GPUDrive: Multi-Agent Driving Simulator

Presentation By: Lesli Perez, Edwin Serna

TABLE OF CONTENTS

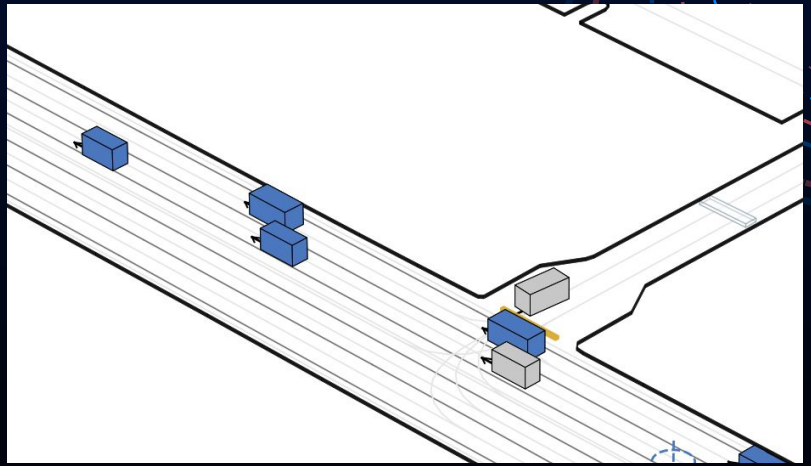
01. Research Paper Overview

A brief overview of the paper “GPUDrive: Data-driven, multi-agent driving simulation at 1 million FPS”

Introduction

What is GPUDrive?

GPUDrive is a fast simulator that combines real driving data with large-scale simulations. Its purpose is to help researchers develop multi-agent algorithms faster and more efficiently, reducing experiment time and allowing academic labs to scale on a limited computing budget.



Introduction

GPUDrive Capabilities

- Runs at 1M+ simulation steps per second
- Lightweight memory footprint → thousands of environments, hundreds of agents
- Useful for studying human driving, cycling, and pedestrian behavior
- Provides Gymnasium environments (Torch & JAX)
- Includes open-sourced policy-gradient training loops and baseline agents achieving ~95% goal success

GPUDrive VS Other Simulators

Simulator	Multi-agent	GPU-Accel	Sensor Sim	Expert Data	Sim-agents	Routes / Goals
TORCS (Wymann et al., 2000)			✓		✓	-
GTA V (Martinez et al., 2017)			✓			-
CARLA (Dosovitskiy et al., 2017)			✓		✓	Waypoints
Highway-env (Leurent et al., 2018)						-
Sim4CV (Müller et al., 2018)			✓			Directions
SUMMIT (Cai et al., 2020)	✓ ($\geq$ 400)		✓		✓	-
MACAD (Palanisamy, 2020)	✓		✓		✓	Goal point
SMARTS (Zhou et al., 2021)	✓					Waypoints
MADRaS (Santara et al., 2021)	✓ ($\geq$ 10)		✓			Goal point
DriverGym (Kothari et al., 2021)				✓	✓	-
VISTA (Amini et al., 2022)	✓		✓	✓		-
nuPlan (Caesar et al., 2021)			✓	✓	✓	Waypoints
Nocturne (Vinitsky et al., 2022)	✓ ($\geq$ 128)			✓	✓	Goal point
MetaDrive (Li et al., 2022)	✓		✓	✓	✓	-
InterSim (Sun et al., 2022)	✓			✓	✓	Goal point
TorchDriveSim (Ścibior et al., 2021)	✓	✓			✓	-
BITS (Xu et al., 2023)	✓			✓	✓	Goal point
Waymax (Gulino et al., 2024)	✓ ($\geq$ 128)	✓		✓	✓	Waypoints
GPUDrive (ours)	✓ ($\geq$ 128)	✓	✓	✓	✓	Goal point

02 Methodology & Source Code

Review of the methodology and given source code.

IPPO

- Agents are trained using Independent Proximal Policy Optimization (IPPO), a multi-agent reinforcement learning algorithm where each agent learns its own policy independently without communication with other agents.
- The paper uses a faster version of the trained agents because they used the 100K Waymo Dataset.

Simulation Engine

Challenges

- Road geometry stored as dense polylines = LOTS of points = high memory use and redundant agent observations
- Large numbers of agents + road objects = collision checking becoming a bottleneck
- Scene variability: each world must allocate memory based on the maximum agent count across **all** worlds = inefficient and slows performance

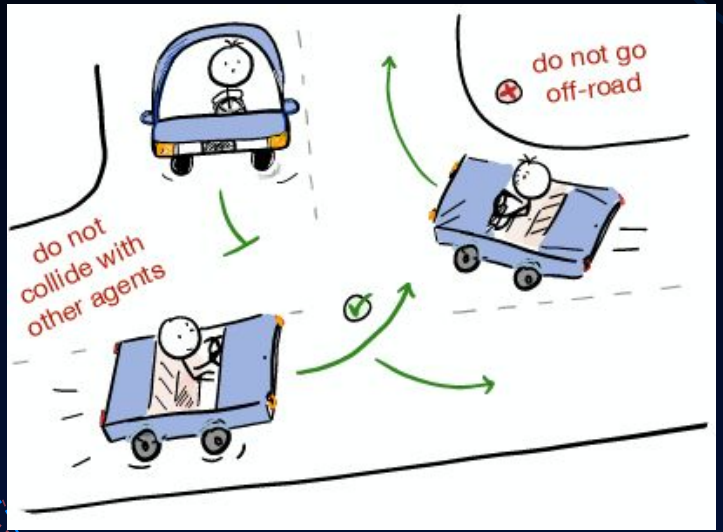
Solutions

- **Polyline Decimation:** Removes unnecessary points on straight road segments, up to **15× fewer points** = lower memory + faster step times
- **Bounding Volume Hierarchy (BVH):** Filters out non-colliding object pairs = reduces collision-check workload
- **Dynamic Memory Allocation**: Only allocates memory for the actual number of agents in each world = avoids waste, improves scalability

Simulation Features

Rewards

- Agents must reach target (the last recorded vehicle position)
- By default agents get a +1 if they reach it, otherwise 0
- Negative points for collisions or driving off-road



Termination Conditions

- Episodes end when an agent reaches target or collides.
- Collisions include driving through pedestrian areas

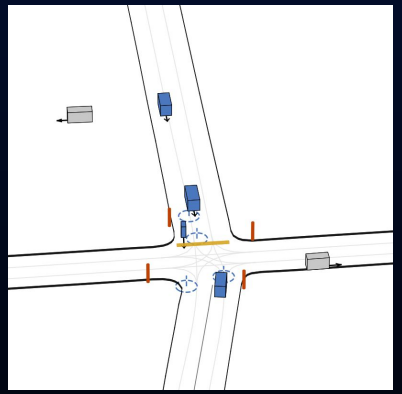
Simulator

Simulator State

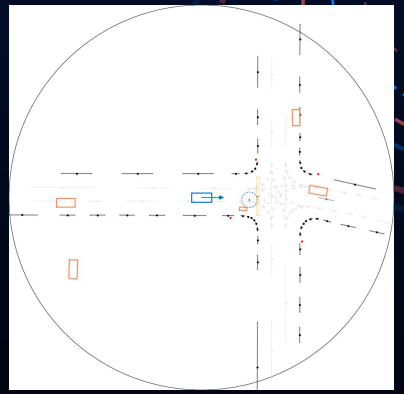
- Bird's-eyeview of dataset scenarios with boxes for controlled agents and circles for their goal destination.

Agent View

- Top-down view that focus on a single controlled agent, shows road points within set radius.

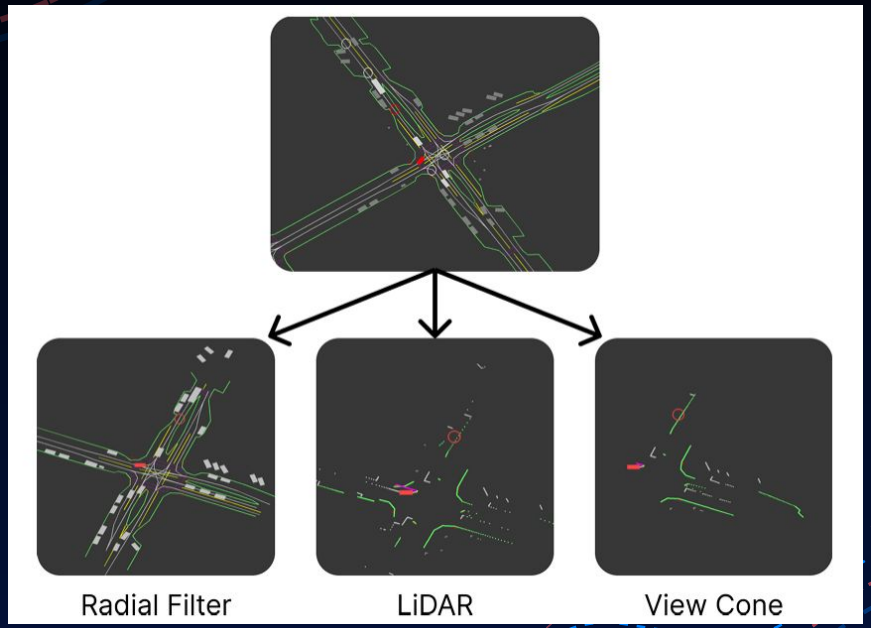


Simulator state



Agent view

Simulation



Observation Spaces

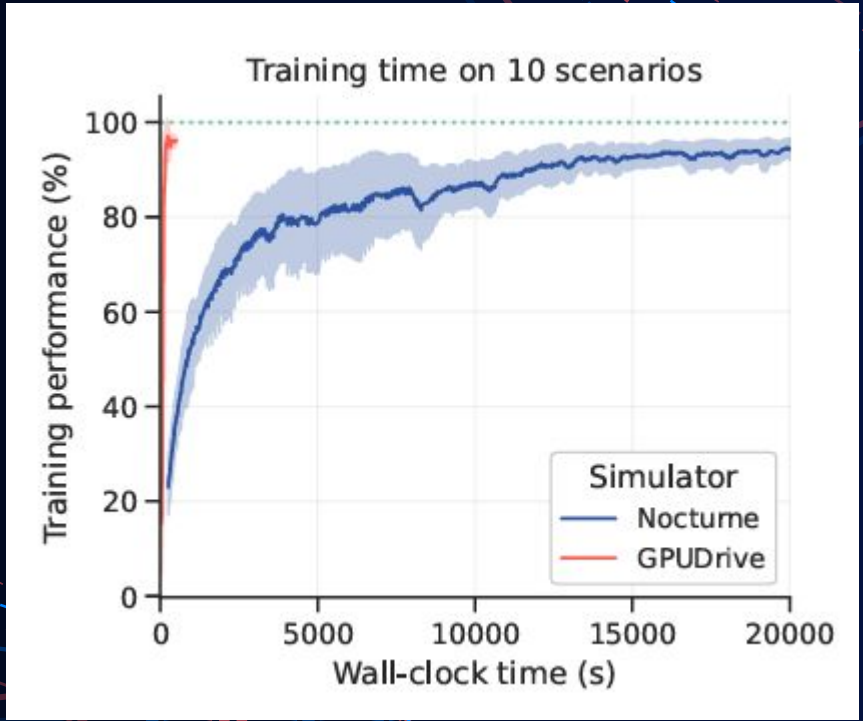
- All agents & objects within a fixed radius are observable
- GPU-accelerated LIDAR: agent view
- GPU-accelerated LIDAR: human-like view

Dataset

- Compatible with most driving datasets
- Uses Waymo Open Motion Dataset: 100K multi-agent traffic scenarios

Simulator Performance

- Comparisons were done with closely replicated environmental and experimental conditions.
- Compared to Nocturne (another similar driving simulator) GPUDrive's training performance turns hours into a few minutes of training time.
- Agents trained using GPUDrive achieve a 95% success rate in a few hours!



03 Our Simulation

Experiments using GPUDrive.

Goals & Challenges

Goals

- Run the simulator successfully.
- Train and test agents using varying amounts of worlds.
- Measure resource trade-offs by evaluating the impact of parallelism on performance.
- Compare CPU vs GPU performance.

Challenges

- Issues: Defaulted to CPU instead of GPU, VSCode kernel, dependencies, cuda conflicts.
- Unresolved tickets on the GitHub.
- Lack of detailed documentation, and not enough resources to help us, which is expected from a novel simulator.

GPU Specifications

- The paper used: NVIDIA RTX 4080 and A100
- We used: NVIDIA 5070

Code Setup

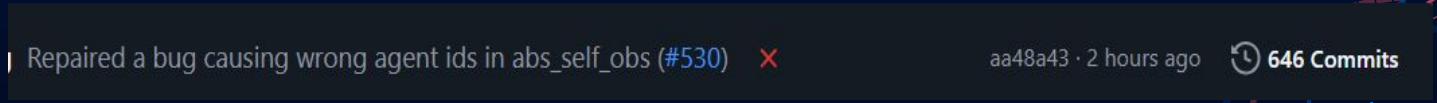
- Followed the README setup and debugged the tutorials until they ran successfully (some tutorials still fail due to unresolved source code issues).
- Created a Python script to initialize our training environment.
- Downloaded and prepared the Waymo Mini Dataset.
- Modified eval_config.yaml to configure the virtual environment using the Waymo mini dataset.
- Trained an agent using IPPO.
- Tested agent using varying amount of worlds and on GPU and CPU.
- Generated graphs to visualize performance.

04 Results

Final results and explanations.

Setup Results

- Bug fixes were being done as we worked on it.



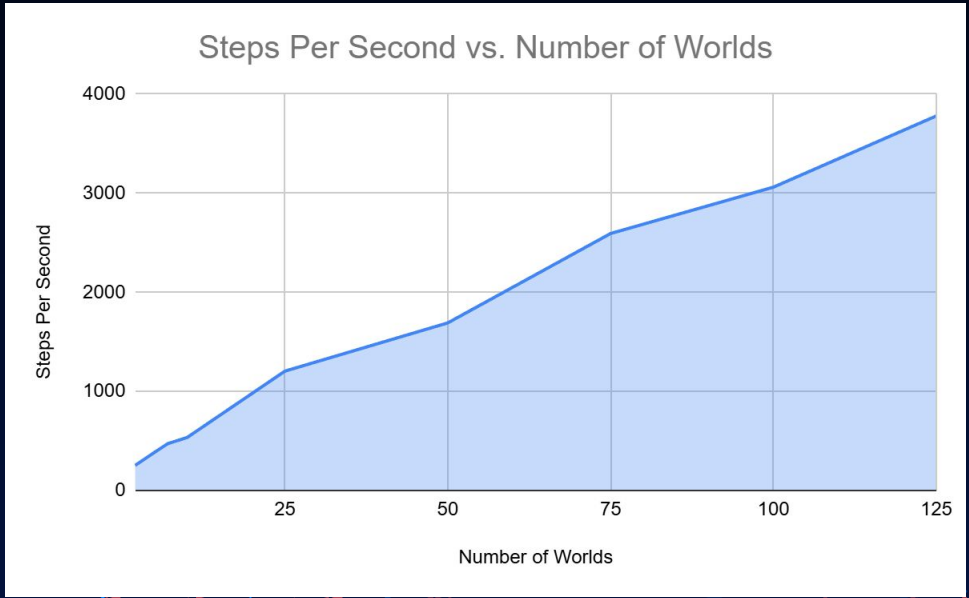
- The README file details exactly what to do for Linux/macOS, but only has one line for Windows.
- Debugging limitations on Windows + CMake + CUDA not mentioned.
- Able to work on CPU eventually on WSL, but GPU never worked.
- Setups we tried that didn't end up working:
  - WSL+CUDA+VS Code
  - Windows + VS
  - Google Colab
  - Live Linux USB
- Finally worked with Ubuntu!!!

Parameters

- Agents: 8 (default)
- Parallel Environments: 50 worlds (default)
  - 2, 5, 7, 10, 25, 50, 75, 100
- Scenarios: 1,000
- Device: GPU (CUDA) or CPU (depending)
- Steps per Episode: 1,000

Parallelism

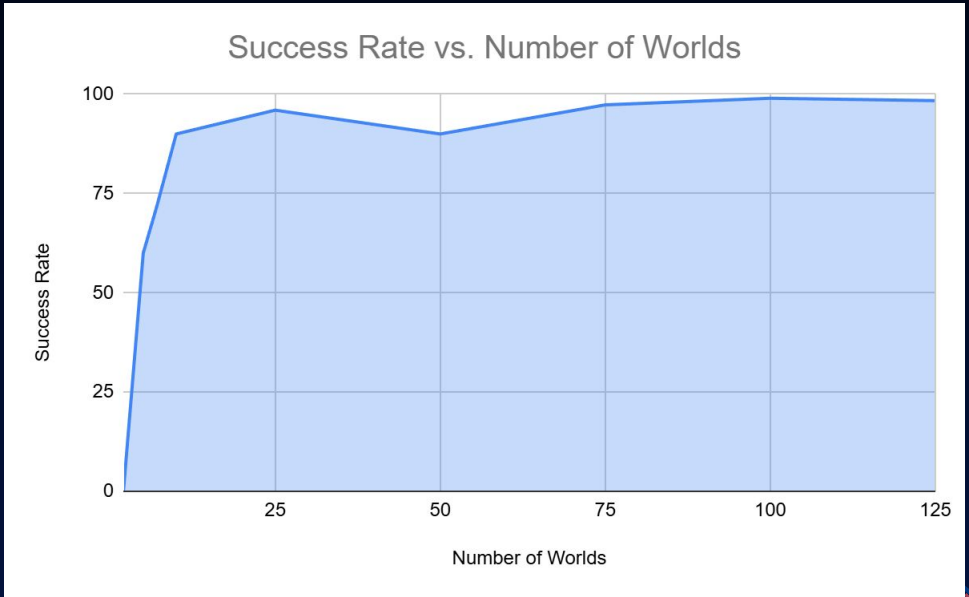
Steps/Sec vs # of Worlds



- Agents: 8
- Steps per Episode: 1,000
- Device: GPU (CUDA)

Parallelism

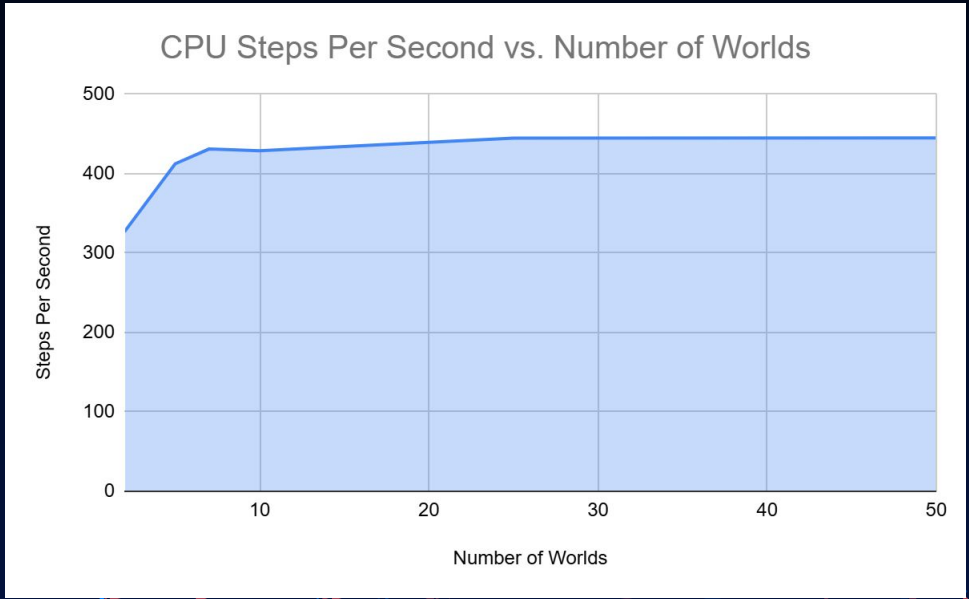
Success Rate vs # of Worlds



- Agents: 8
- Steps per Episode: 1,000
- Device: GPU (CUDA)

CPU

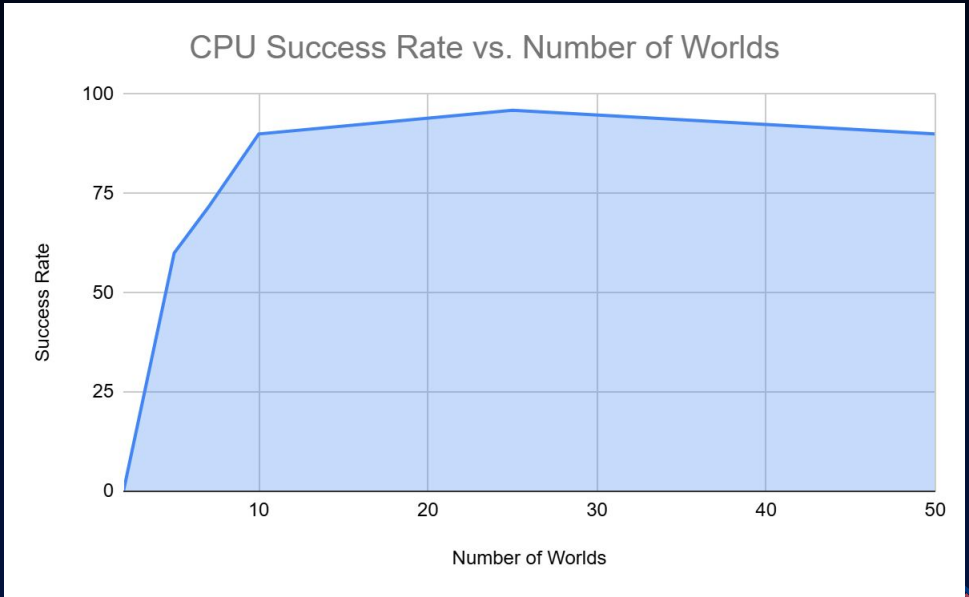
Steps/Sec vs # of Worlds



- Agents: 8
- Steps per Episode: 1,000
- Device: CPU

CPU

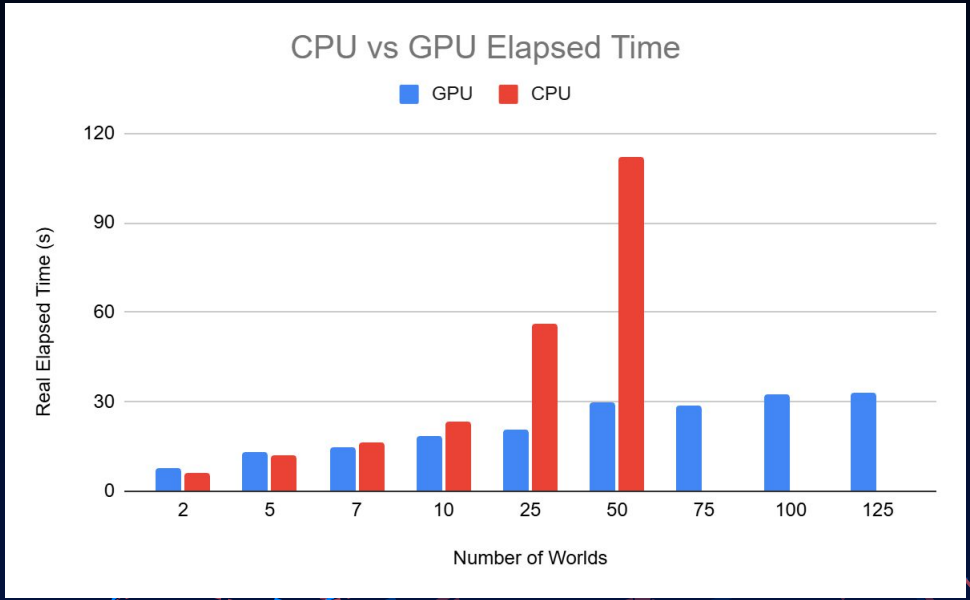
Success Rate vs # of Worlds



- Agents: 8
- Steps per Episode: 1,000
- Device: CPU

GPU vs CPU Real Elapsed Time

Elapsed Time vs # of Worlds



- Agents: 8
- Steps per Episode: 1,000
- Device: GPU (CUDA)

Conclusion

- GPU performed faster than the CPU.
- Steps per Second increased nearly exponentially when using GPU.
- GPU performed slightly better with the success rate, but considering time spent, GPU is better overall.
- CPU is more limited than GPU.
- GPUDrive is faster than other simulators we have both used such as CARLA and GTA V. However, setup needs work.

THANKS!