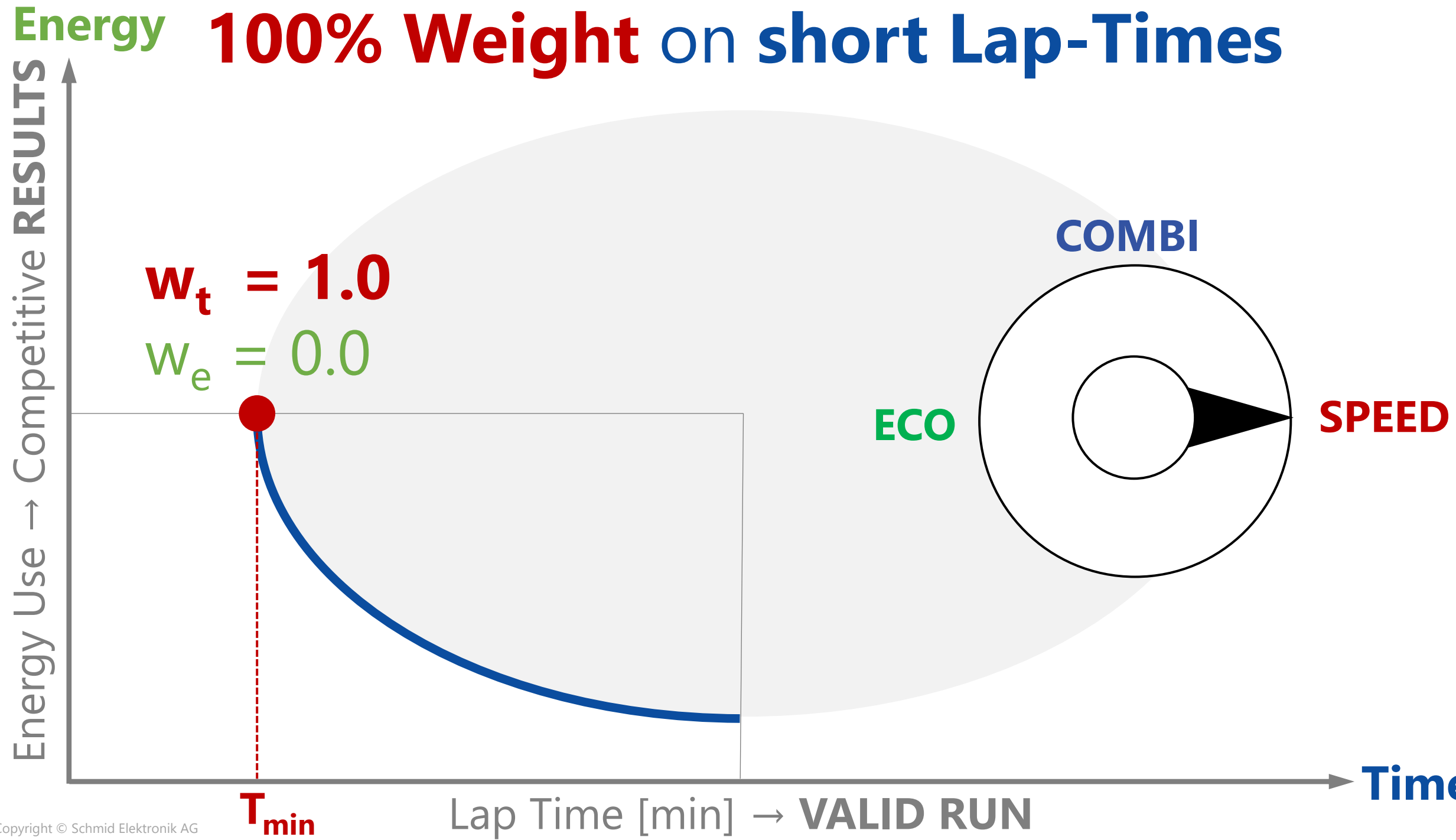
Constrained Global Optimization Problem



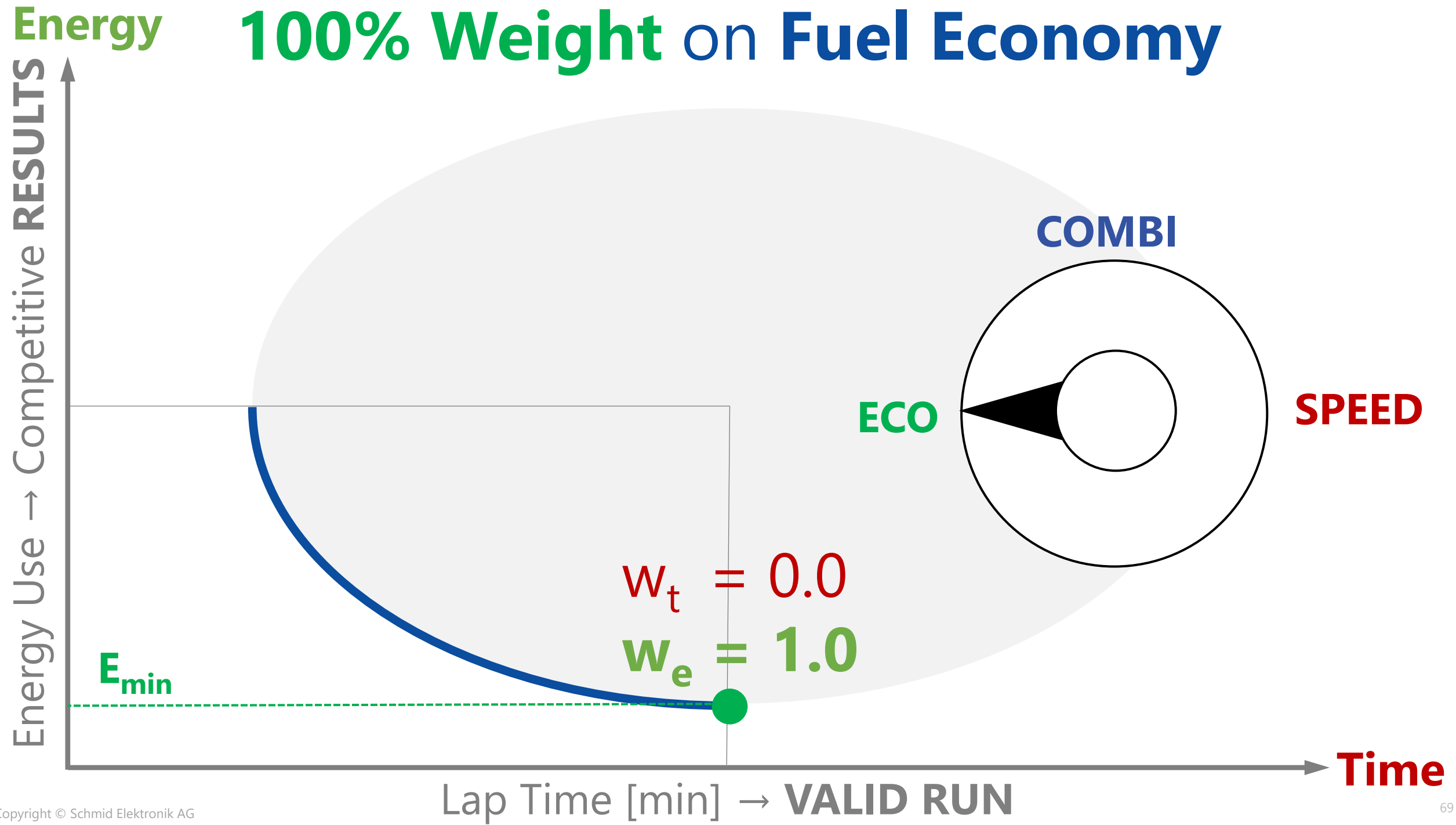
Example: **Lap-Time** vs. **Energy Use**



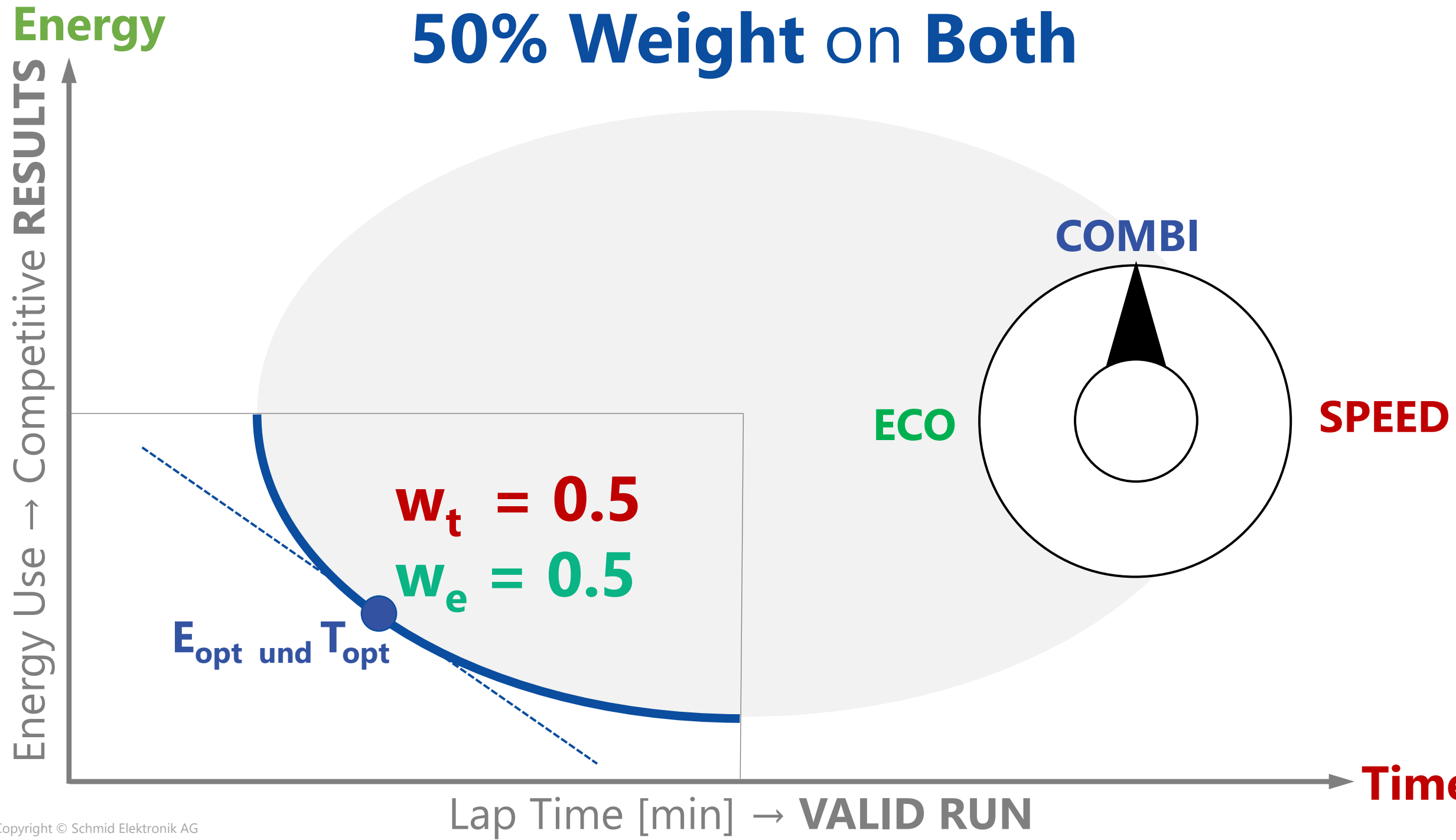
100% Weight on short Lap-Times



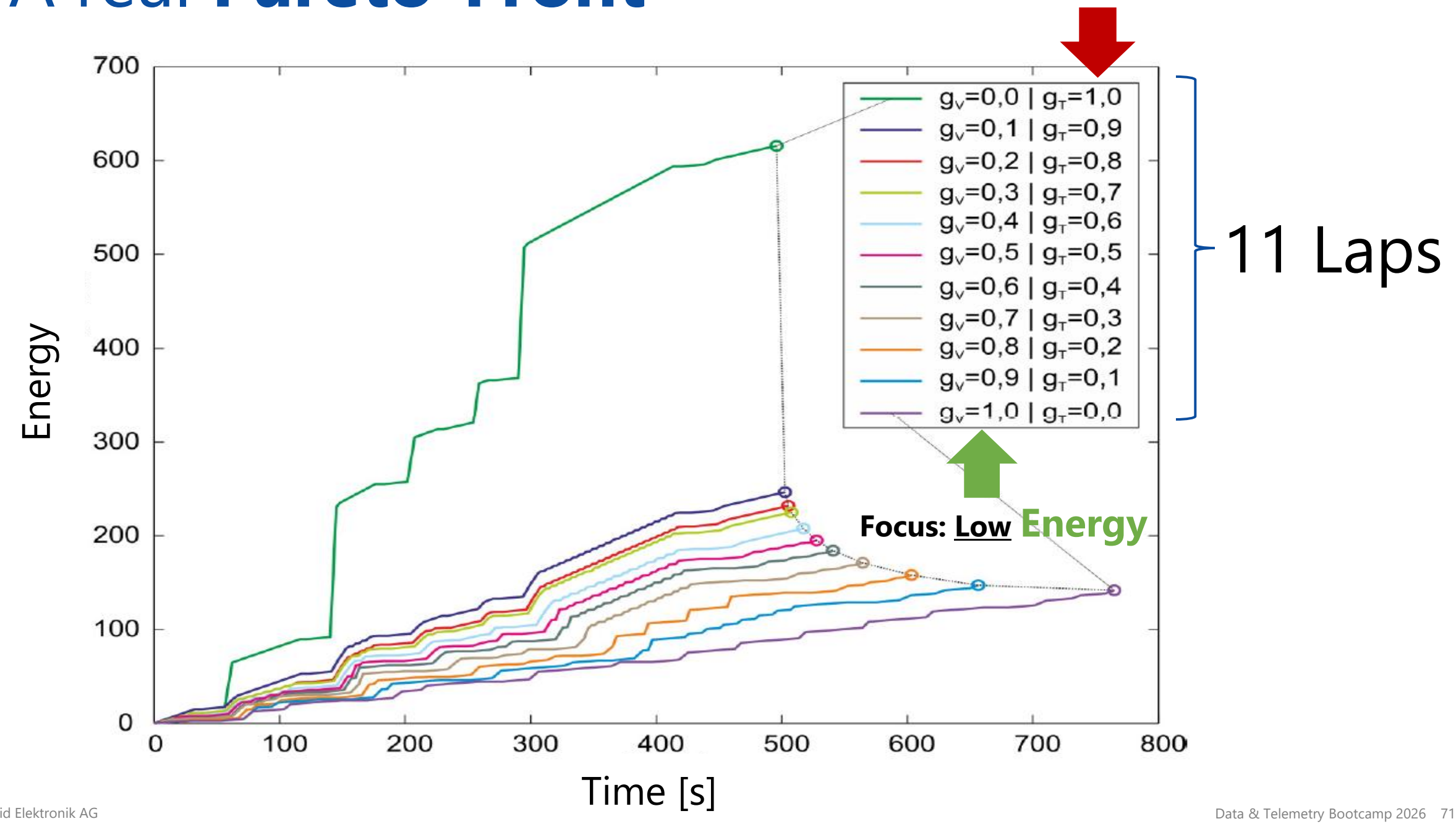
100% Weight on Fuel Economy



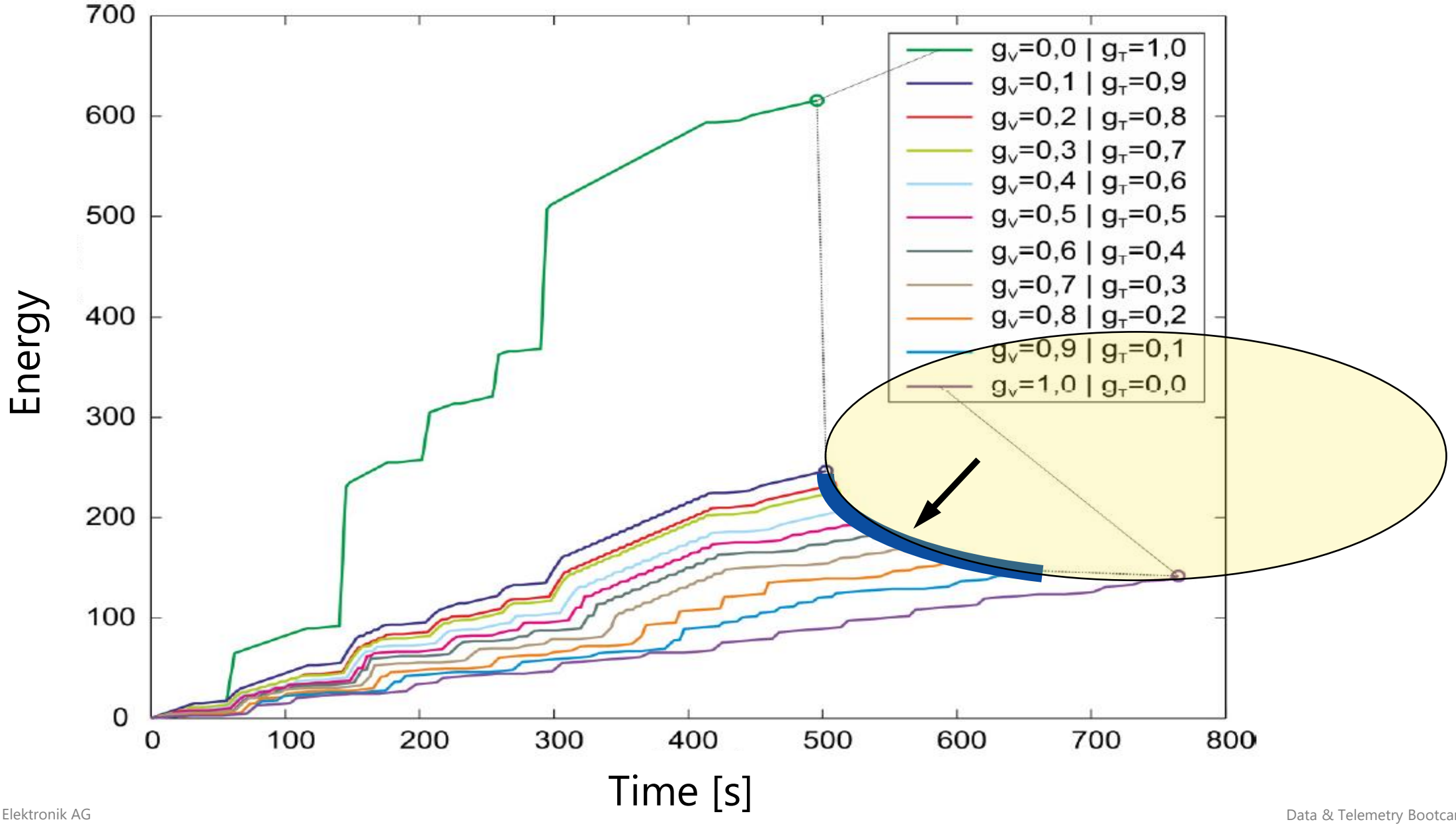
50% Weight on Both



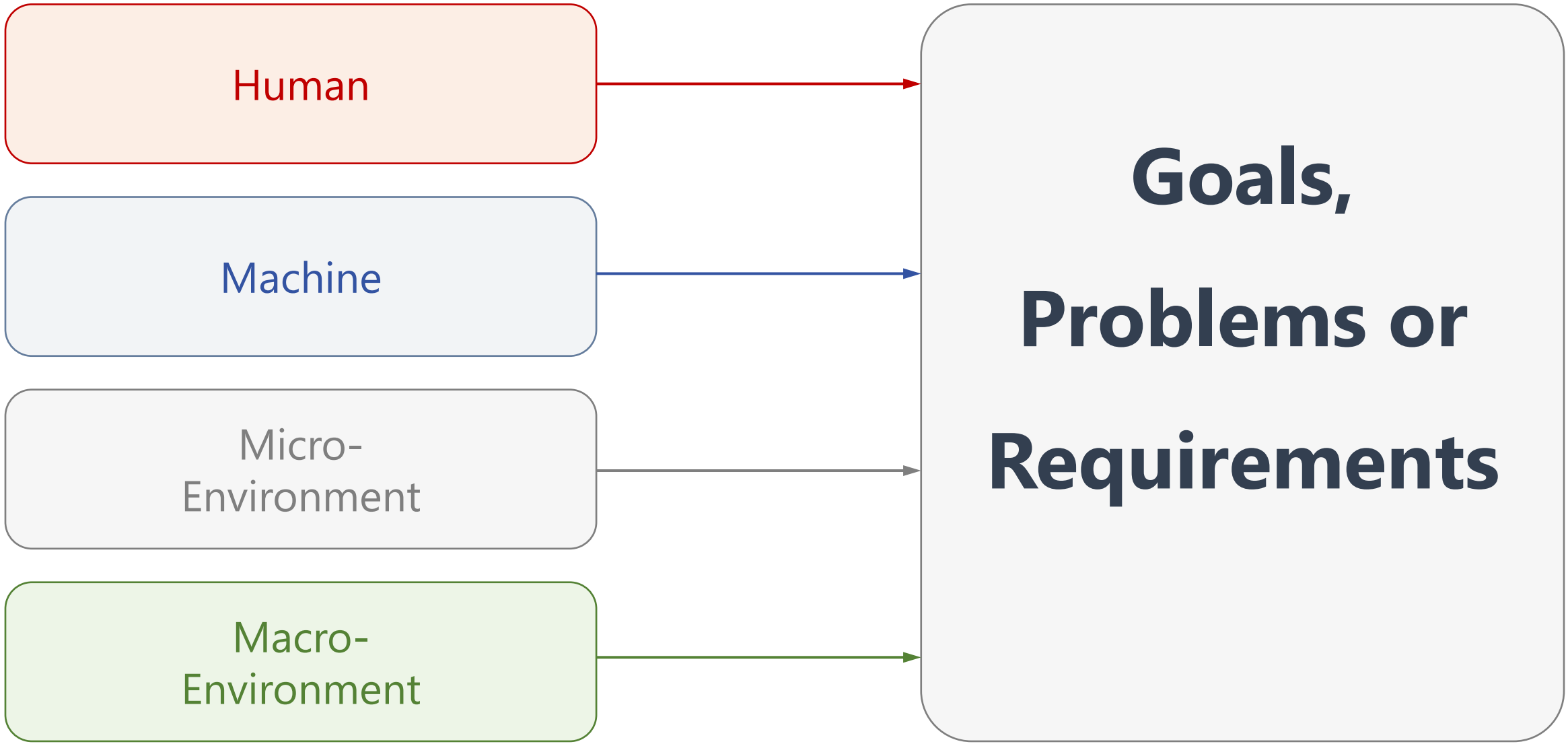
A real **Pareto-Front**

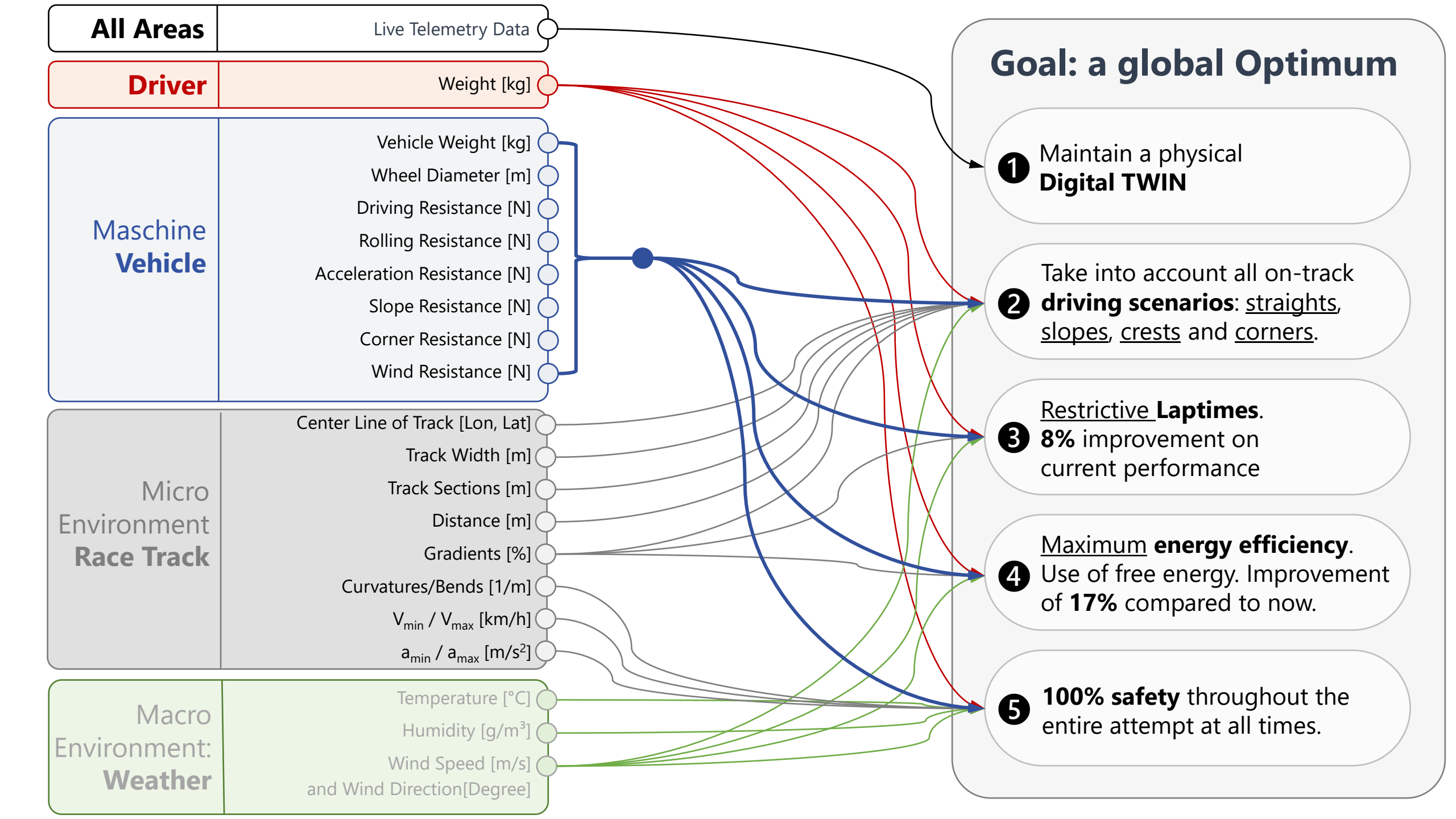


A real **Pareto-Front**



Scaling the Data Strategy




All Areas
Live Telemetry Data
Driver
Weight [kg]
Maschine
Vehicle
Vehicle Weight [kg]
Wheel Diameter [m]
Driving Resistance [N]
Rolling Resistance [N]
Acceleration Resistance [N]
Slope Resistance [N]
Corner Resistance [N]
Wind Resistance [N]
Micro
Environment
Race Track
Center Line of Track [Lon, Lat]
Track Width [m]
Track Sections [m]
Distance [m]
Gradients [%]
Curvatures/Bends [1/m]
V_{min} / V_{max} [km/h]
a_{min} / a_{max} [m/s²]
Macro
Environment:
Weather
Temperature [°C]
Humidity [g/m³]
Wind Speed [m/s]
and Wind Direction[Degree]
Goal: a global Optimum
1 Maintain a physical Digital TWIN
2 Take into account all on-track driving scenarios: straights, slopes, crests and corners.
3 Restrictive Laptimes. 8% improvement on current performance
4 Maximum energy efficiency. Use of free energy. Improvement of 17% compared to now.
5 100% safety throughout the entire attempt at all times.

Driver

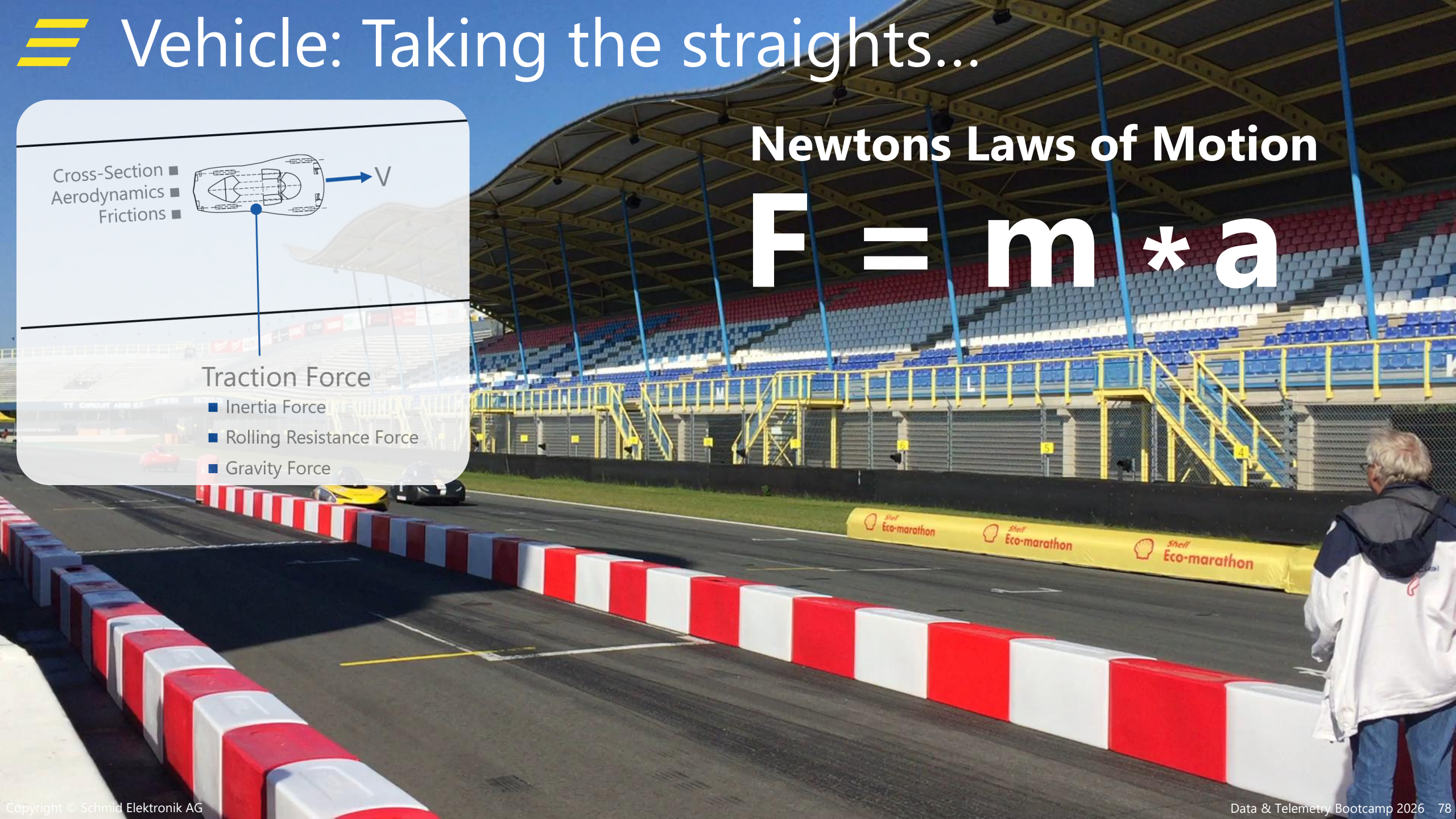
Fitness

- Heartbeat
- Eye movement
- Cognitive Load

Behavior

- Throttle
- Brake
- Steering Wheel

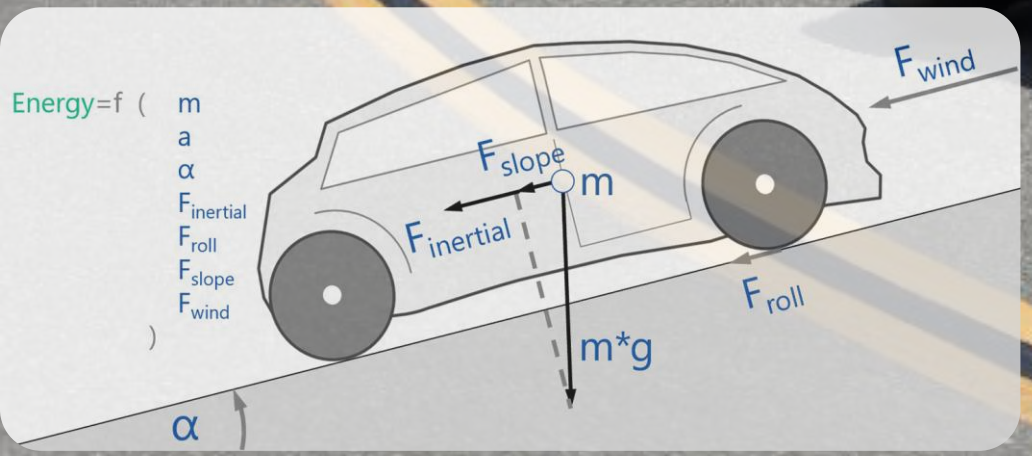
Vehicle: Taking the straights...



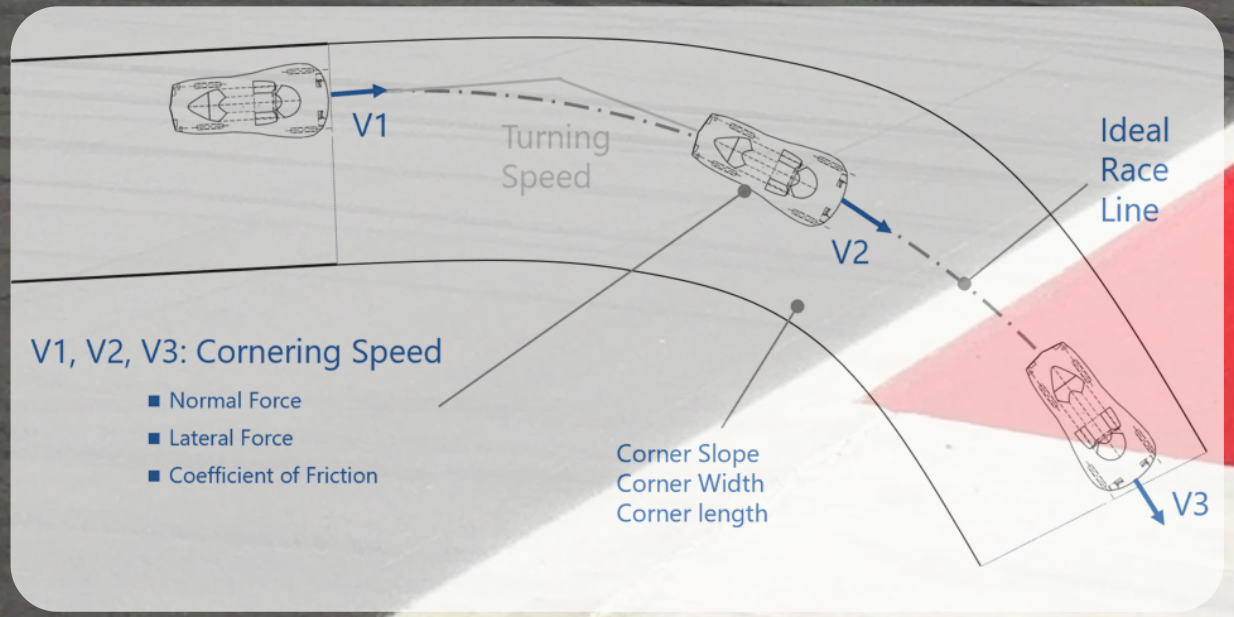
Vehicle: Managing the slopes ...

$$dE = \frac{1}{3600}\left[mg(f\cos\varphi + \sin\varphi) + \frac{1}{2}(\rho C_x A \frac{(v_{EV} + v_w)^2}{3.6}) + \left(m + m_f\right)\frac{dv}{dt}\right] ds$$

dE	Mechanical energy required at the wheels to drive a distance ds [kWh]
m	Total vehicle mass [kg]
m_f	Fictive mass of rolling inertia [kg]
g	Gravitational acceleration [m/s^2]
f	Vehicle coefficient of rolling resistance [-]
φ	Road gradient angle [°]
ρ	Air density [kg/m^3]
C_x	Drag coefficient of the vehicle [-]
A	Vehicle equivalent cross section [m^2]
v_{EV}	Vehicle speed between the point i and the point j [km/h]
v_w	Wind speed projected to the opposing direction of the driving direction [km/h]
ds	Distance driven from point i to point j [km]



Vehicle: The corners are the holy grail

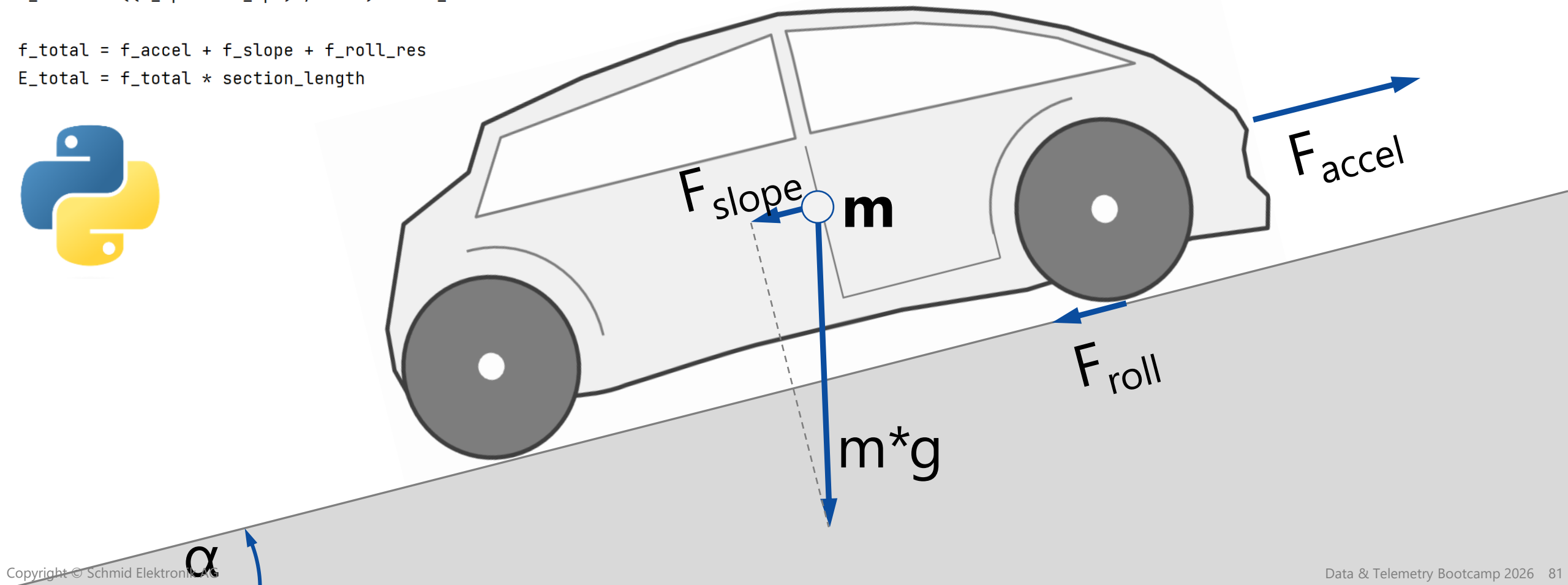


Vehicle Model

$$F_{total} = F_{roll} + F_{slope} + F_{accel}$$

```
vold_mps = vold/3.6
v_mps = v/3.6
time = 2 * section_length / (v_mps + vold_mps)
gradient_angle = math.tanh( (config.loc[x, "m above sea"]-config.loc[x-1, "m above sea"] ) / section_length)
f_roll_res = car_mass * g * math.cos(gradient_angle) * coeff_roll_res
f_slope = car_mass * g * math.sin(gradient_angle)
f_accel = ((v_mps-vold_mps) / time) * car_mass

f_total = f_accel + f_slope + f_roll_res
E_total = f_total * section_length
```

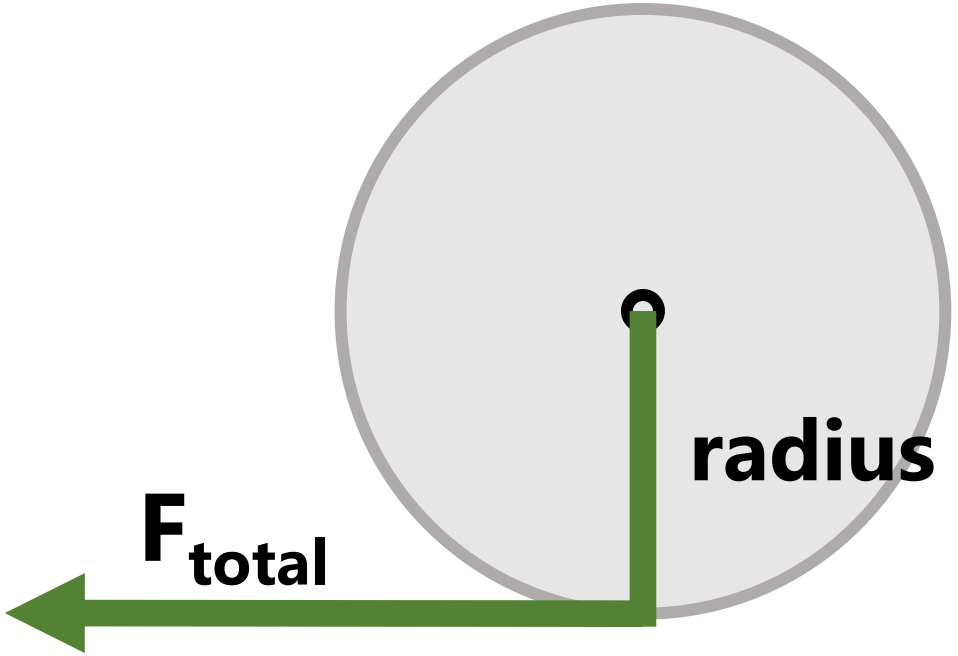


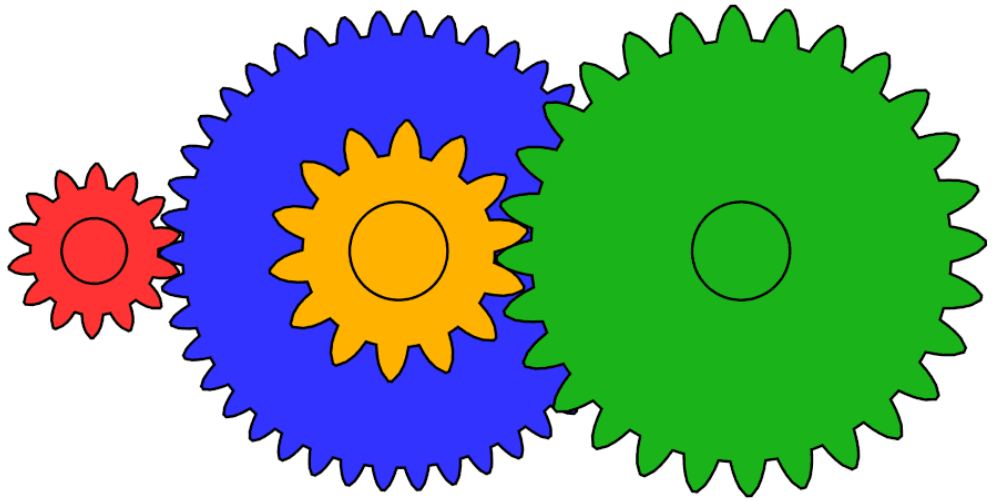
Power calculation for a series wound motor

Total Power P = F_{total} * v

Rotation speed ω = **v** * gear_ratio / radius

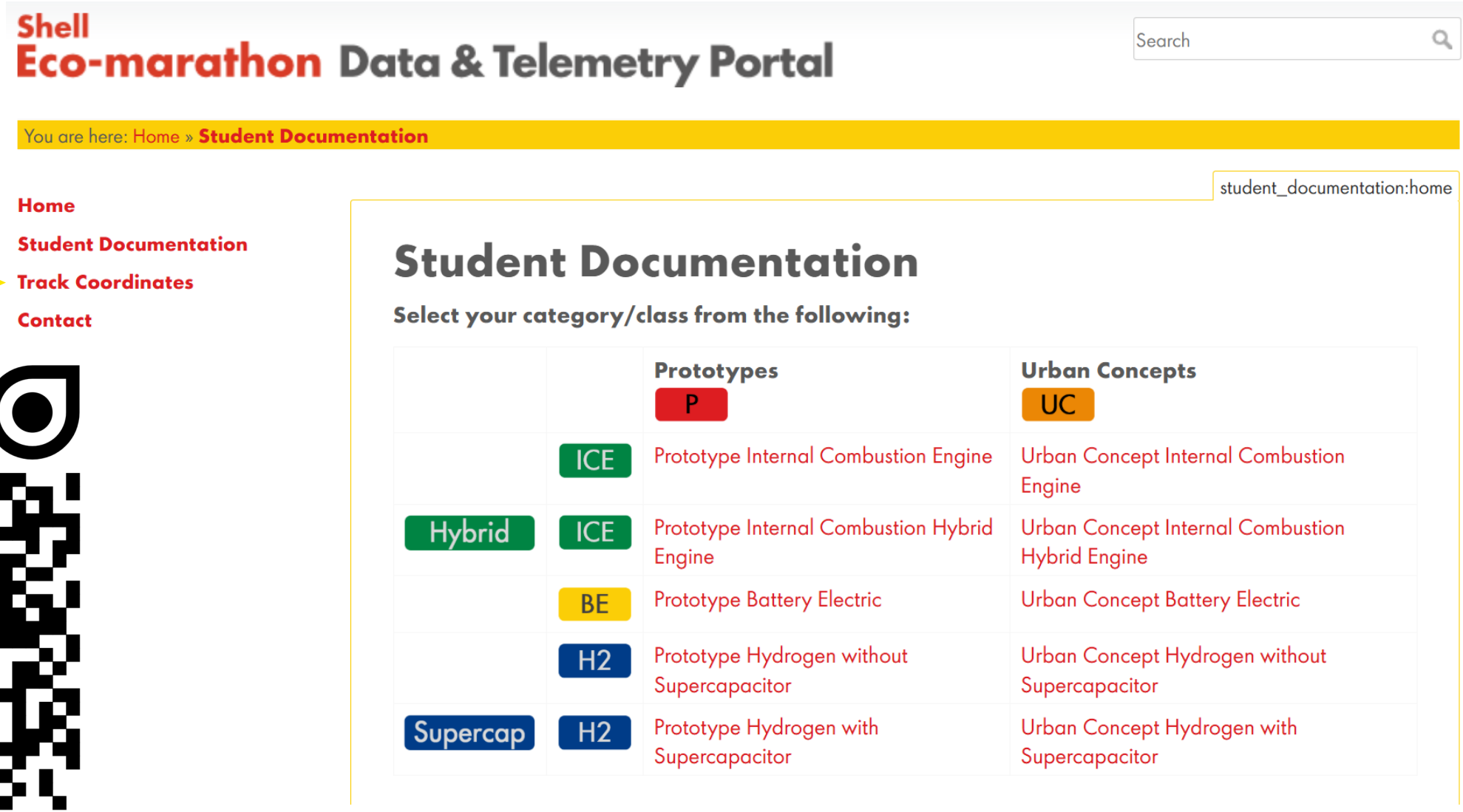
Torque τ = **P** / **ω**

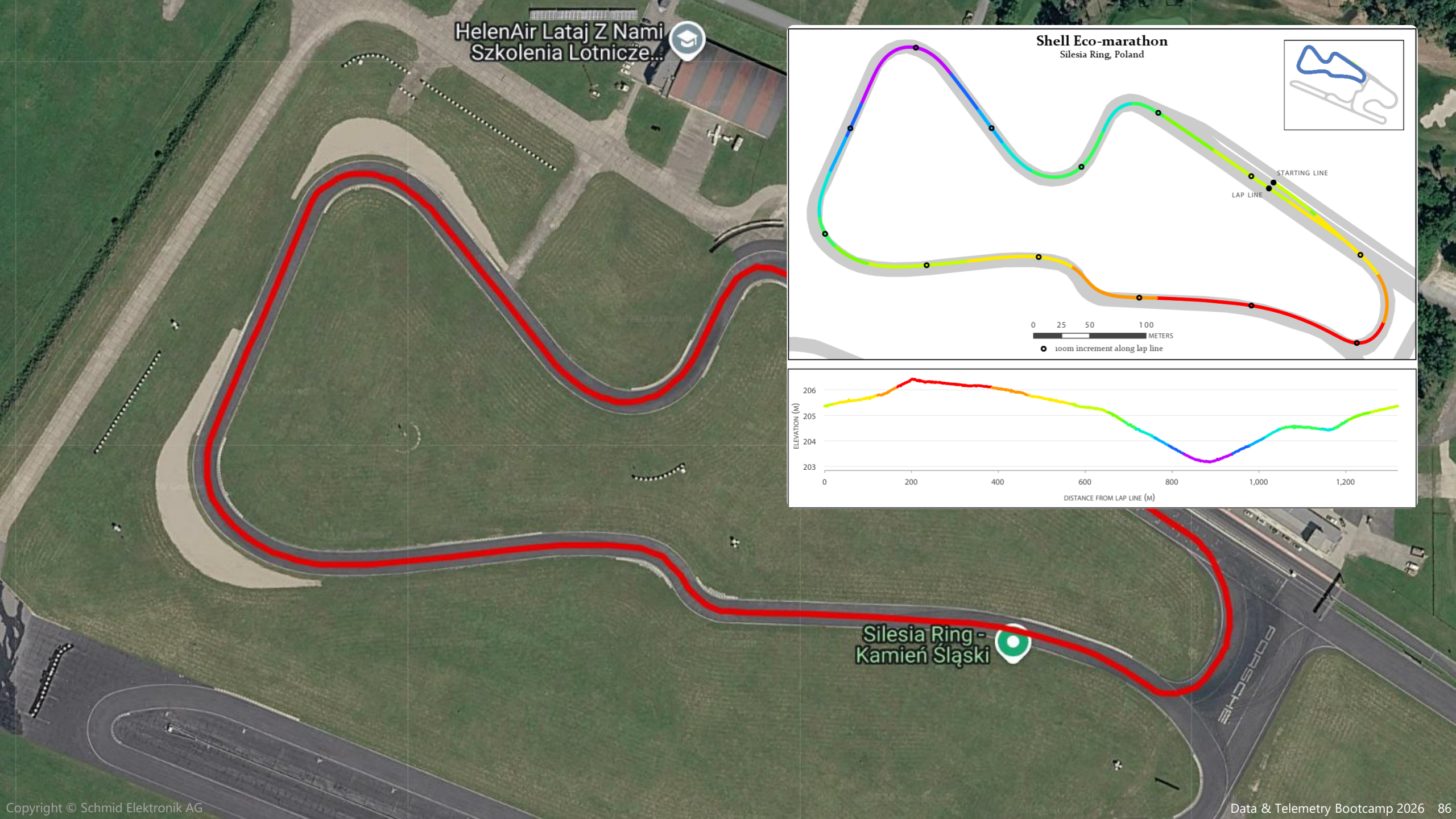




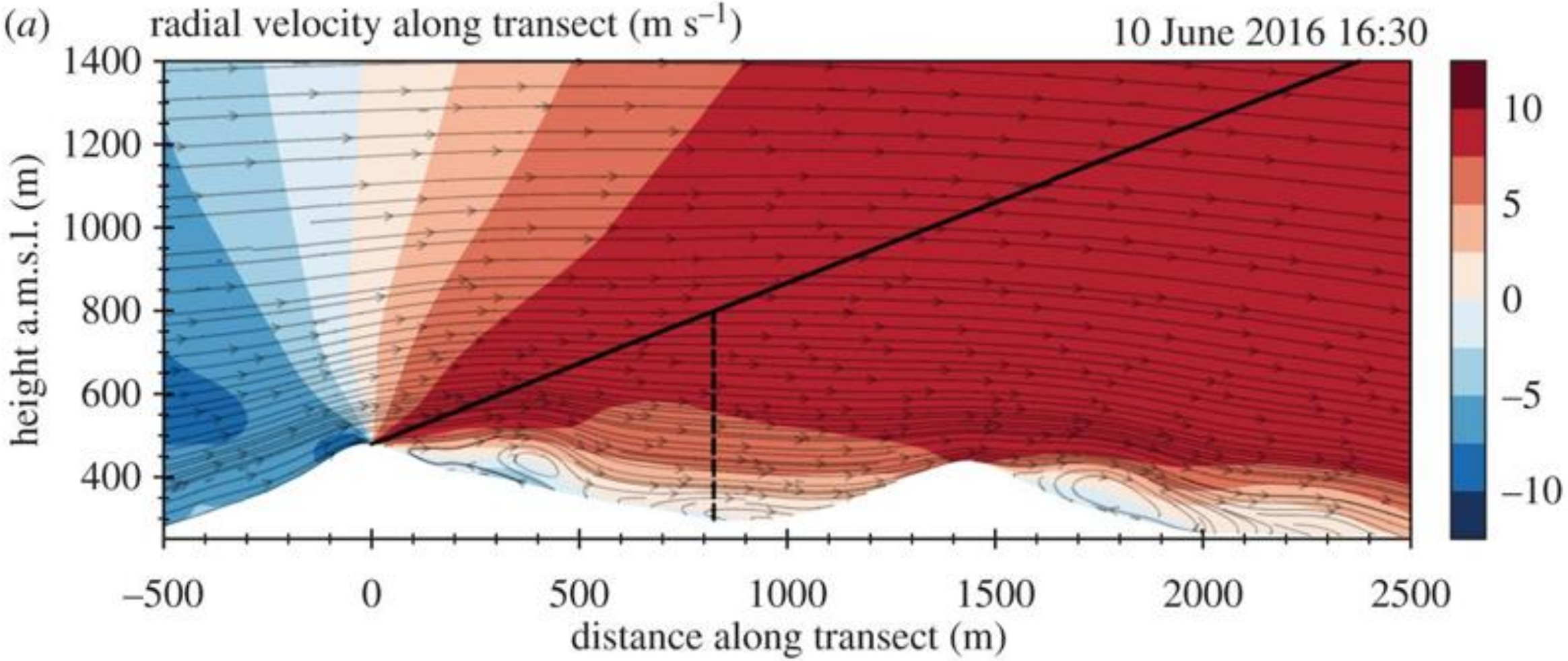
Modelling the Track

https://student.sem-app.com/wiki/doku.php

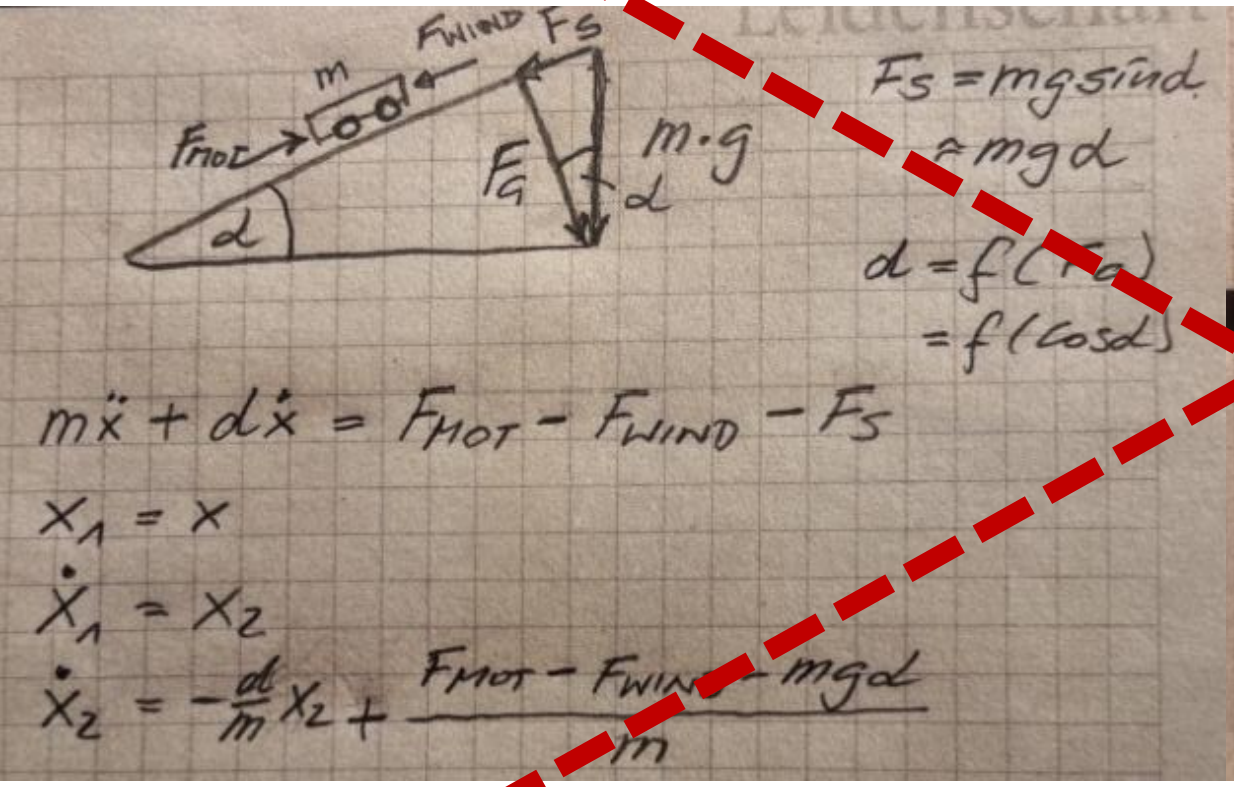


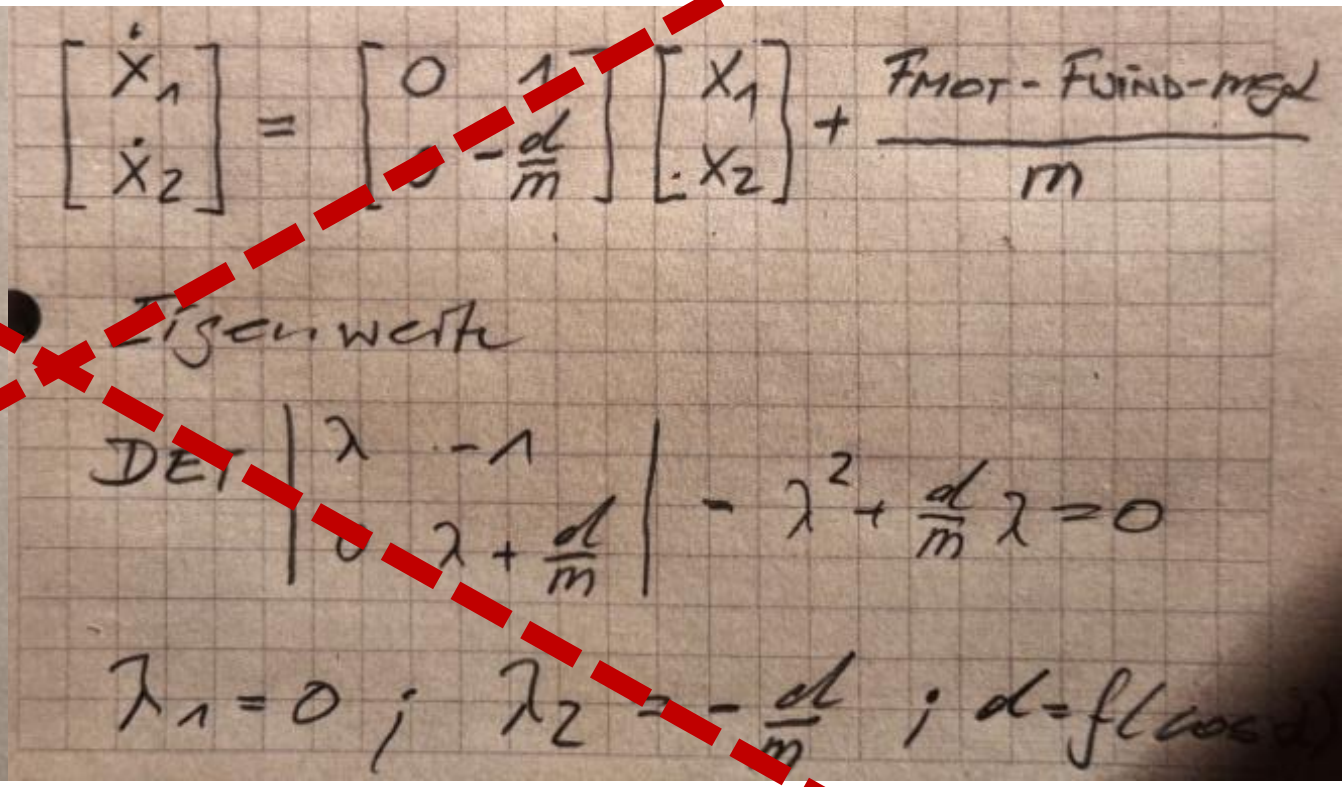


Including the Weather



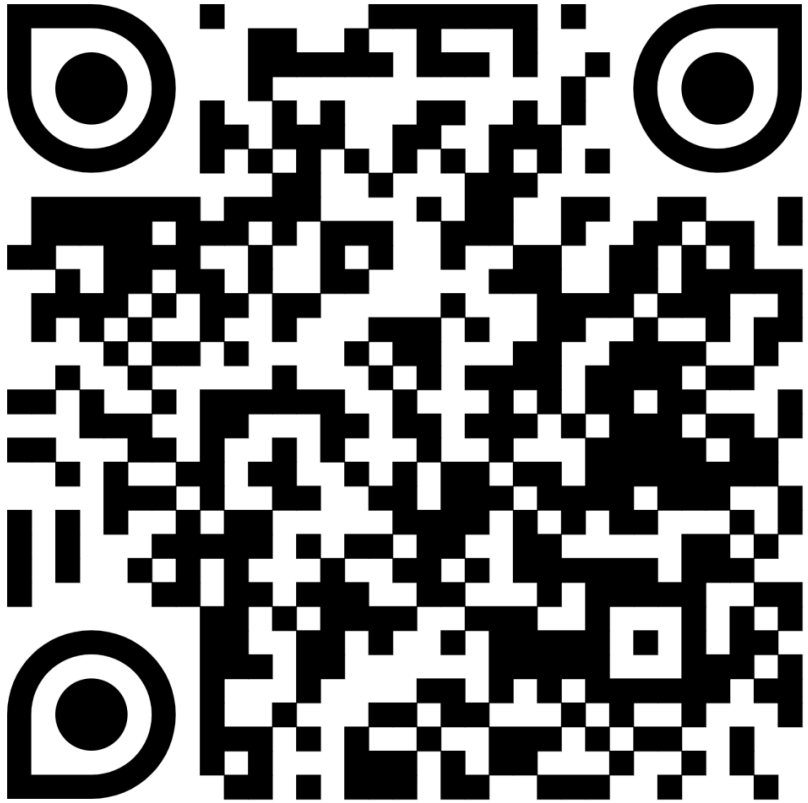
Motorsport = solving **complex** physical Problems





Traditional Way

Python Sandbox **#2** Using Knowledge Graphs



```
nodelist = nx.dijkstra_path(G, "x0_k0",f"x{x_end}_k{k_end}")
outputlist = []
for i in range(len(config.index)):
    outputlist.append(G.nodes[nodelist[i]]["v"])
```


```
output = pd.DataFrame(outputlist)
output.rename(columns={0:"Speed optimal"}, inplace = True)
output['distance'] = output.index * section_length
```

Programming Sandbox #2

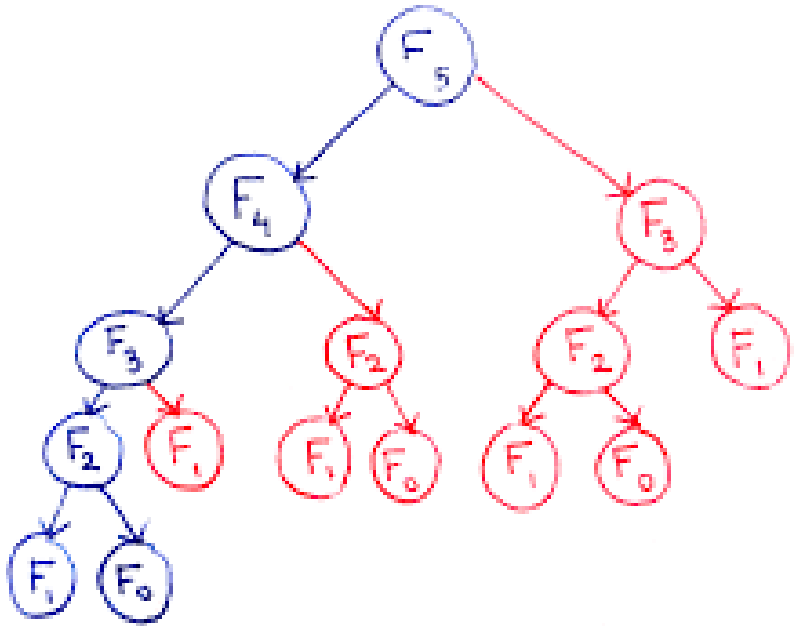
JUPYTER NOTEBOOK AND PYTHONCODE FOR LEVEL-5 (PHYSICAL MODELL)

Start up the Jupyter notebook in your browser (Binder, ~1-2 min.) and play with Python. Learn how to formulate the multigoal problem of data-driven racing with python and how to use a knowledge graph. Save the source code locally, because when you close the browser, the environment will be gone.

START SANDBOX #2 IN YOUR BROWSER ▷

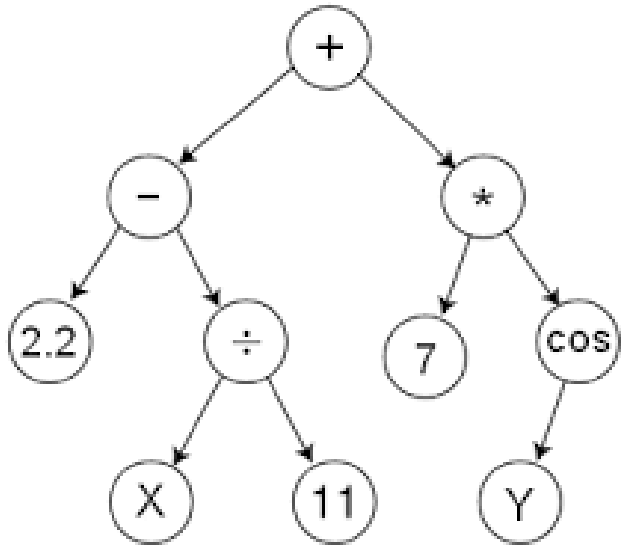
Promising and well established Approaches:

Dynamic Programming



Breaking Complex Problems **down** into simpler subproblems

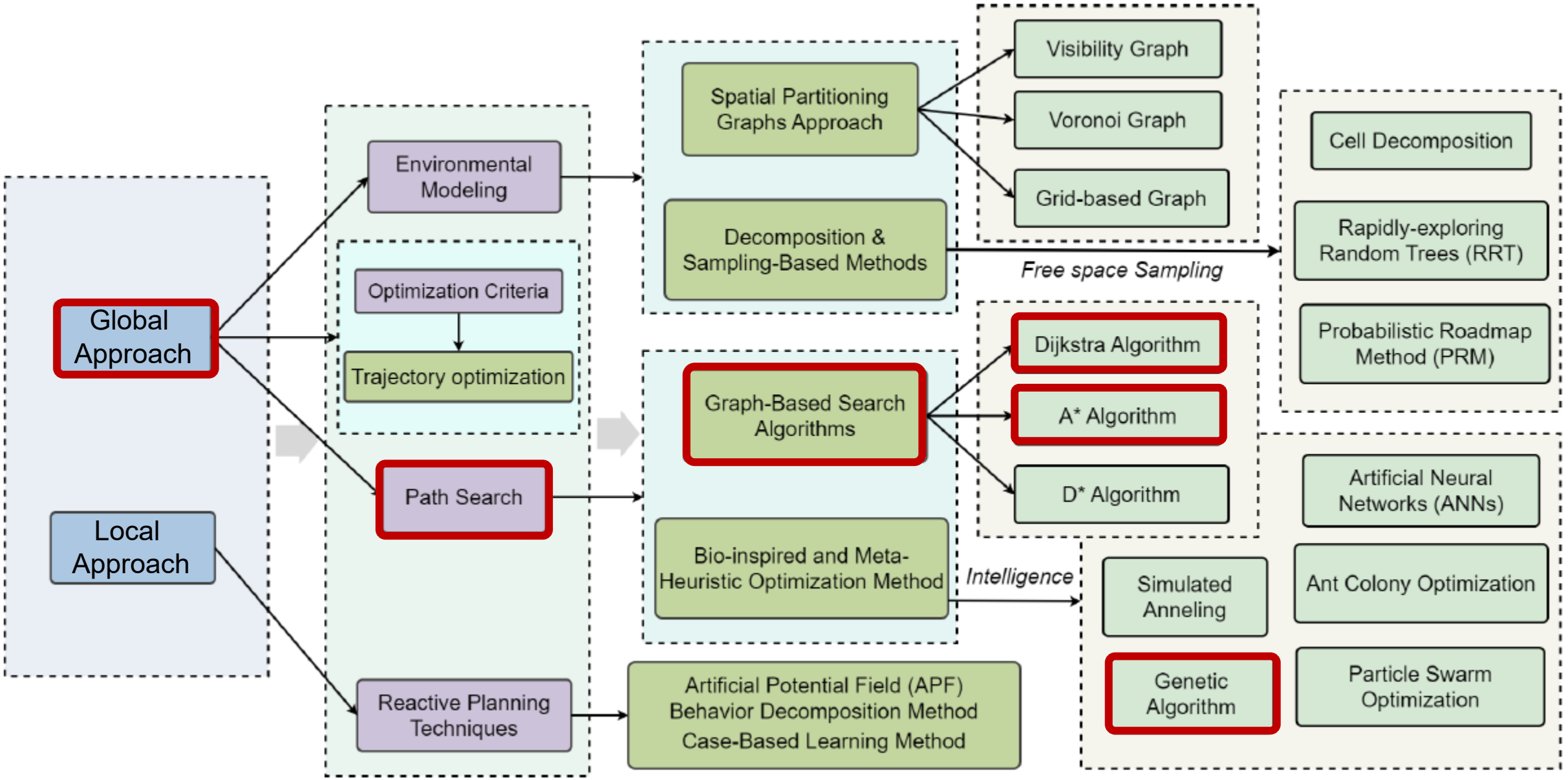
Genetic Programming



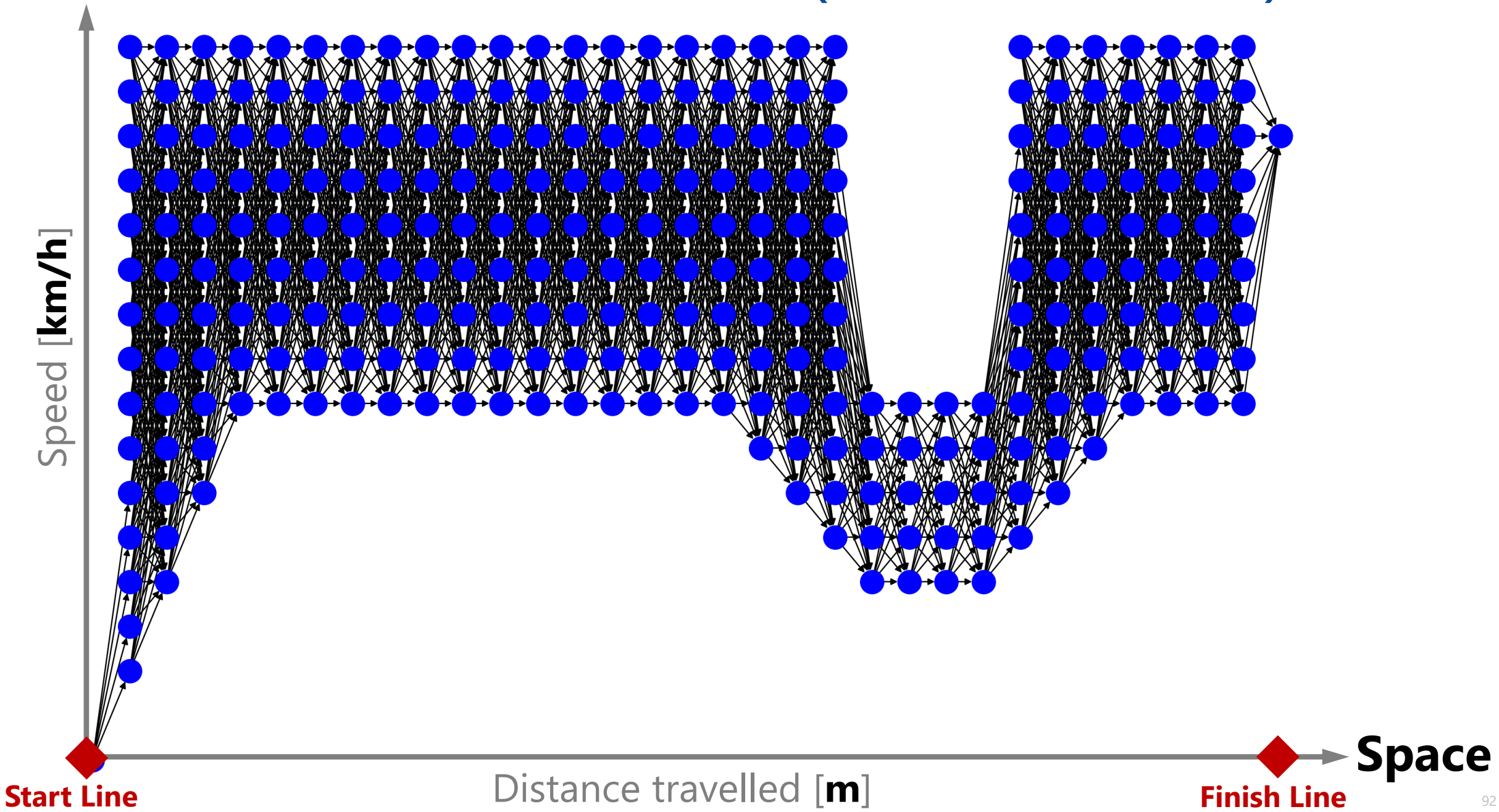
$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$

AI technique inspired by **natural evolution** that evolves computer programs to solve complex problems

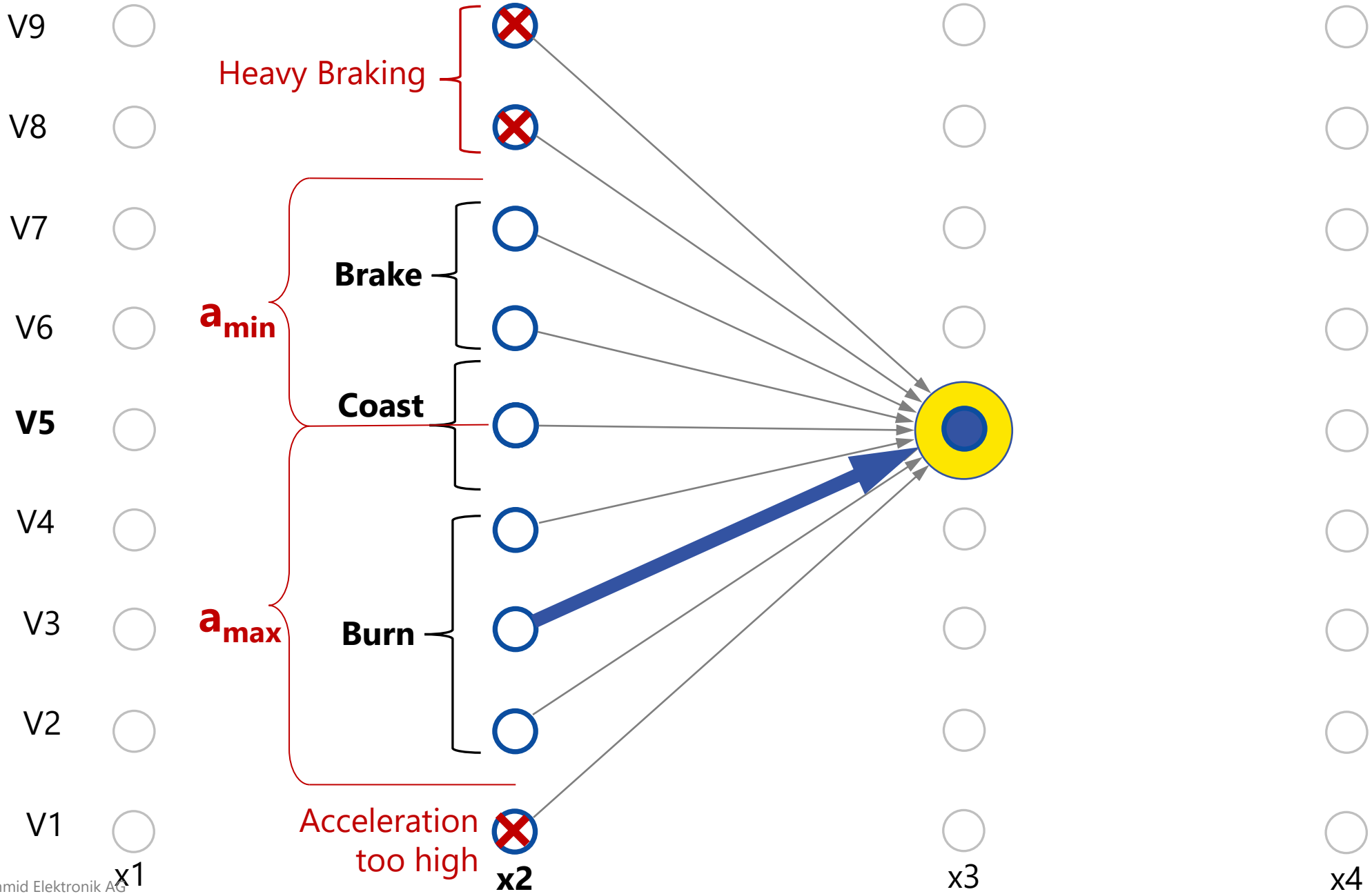
Overview How to Solve Complex Problems



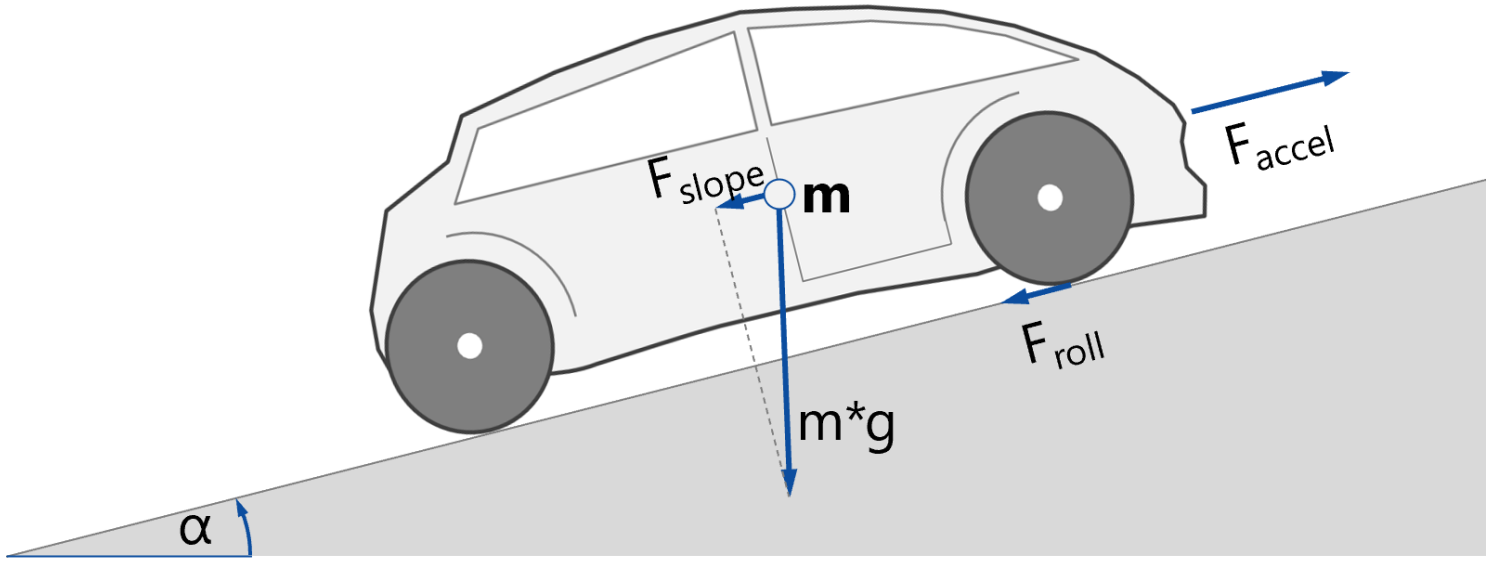
Define the **Nodes** (Vehicle States)

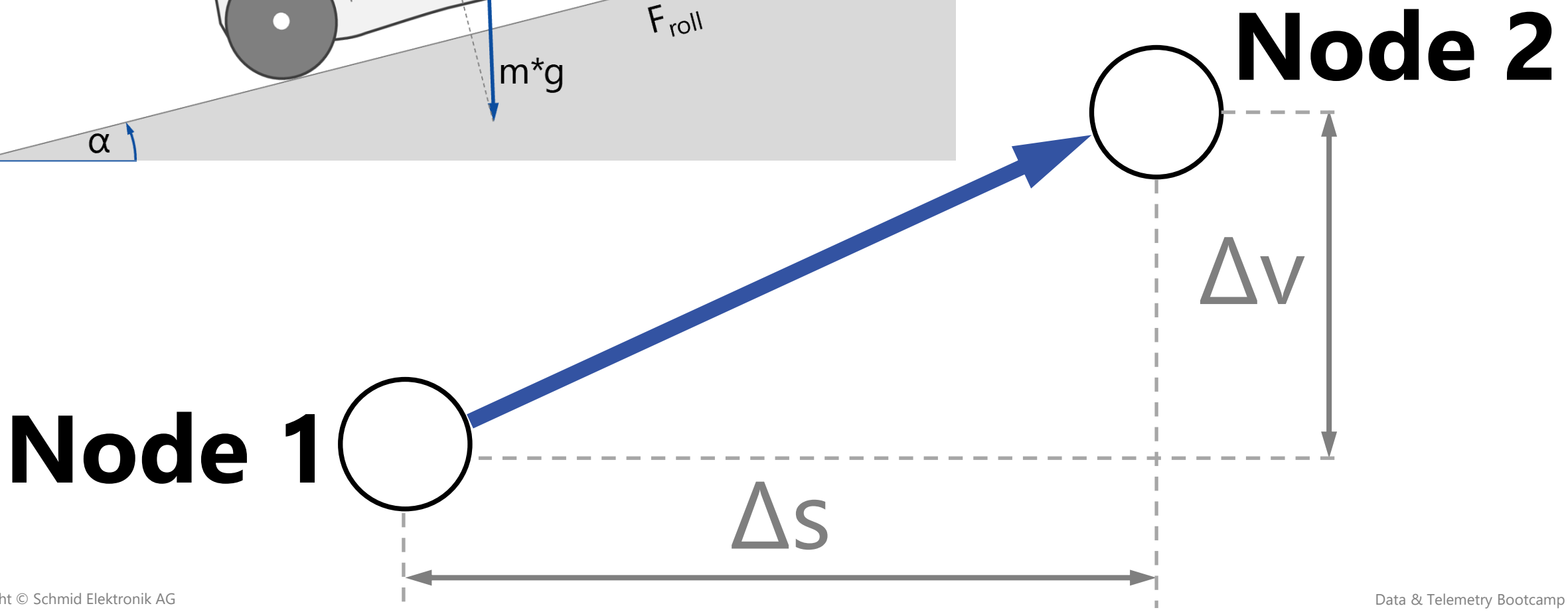


The Graphs **Edges**



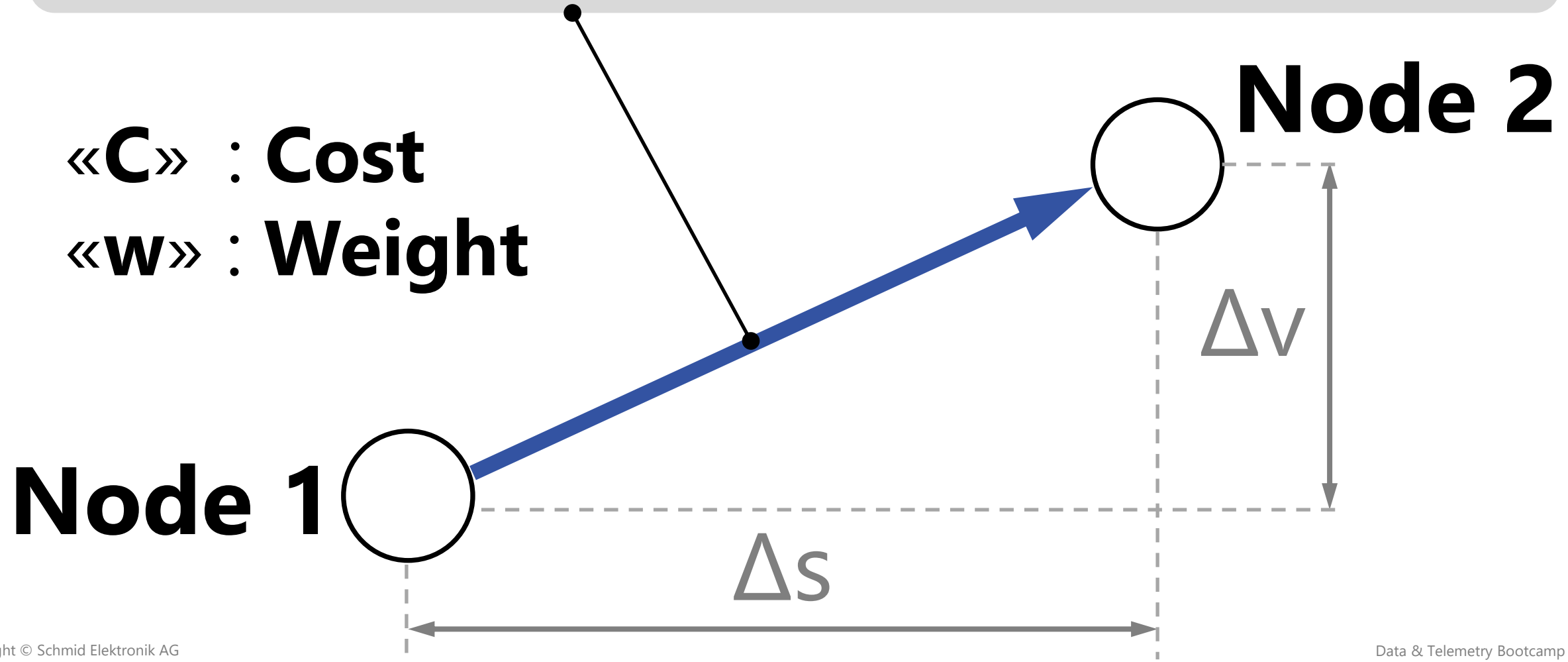
Each **State Change** is a **Decision** with a **Cost**

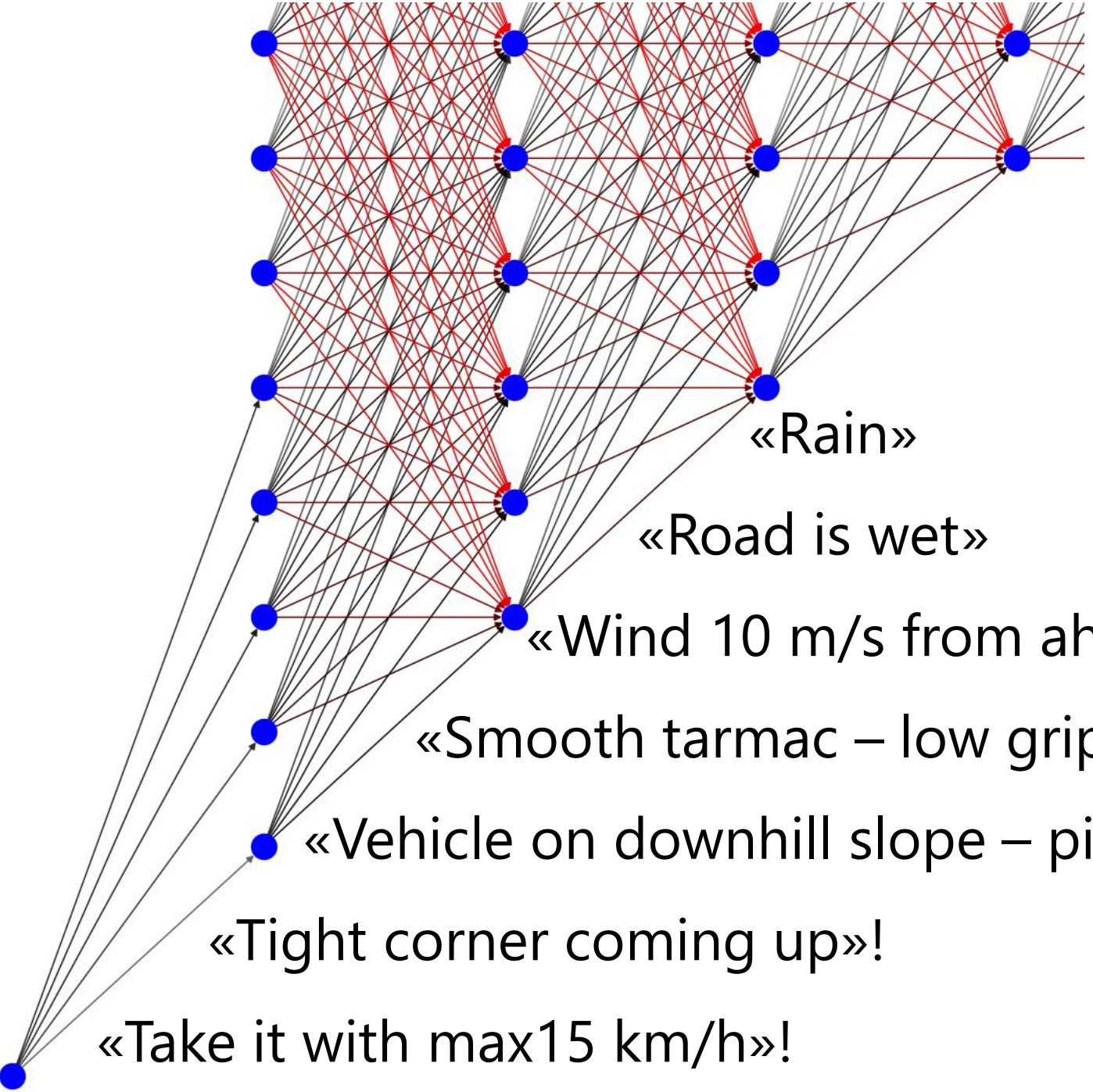




Each **State Change** is a **Decision** with a **Cost**

$$C_{edge} = C_{time} * w_{time} + C_{energy} * w_{energy} \rightarrow \text{MIN!}$$





Meaning

The perfect **speed profile**, **balancing** speed, energy efficiency, safety, vehicle characteristics and external influences

«Rain»

«Road is wet»

«Wind 10 m/s from ahead»

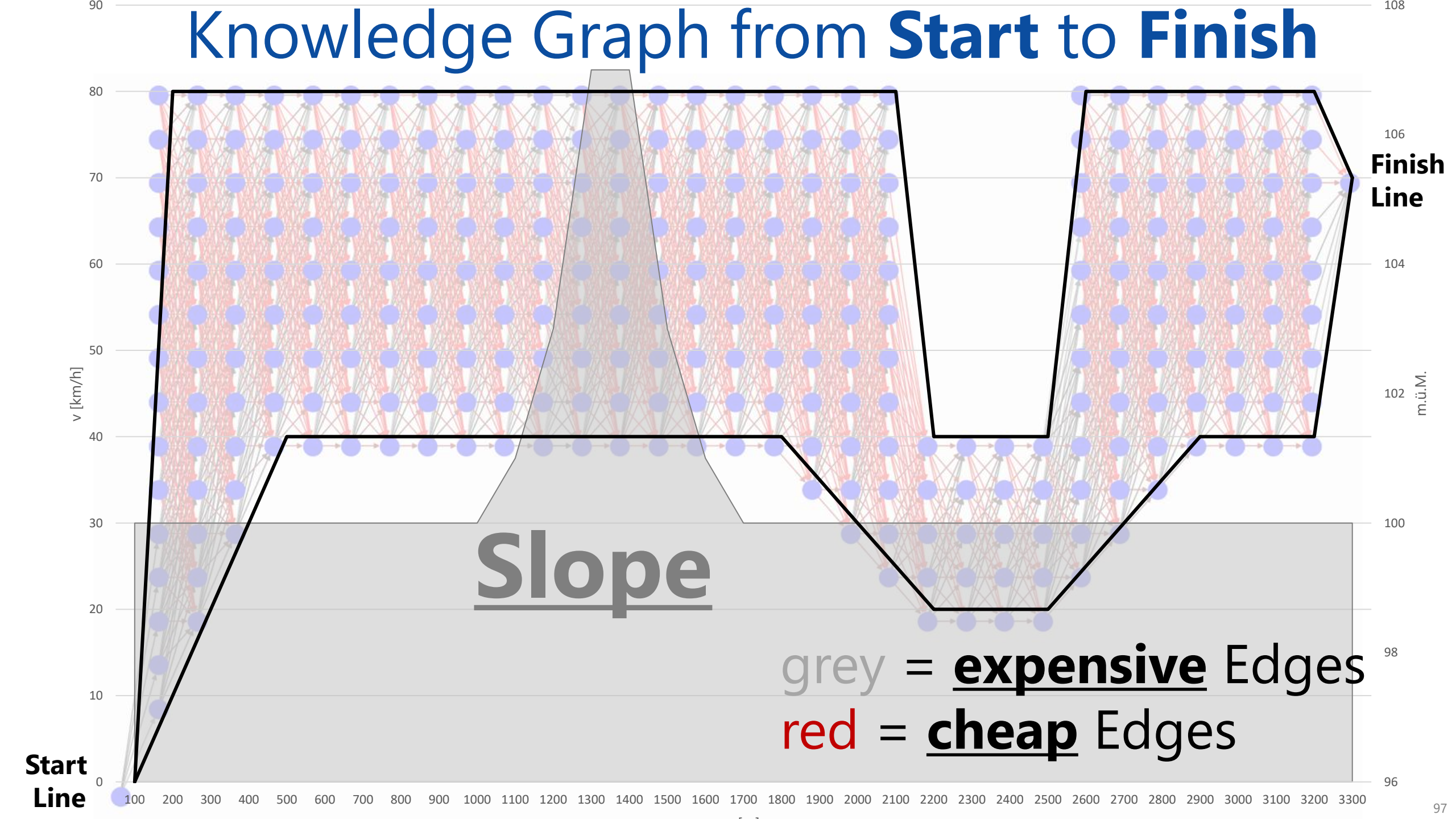
«Smooth tarmac – low grip»

«Vehicle on downhill slope – picking up speed»

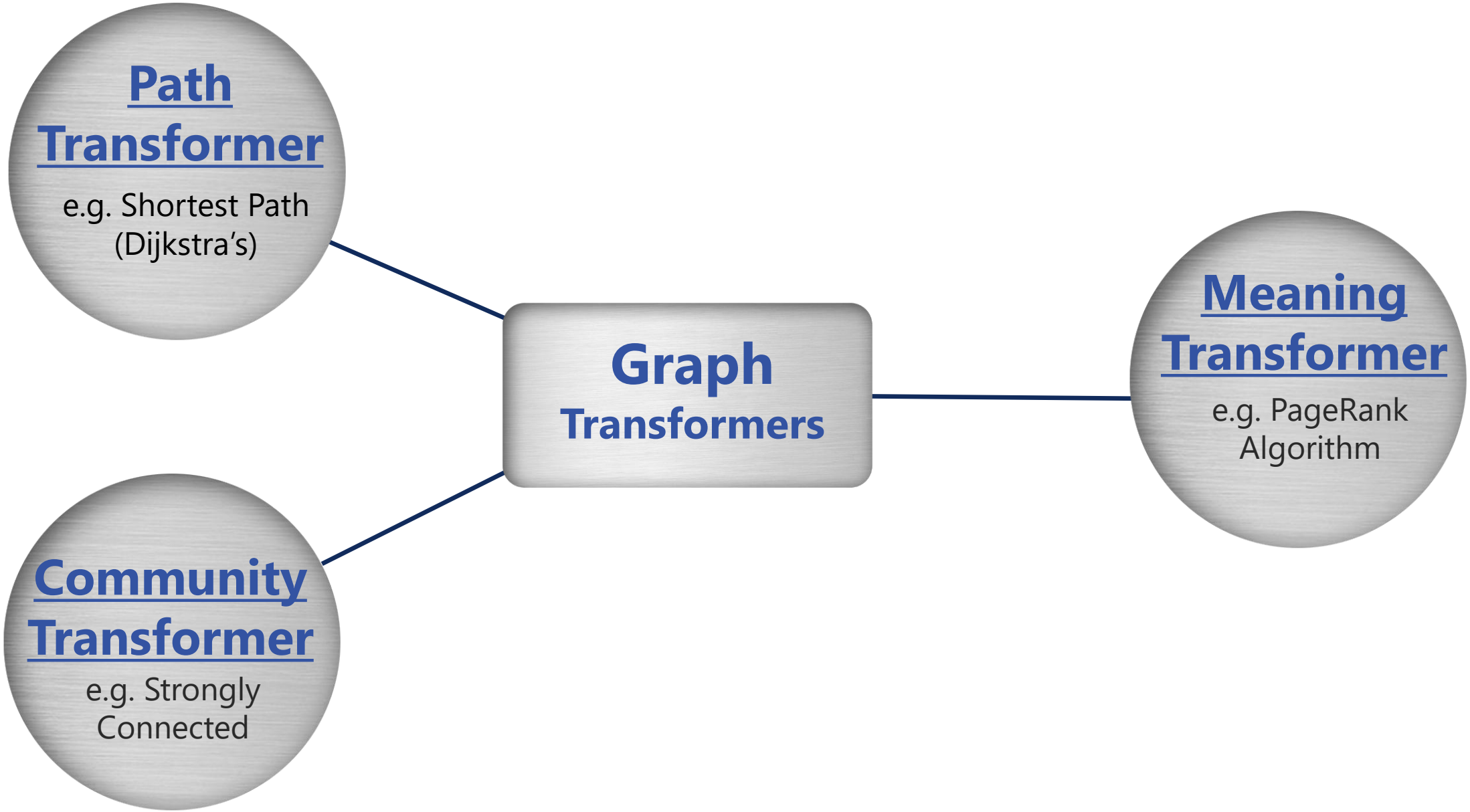
«Tight corner coming up»!

«Take it with max15 km/h»!

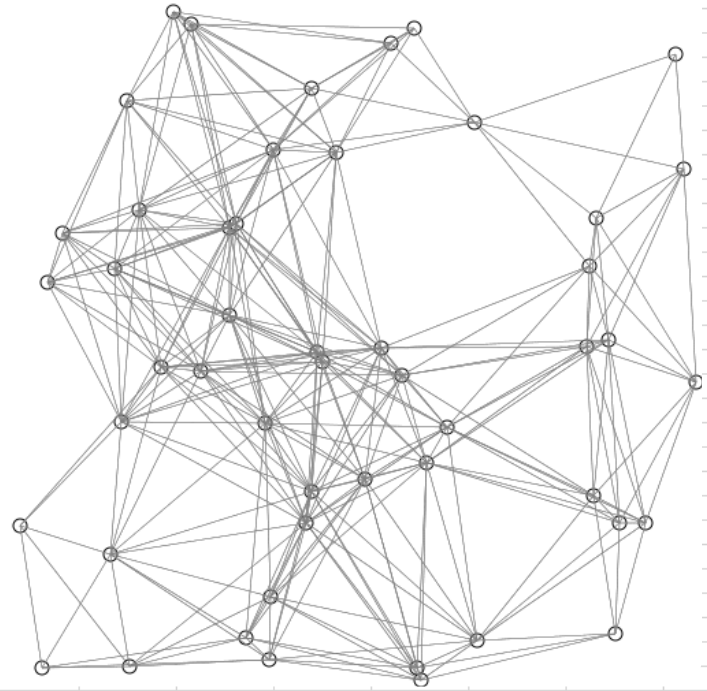
Knowledge Graph from **Start** to **Finish**



Transform a Graph into Meaning



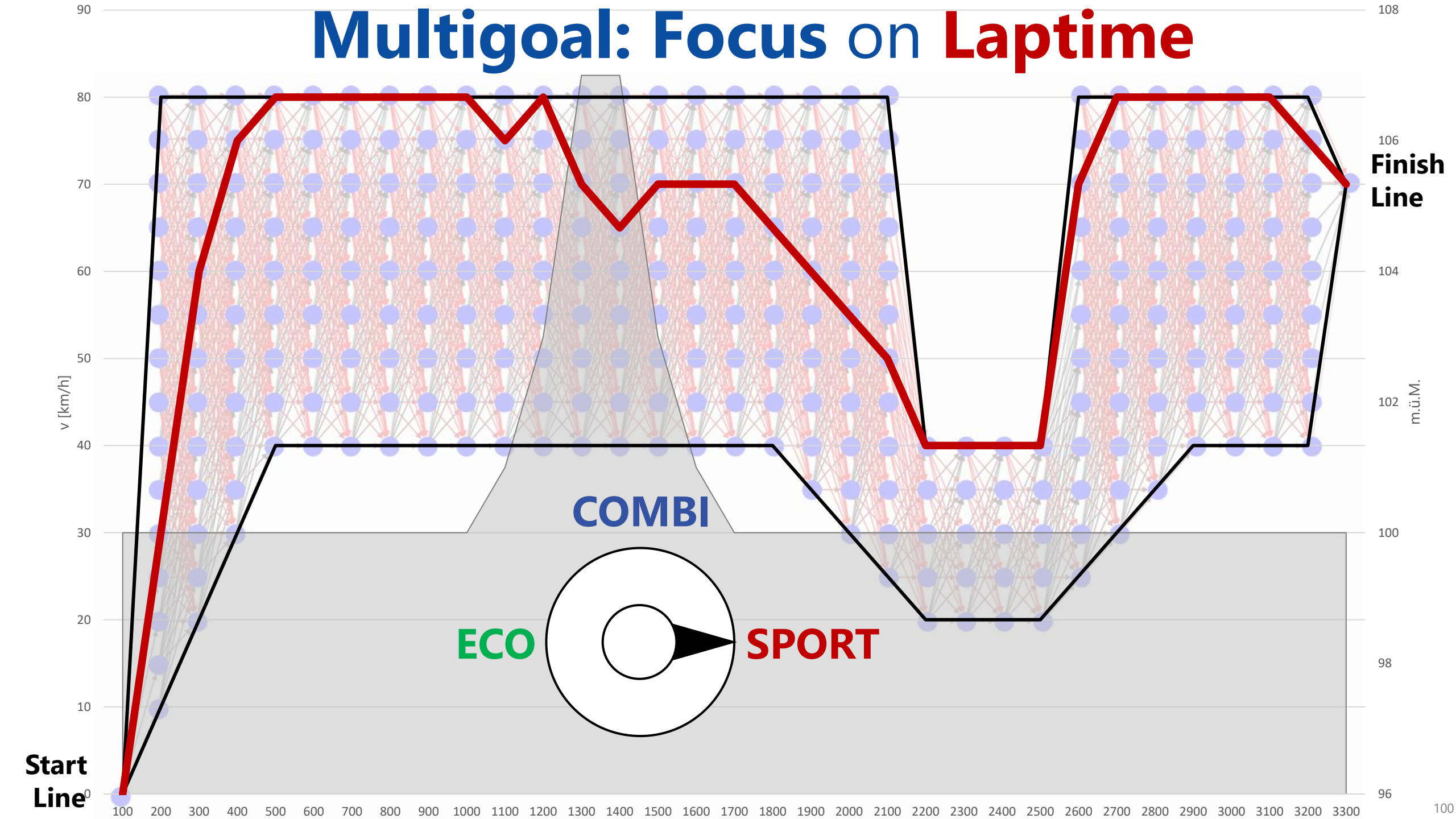
Path Transformer for our Race Problem



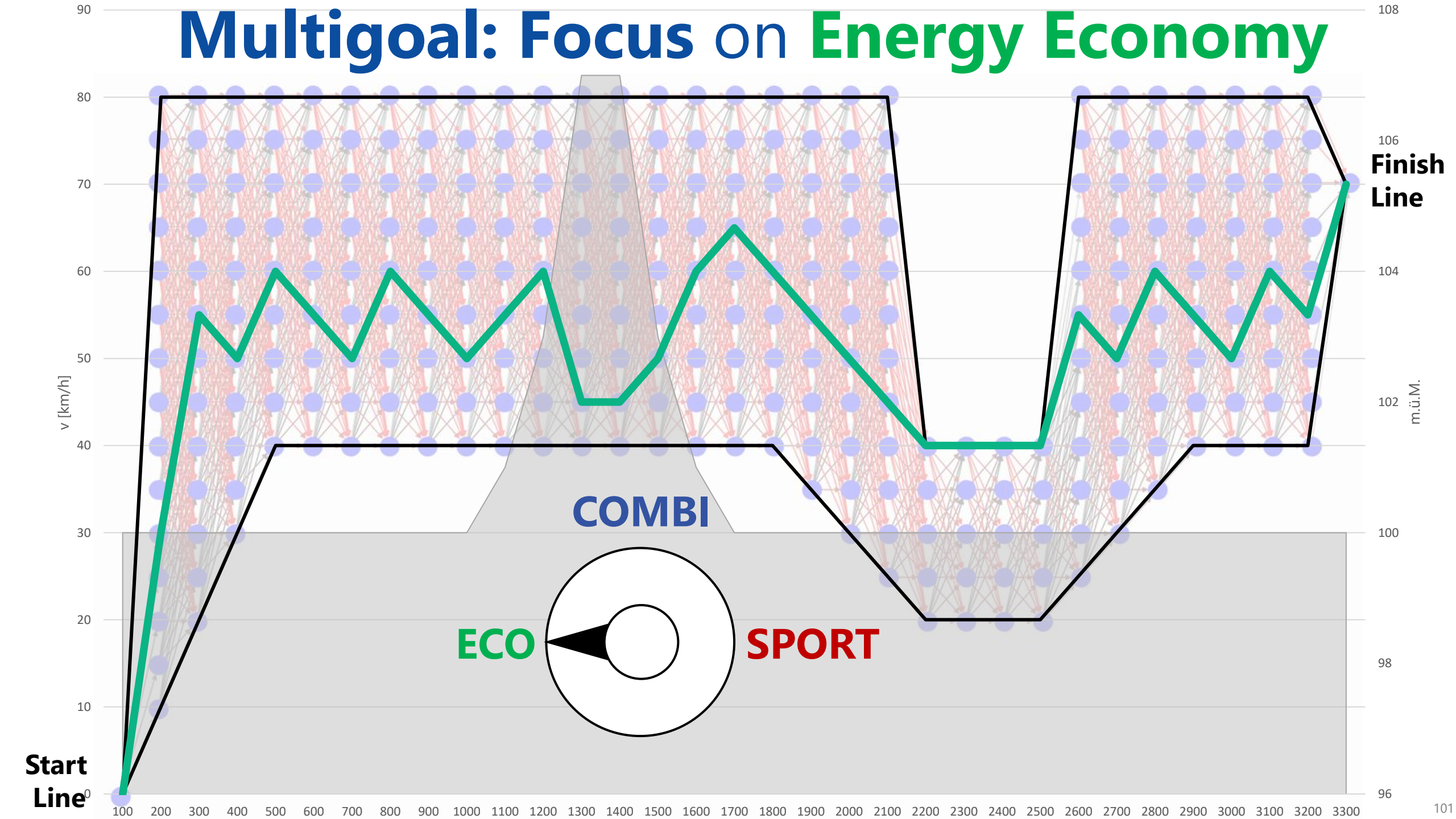
Dijkstra-Algorithm

A*-Algorithm

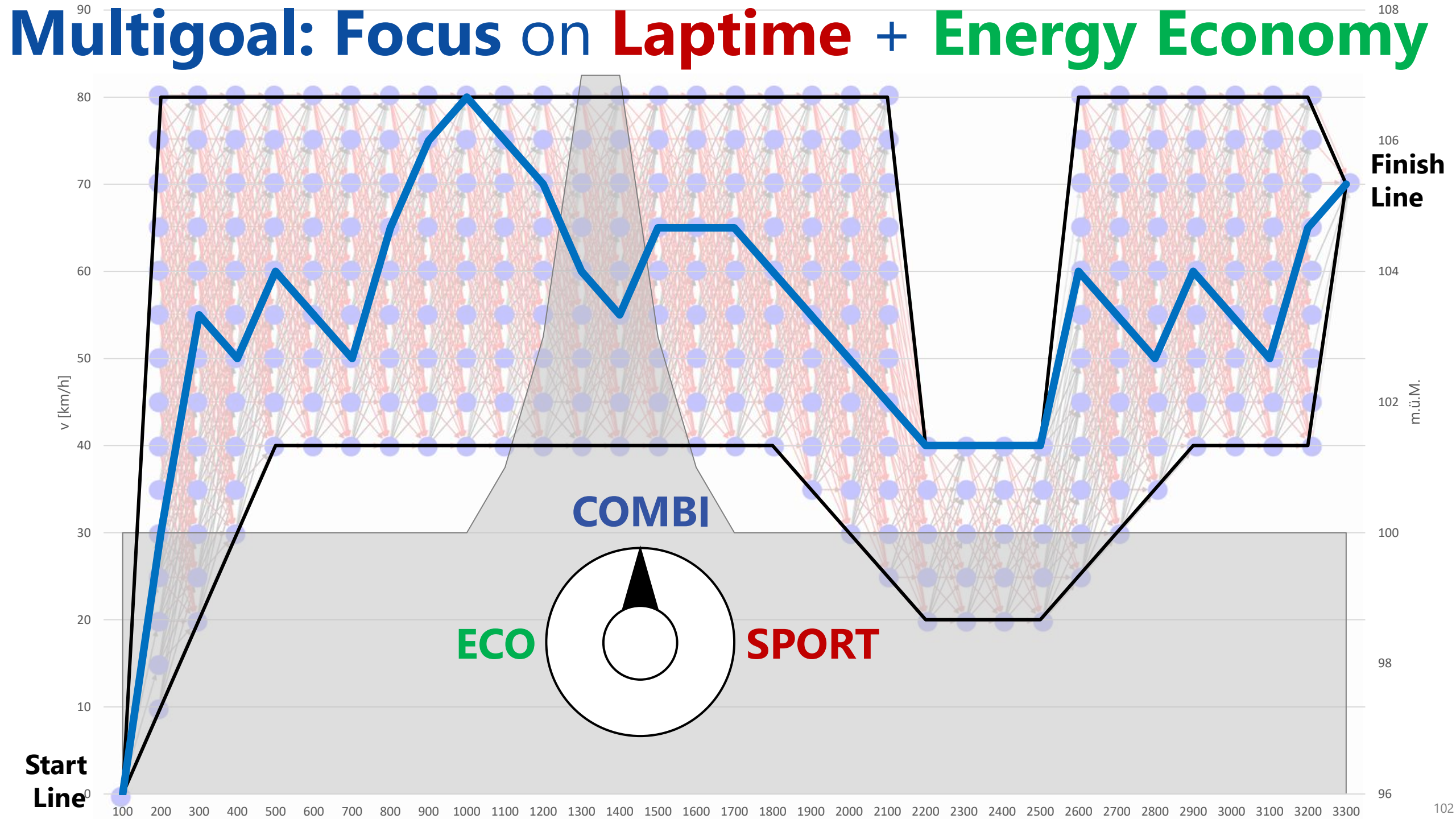
Multigoal: Focus on Laptime



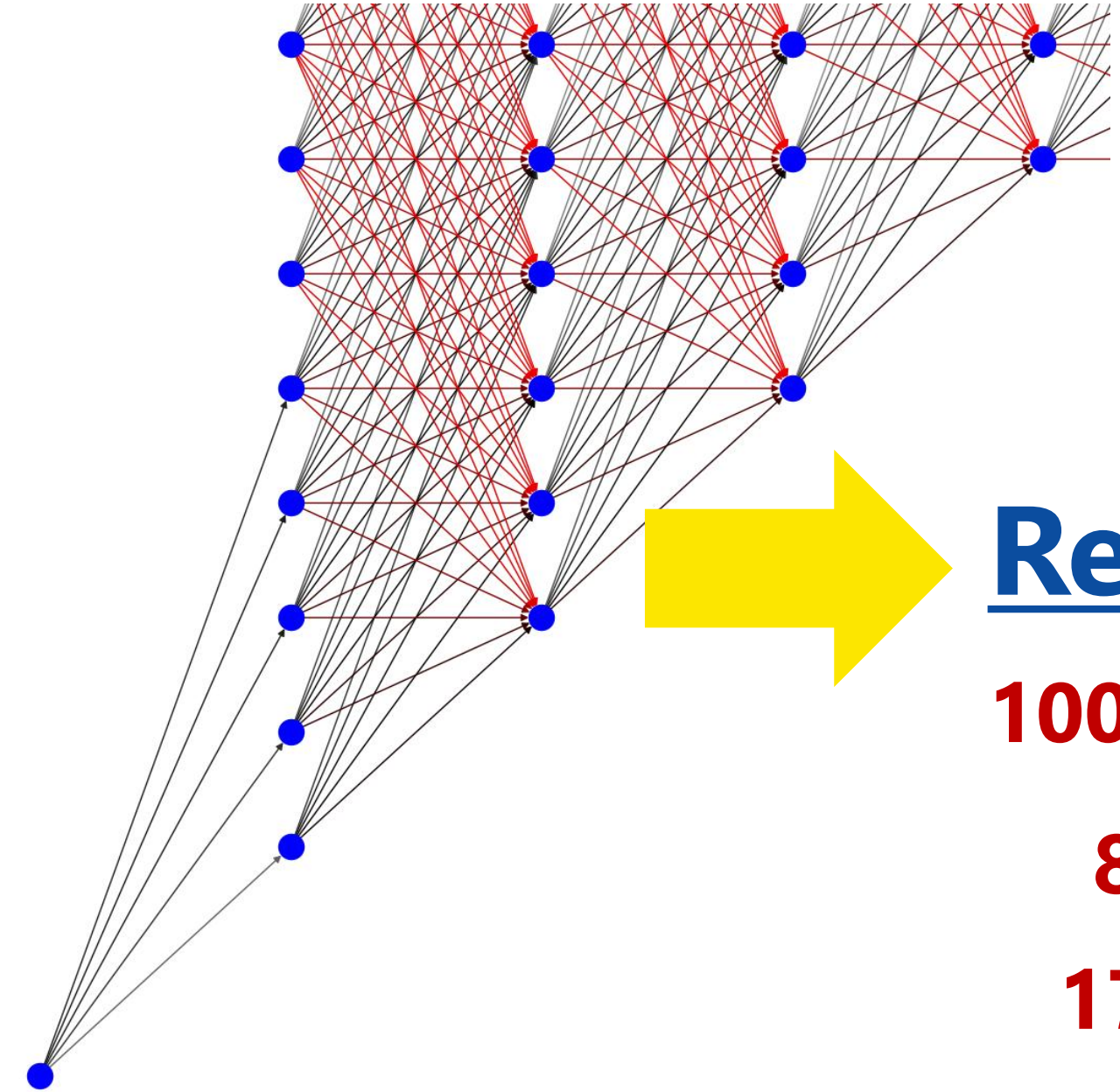
Multigoal: Focus on Energy Economy



Multigoal: Focus on Laptime + Energy Economy



Collecting **Knowledge** in the Nodes and Edges



Result:

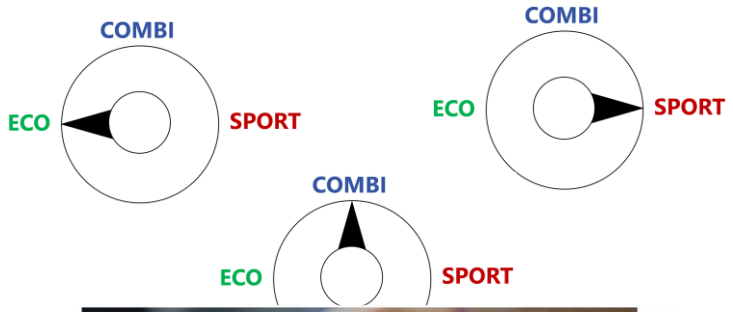
100% Accident-free Attempts

8% Shorter Lap Times

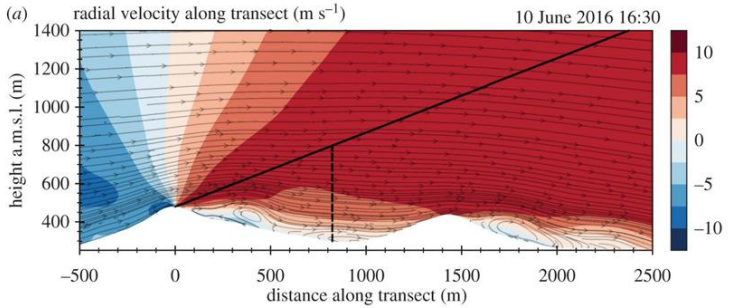
17% Higher Energy Efficiency

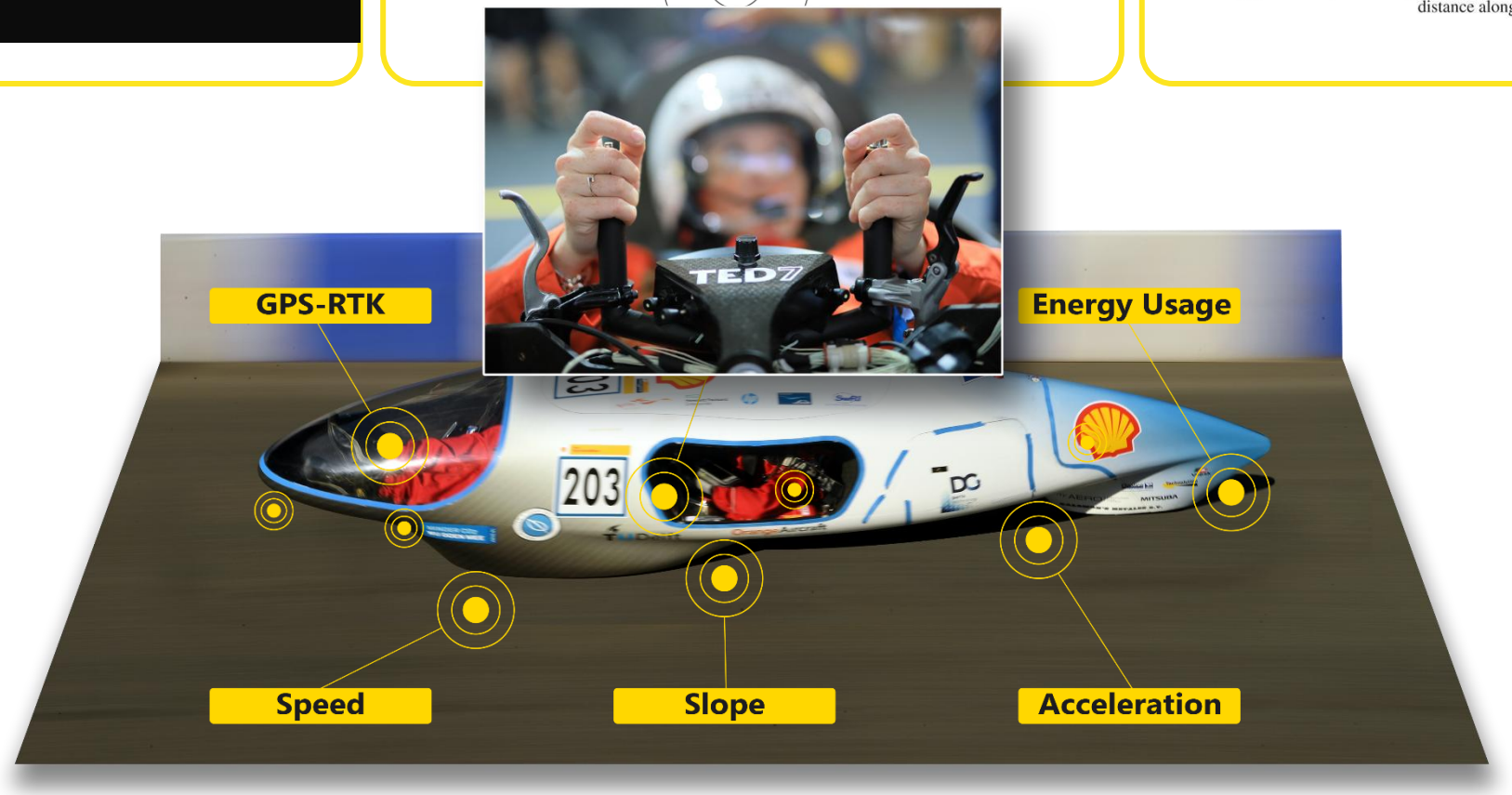
Include and adjust with **Telemetry Data**

Dynamically Changed **Driving Patterns**



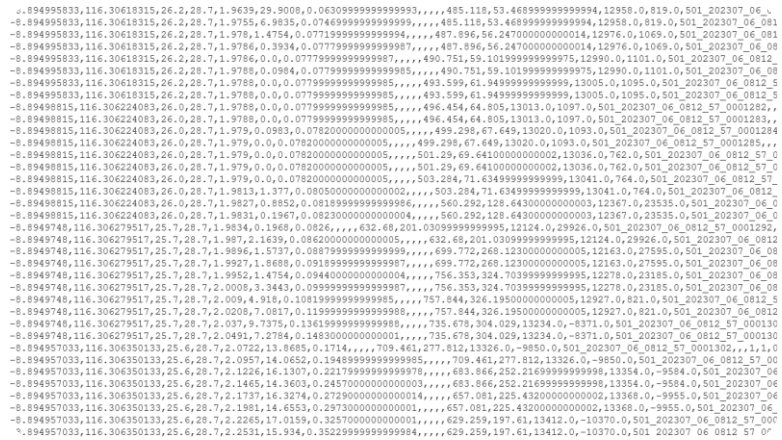
Include dynamic **Wheather Data**

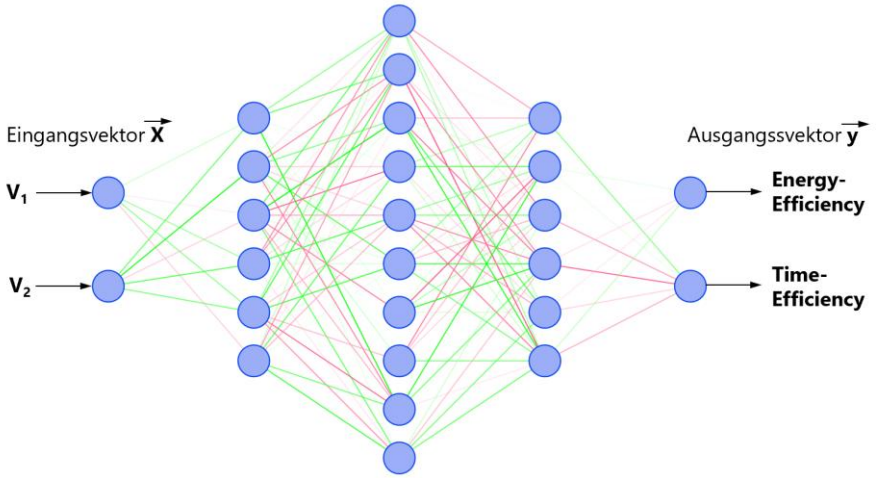


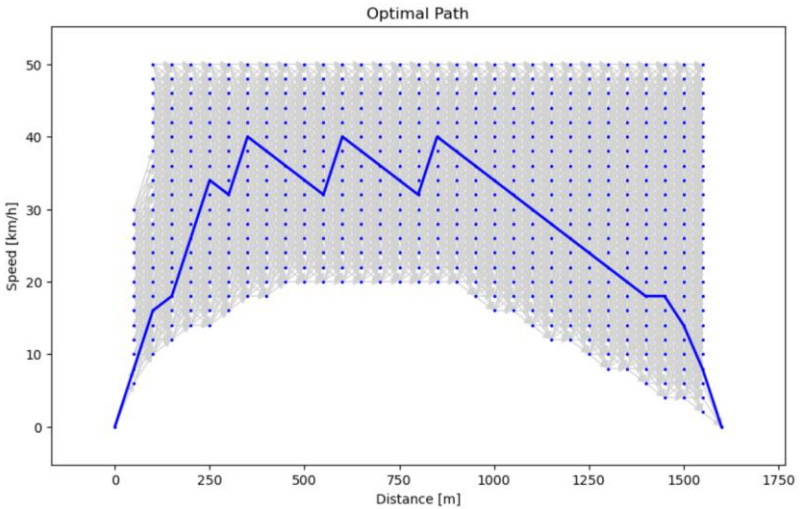


Then move to Neural Networks with DL/ML

NEW 2026







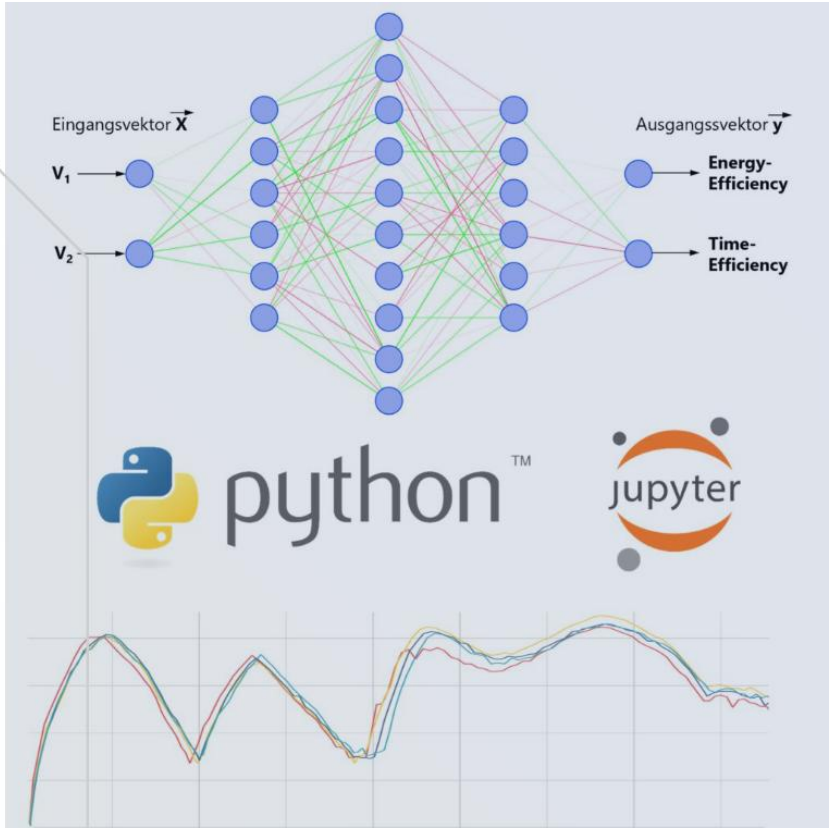
Python Sandbox **#3** Use AI with real Race Data

Programming Sandbox #3

JUPYTER NOTEBOOK AND PYTHONCODE FOR LEVEL-5 (AI-APPROACH WITH NN, DL UND ML)

Start Jupyter Notebook in your browser (Binder, ~1-2 min.) and play with Python. There is some real race data waiting for you and you experiment with a neural network, deep learning and machine learning. Save the source code locally, because when you close the browser, the environment will be gone.

START SANDBOX #3 IN YOUR BROWSER ▷

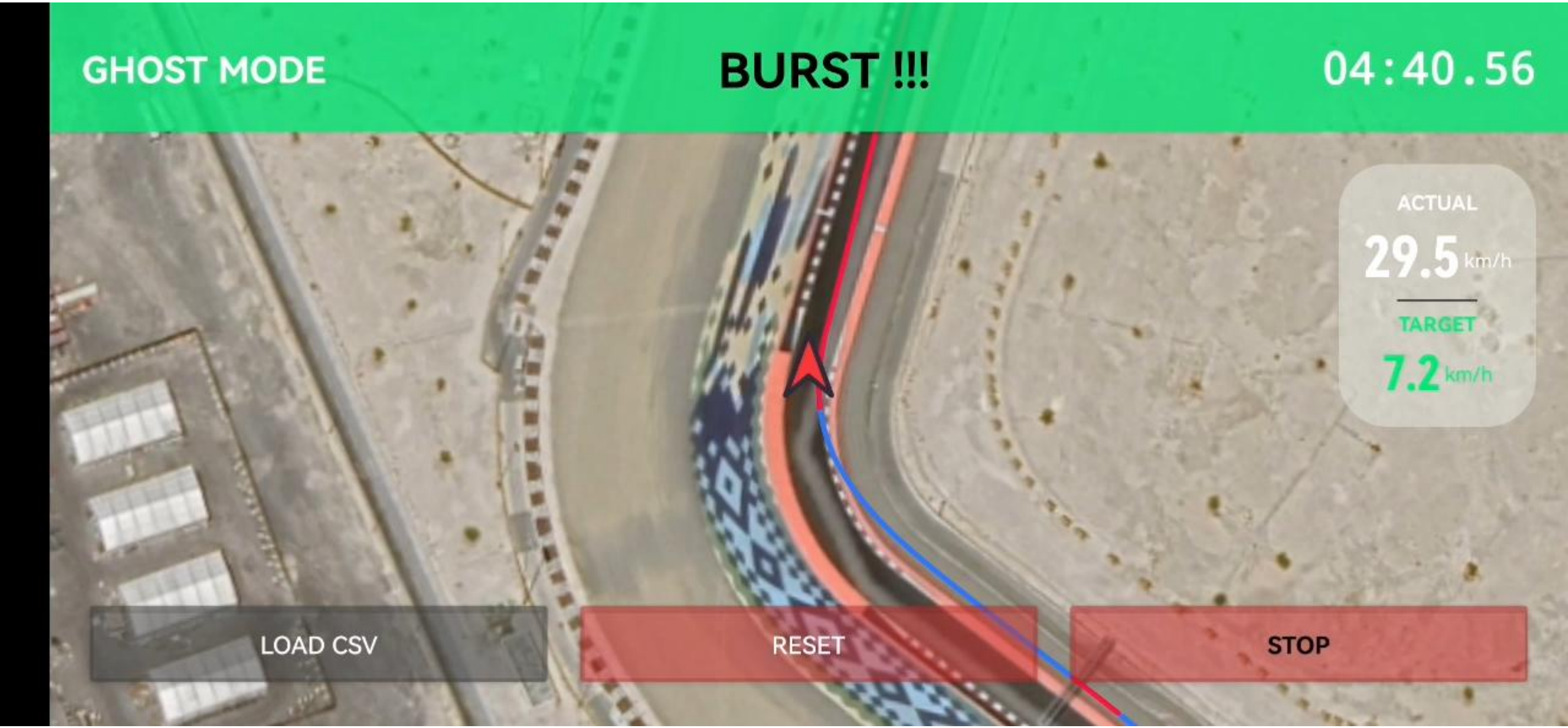


Driver Performance

Driver Support and Qualitative and Quantitative Results Improvement

Level 5

Examples of Driver Performance



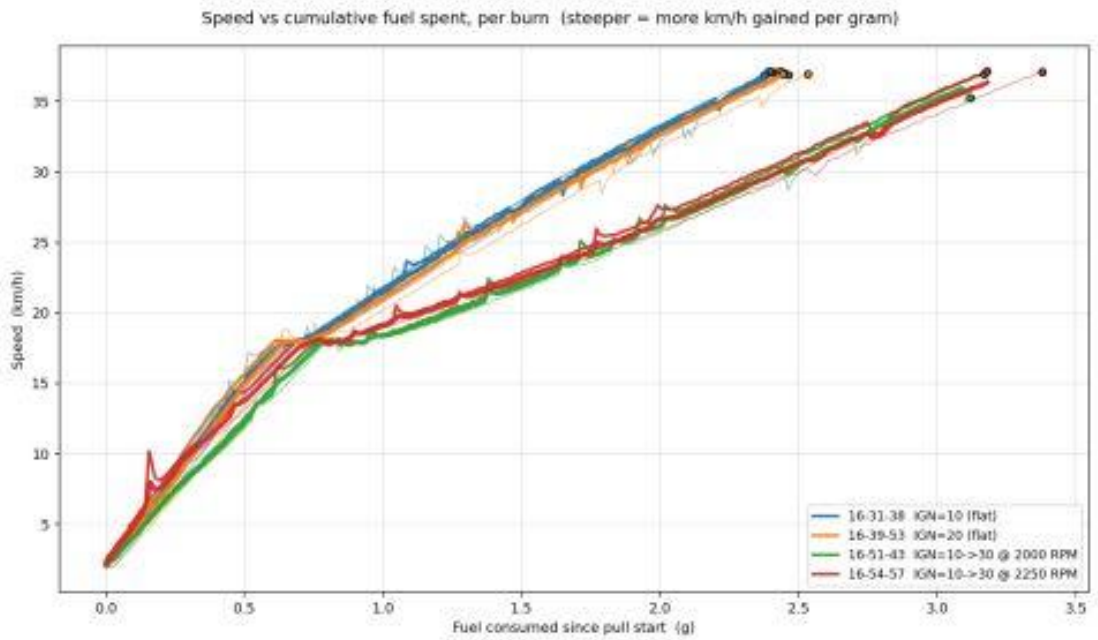
Results Improvement

- Continuously (live data)
- **Lap by Lap (close to real-time)**
- Attempt by Attempt
- Season by Season

Table 8.1 Si Pitung Results after 9 different runs with 3 different strategies at Jakarta International E-prix Circuit.

Attempt	Km/L	JM (J)	Lap-time	Improvement	Driver Feedback
Strategy 1 (S1 / No AI)					
1	549.15	3.4	25:43:003	-1.05%	Unstable and based on manual strategy.
2	527.05	3.5	25:15:434	-5.85%	No clue of where the improvements or
3	554.76	1.5	25:00:911	-0.04%	changes should be placed.
Strategy 2 (S2 / AI – Heuristic Dataset)					
4	867.1	1.2	25:12:453	+56.23%	More stable information given by the
5	912.2	1.6	25:04:628	+64.36%	AI. Helps on determining pull-point,
6	887.4	1.3	25:00:455	+59.89%	but not always right or accurate.
Strategy 3 (S3 / AI – OBC's Dataset)					
7	667.32	2.0	25:12:113	+20.24%	More unstable information in every
8	546.45	2.5	25:23:147	-1.54%	attempt compared to S2
9	647.14	2.3	25:07:235	+16.60%	

$$\text{Efficiency Gain } (\%) = \left(\frac{628 - 431}{431}\right) \times 100 \approx 45.7\%$$



Iterative Learning based on MVP Philosophy

Pivot or Continue?

Idea

Learn

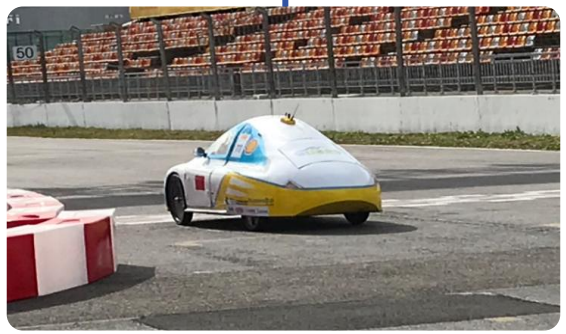
MVP

Build

Measure

Telemetry Data

Race Strategy



Learn: Quantitative Analysis of Energy Use

Strategy 1

Strategy 2

Strategy 3

Energy Use

500 1000 1500 2000 2500 3000 3500 4000

Lap Distance [m]

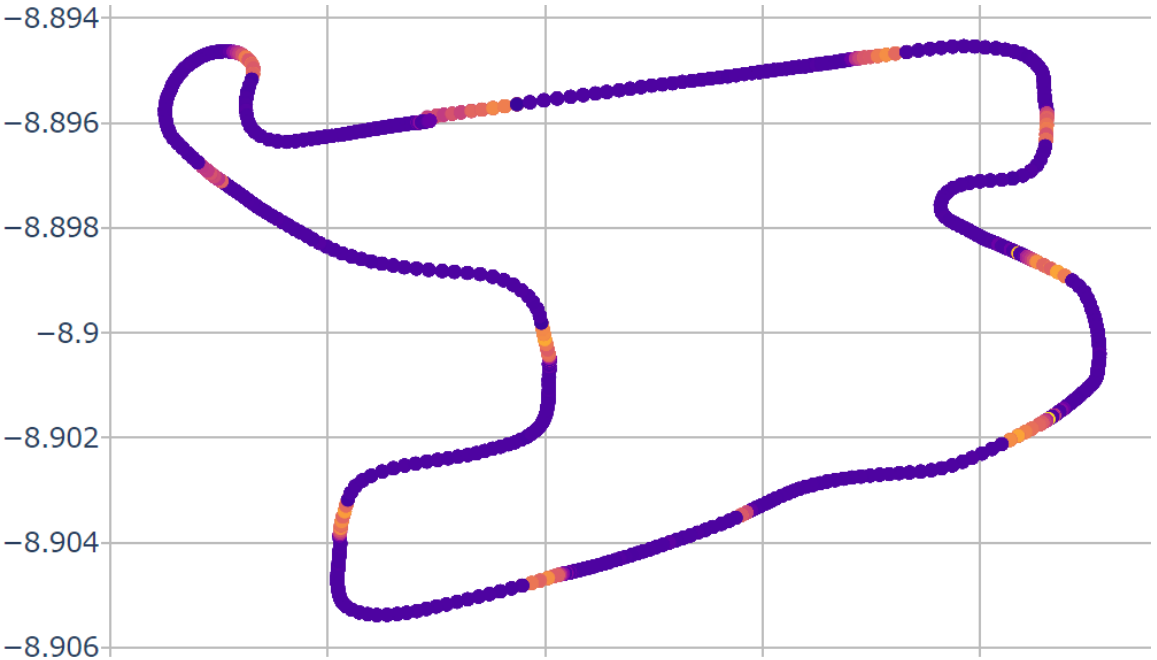
Best Strategy: 3

Distance travelled (m)

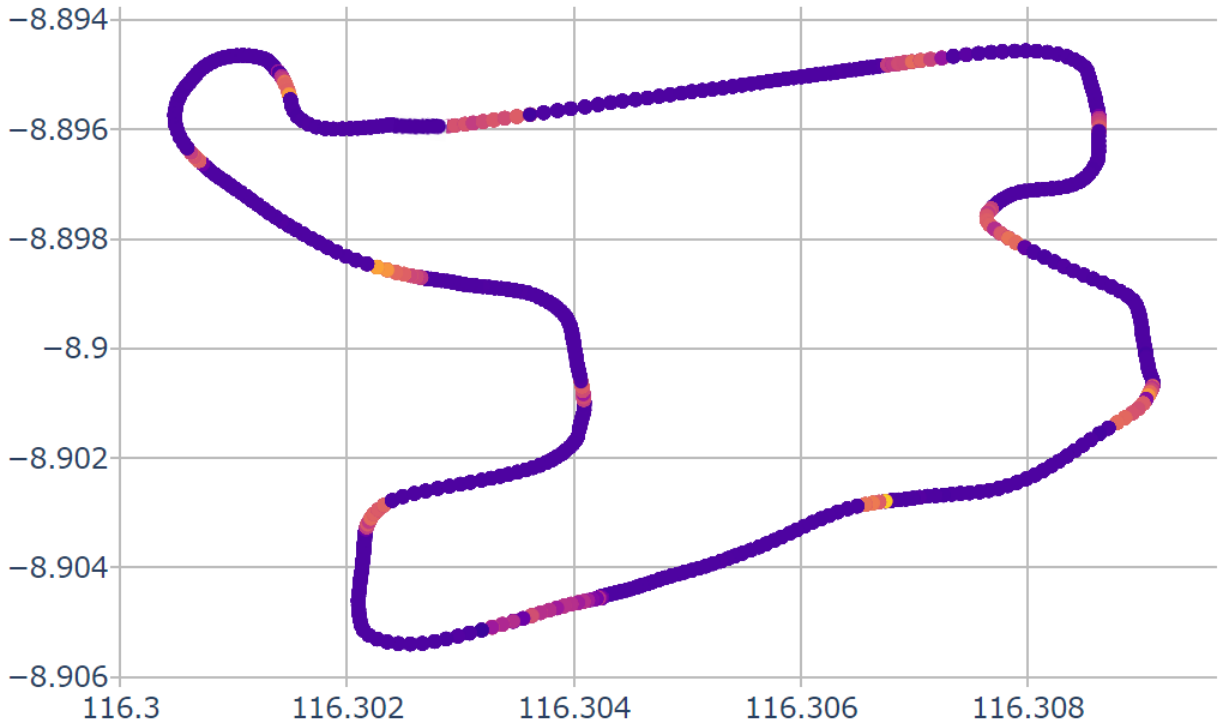
Learn: Qualitative Analysis of Driving Patterns

From Quantitative Analysis, we know:

Strategy **1**



Strategy **3** (**better**!)



But Why?

The Sky is the Limit...

Train LLMs &
Race Agents

Vehicle Health
Monitoring

Efficiency
Prediction

CollisionWarning

ADAS

Use
Oponents

Driver Training
(Simulator)

Environment
- Wind
- Rain

Gameification

etc, etc, etc

Sharing the Data on Github

cornellev / Race-GPT Public

Code | Issues | Pull requests | Actions | Projects | Security an

master | Go to file | Code

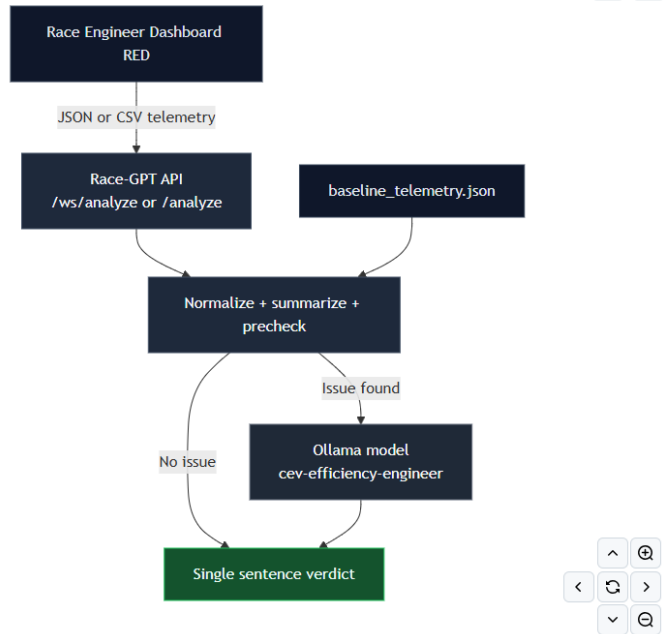
AjayParthibha single process arch — ae10079 · 3 months ago

.gitignore	Local Changes	3 months ago
LICENSE	Add MIT License to the project	3 months ago
Modelfile	merge	3 months ago
Oldfile	Significant model refinements: l...	4 months ago
README.md	More diagram updates	3 months ago
bad.csv	Initial Commit w/ recent telem	5 months ago

Race-GPT

Race-GPT is a FastAPI service that receives live telemetry from RED (Race Engineer Dashboard), compares it to a baseline profile, runs deterministic prechecks, and returns a single sentence verdict.

Architecture




```python
# python write_telemetry.py blot.csv
# ollama create cev-efficiency-engineer -f Modelfile

import argparse
import json
from pathlib import Path

from telemetry import build_summary, load_telemetry_path


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "input",
        nargs="?",
        default="mock_baseline.csv",
        help="Telemetry CSV or JSON (default: mock_baseline.csv)",
    )
    parser.add_argument("--out", default="baseline_telemetry.json")
    args = parser.parse_args()

    df = load_telemetry_path(Path(args.input))
    summary = build_summary(df)

    with open(args.out, "w") as f:
        json.dump(summary, f, indent=2)

    print(f"Baseline written to {args.out}")


if __name__ == "__main__":
    main()
```

Level 5

Driver Performance: Results Improvement

Level 4

Race Strategy: Data- and Modeldriven Approach

Level 3

Knowledge: Gained from Race Data

Level 2

Telemetry System: Capture, Transport and Store Race Data

Level 1

Data Strategy: to achieve the three goals