PacSim – A spiritual successor to FSSim

Alexander Phieler

ELBFLORACE
FORMULA STUDENT TEAM TU DRESDEN

About me

- Alexander Phieler (alexander.phieler@elbflorace.de)
- TU Dresden Diplom Informationssystemtechnik
- Season: Yes
  - EFR13: Head of AS
  - EFR14: Controls (and other stuff)
  - EFR15: SLAM (and other stuff)
  - EFR16: PacSim (and other stuff)
- Connect with me on LinkedIn

https://www.linkedin.com/in/alexander-phieler/

Basic motivation

- Ensure that a new functionality (and autonomous system) works
- … but how?

Hand written input data

➕ Specificity

➕ Low barrier of entry

➕ Determinism

➕ Easy to debug (no interaction with other modules, can debug step by step)

➖ Only feasible on component level

➖ Scalability

➖ Realism

Script generated input data (e.g. path planning test script)

➕ Determinism

➕ Scalability

➕ Easy to debug (no interaction with other modules)

➖ Only feasible on component level

➖ Realism

Recorded data (aka bagfiles)

➕Realism

➕(Kind of) deterministic

➕ Almost full pipeline covered

➖ Requires data (difficult for beginner teams)

➖ Variability (setup and scenario fixed)

➖ Only open-loop

Simulator (raw perception input simulated)

➕Closed-loop behavior

➕Full pipeline covered

➕Variability

➖ High barrier of entry (complex design, powerful computer with GPU required)

Simulator (model of perception)

➕Closed-loop behavior

➕(Almost) full pipeline covered

➕Variability

➖ Realism (simplifications / assumptions about perception system)

Real life testing

➕Closed-loop behavior

➕100% code coverage

➕Full realism

➖ High barrier of entry (requires working vehicle, testing location and crew)

➖ Dependence on external factors (working vehicle, weather)

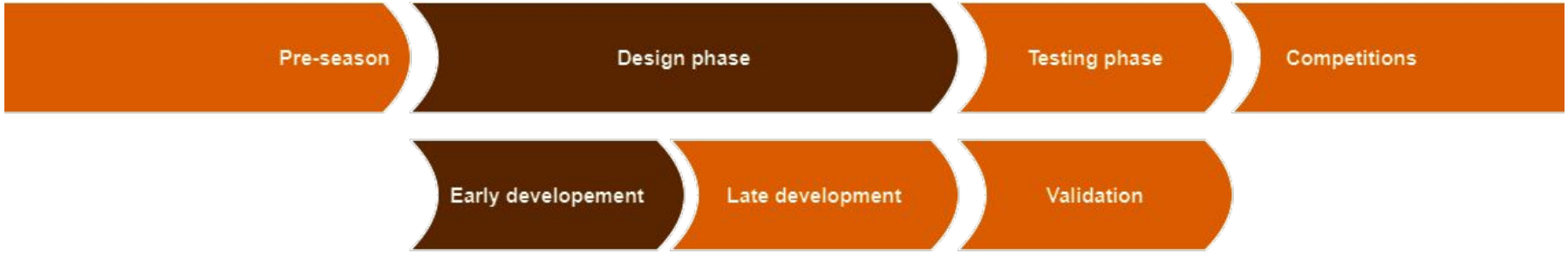
➖ Low throughput

➖ Variability (Sensor setup and environment constraints)

Conclusion

- There is no perfect solution
- Real life testing is extremely valuable but scarce, should not be “wasted”
  - Be well prepared
- Combine different approaches based on development progress

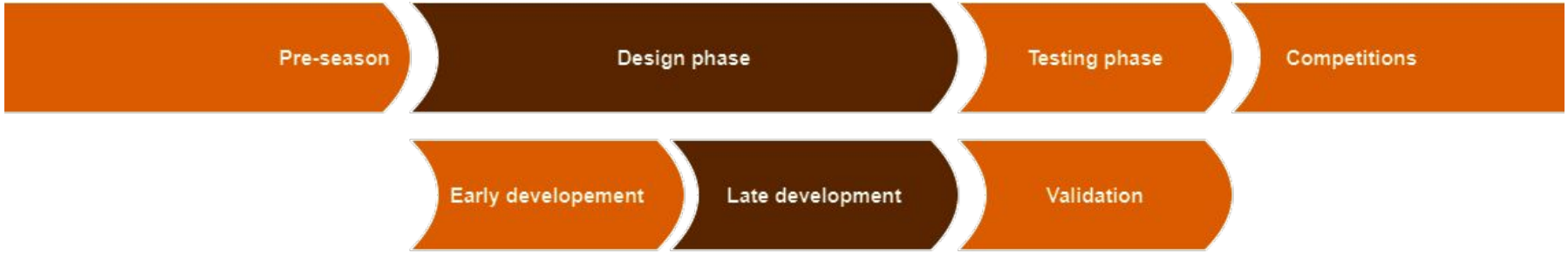
Early implementation



- Make it easy to implement and debug
- Focus on determinism and the essentials
- Open loop behavior

- **Hand written data, recorded data, simulator (easy scenarios)**

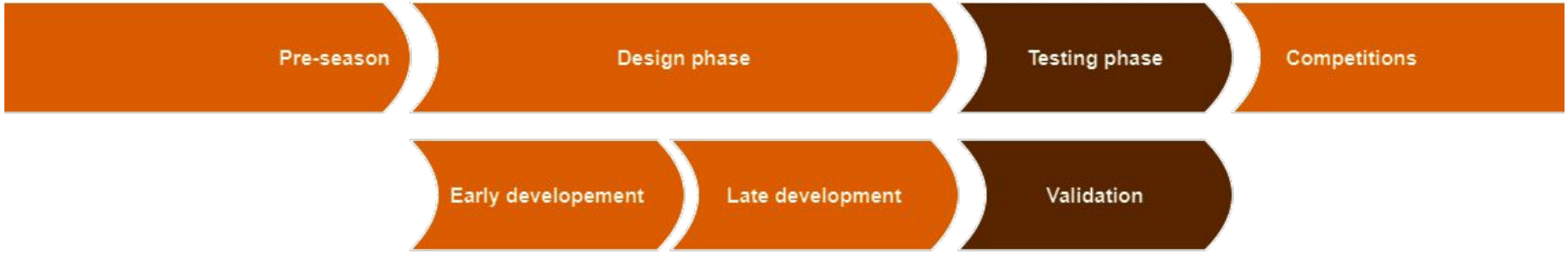
Later implementation



- Increase difficulty and realism
- Start closing the loop

- **Recorded data, simulator (hard scenarios)**

Validation



- Everything should work now, make sure you really didn’t miss something
- **Recorded data, simulator (hard scenarios), real life testing, script generated data**

Simulators

- We see that simulations are a powerful tool (useful in every stage)
  - Only way to test closed-loop performance outside of the car
  - High variability (easy-hard, all variations of scenarios)
- What are the options out there?

Simulators (without raw perception data)

- FSSIM
  - Developed by AMZ for 2018 season (I think)
  - What we used since release in 2019

Simulators (with raw perception data, FS specific)

- FS Online simulator (Formula-Student-Driverlerss-Simulator)
  - Originally developed by Delft and MIT Driverless for 2020 online compeitions
  - Modified version of Airsim (UE4 based robotics simulator tailored for drones)
- eufs_sim
  - Developed by Edinburgh Formula Student team
  - Gazebo based

Simulators (with raw perception data, others)

- Carla (UE4 renderer)
- Nvidia Isaac sim
  - Even more constrained hardware (nvidia only)
- Proprietary
  - IPG Carmaker (Physical sensor models)
  - Siemens Prescan (UE5 renderer, Physics based lidar)

My opinion on simulators

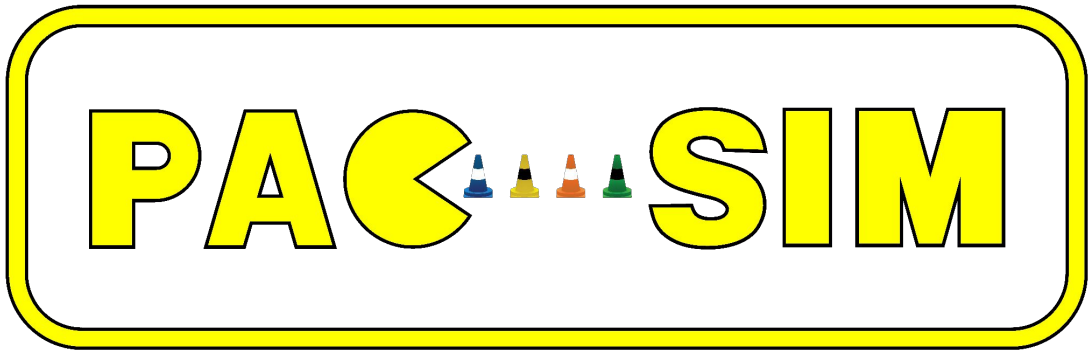
- I am a big fan of the FSSIM style simulators for FS
- Open-source, runs on pretty much anything → Everyone can work with it
- Perception sensor simulation only makes sense as an addition i.m.o.
  - Limited realism, high barrier of entry (implementation, powerful computer required)
  - Using recorded data is the overall better way to go for most teams

Problems with FSSIM

- First of all: It is a great project which helped us immensely!
- Dependence on Gazebo
  - Gazebo classic is deprecated
  - Unnecessary complexity
- ROS1 only
- No active development
  - Original release for Ubuntu 16.04 / ROS1 Kinetic
  - We ran our own modified version up to Ubuntu 20.04 / ROS1 Noetic

PacSim: Do it even better

- With our ROS2 transition sticking to FSSIM wasn't feasible
- Take our learnings from FSSIM and write a new simulator
- In August 2023, on the FSG campsite PacSim (Planning and Controls simulator) was born



PacSim: Project goals (1)

- Standalone simulator project which is comparable to FSSIM in it’s scope
  - Simulate all relevant sensors in a vehicle except camera/lidar
  - Provide a model of perception sensors
  - Competition logic such as lap counting or off course detection
  - Provide tools such as a track editor

PacSim: Project goals (2)

- Simulation “framekwork” / backend
  - Provide same functionalities as standalone mode
  - Run in tandem with other tools such as Prescan to simulate perception sensors
- End goal: Common simulator for all usecases

Similarities with FSSIM

- Runs on any machine
- Don’t simulate raw perception data
- Also has the features FSSIM has (superset)

Differences with FSSIM

- Minimal dependencies
  - Right now: ROS 2 stuff, yaml-cpp, Eigen
  - Better maintenance, little dependence on other projects
- Superset of features
  - Simulate more sensors of the vehicle (IMU, Wheelspeed, GNSS)
    - Velocity estimation can be tested in sim
    - Better system coverage and Sim2Real
- Active development
  - We plan to submit our improvements to upstream
  - We encourage you to submit own features / ideas

How to use

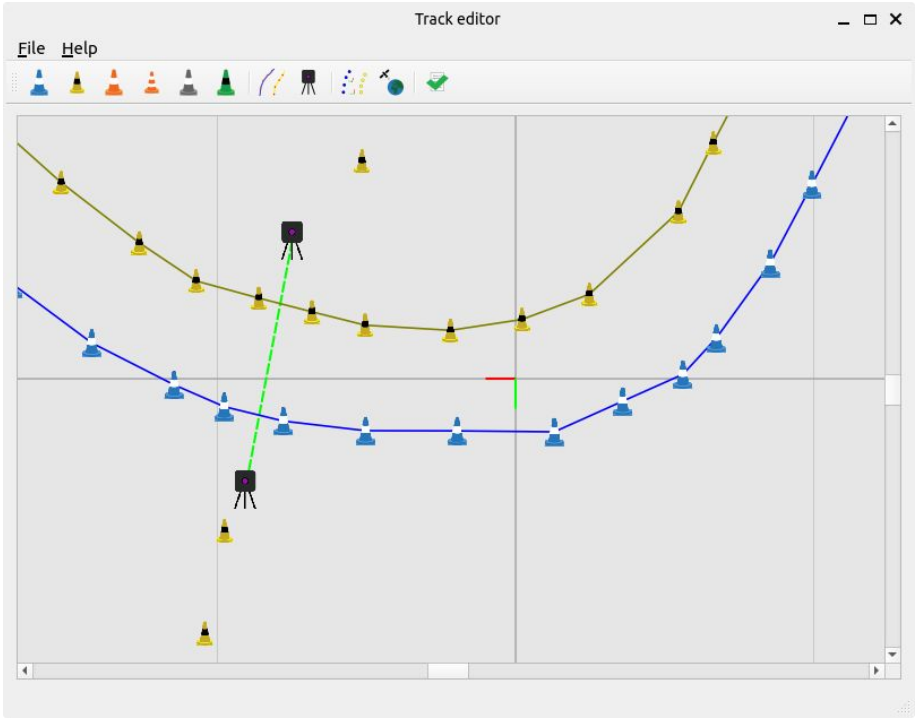
- git pull / create fork
- Implement message converter node
  - Match PacSim messages with your system
- Adjust configs to your use case
- (Replace car 3d model with your own)
- (Implement own vehicle model)
  - Default: Nonlinear dynamic bicycle model, should be sufficient for fresh teams
- Implement new feature and submit upstream PR

Current features (Sensors)

- All sensors can be specified by some rate, dead time (delay), noise (and more specific properties)
  - Wheelspeed / Wheel torque
  - IMU
  - Perception (cones)
  - …

Current features (Track editor)

- We provide an editor which allows to create or modify tracks in our file format



Current features (Competition logic)

- Lap count / lap time tracking
- Cone hit detection
- Off course detection
- Create report at end of run
  - Can be evaluated in CI pipeline

Use cases

- Test planning and controls pipeline (surprise)
  - Everything from state estimation (... path planning ...) to controls
- Use ground truth values to test subset of pipeline:
  - e.g. ground truth map+pose+velocity to test everything from path planning on
  - (e.g. ground truth localization + track boundaries to test trajectory planning + controls)

Short term plans

- As usual in the FS, I had to take some some “shortcuts” to have a first version to publish with this presentation
- Focus should be on stability and quality
  - Only 1 new feature: GNSS
  - Some parts should be refactored
  - General documentation
  - Doxygen code documentation
  - CI build test
  - Any further polish and bugfixing
- Investigate ROS1?
  - What is the demand? Please give feedback if interested
  - ROS1 bridge vs ROS1 implementation (port)?

Long term plans

- Improve perception sensor modelling
  - Realistic false positives / false negatives
- Variable grip (grip map)
- Ground truth path from track file
- External clock control
  - For debugging (e.g. pause after every perception output)
  - When running with other simulators (e.g. rendering camera images)
- Maybe: Non-planar tracks (3d physics)
  - Depends on scope of required changes and manpower

- Own ideas? Create issue (or tell me otherwise)

Acknowledgment

- Alexander Phieler (Me): Most of the stuff
- Niklas Leukroth: Track and Config file parser
- Sergio Antuna: Artwork
- Tim Hanel: 3d car model integration

- Maybe you soon?

Thank you for your attention!

- Discussions / Opinions / Suggestions / Criticism welcome

https://github.com/PacSim/pacsim

https://www.linkedin.com/in/alexander-phieler/

Thank you for your attention!

- Discussions / Opinions / Suggestions / Criticism welcome

Feedback page