AuxMε

Universität Augsburg
Fakultät für Angewandte Informatik

Master Thesis

Lokale Navigation von Mikromobilitätsfahrzeugen mittels Reinforcement Learning

Marco Tröster

28.08.2023

AuxMε UNI

Agenda

1 Research Context and Goals

2 Reinforcement Learning

3 Simulation Environment

4 Trainings and Results

5 Summary and Outlook

01

RESEARCH CONTEXT AND GOALS

Research Context and Goals

Research Context: Micromobility

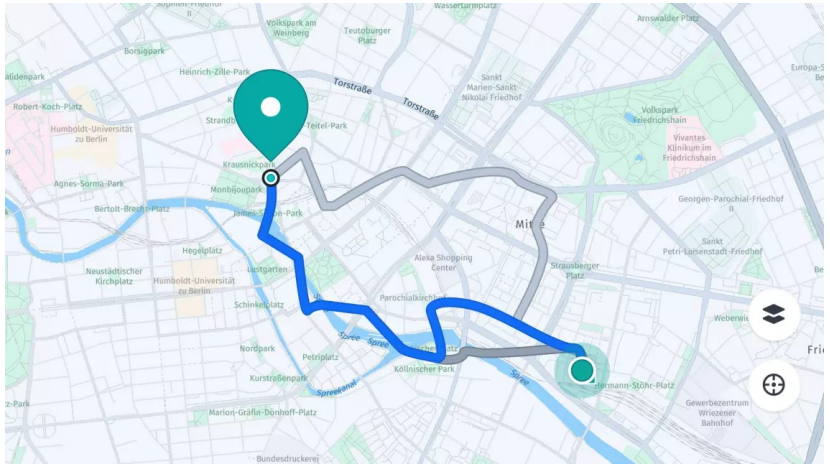
- **Use Cases**
  - Delivery drones / robots
  - Autonomous passenger transport
  - Autonomous warehouse transport
- **Challenges:**
  - Safe interaction with pedestrians
  - Adaptation to new situations

Scoomatic, a self-driving e-scooter for passenger transport

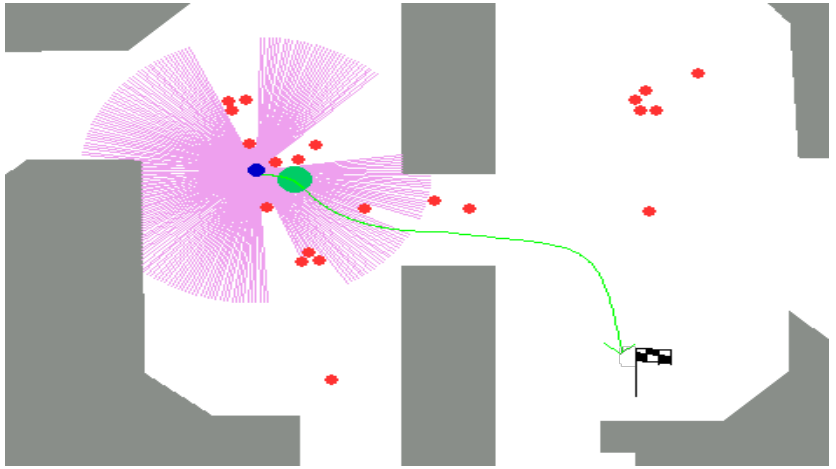
Research Context and Goals

Research Context: Autonomous Driving

- Global Navigation (= Route planning, shortest paths)
- Local Navigation (= Replacing a human driver)
  - *Control:* Acceleration and steering
  - *Perception:* Object detection
  - *Dynamics:* Prediction of movement
  - *Planning:*
    - Interaction with pedestrians / other vehicles
    - Obeying driving rules



Route planning, Source: www.here.com



Local planning in extended RobotSF simulator

Research Context and Goals

Research Context: Existing Approaches

	RobotSF (Caruso et. al)	Multi-Robot (Fan et. al)
Approach	Controllable vehicle + pedestrians (Social Force)	Swarm of controllable vehicles, no pedestrians
Issues	Unacceptable high crash rates of trained agent	No evaluation with pedestrians, just other vehicles

Research Focus

- Extend RobotSF simulation
- Optionally include good parts of Multi-Robot

Research Context and Goals

Research Goals

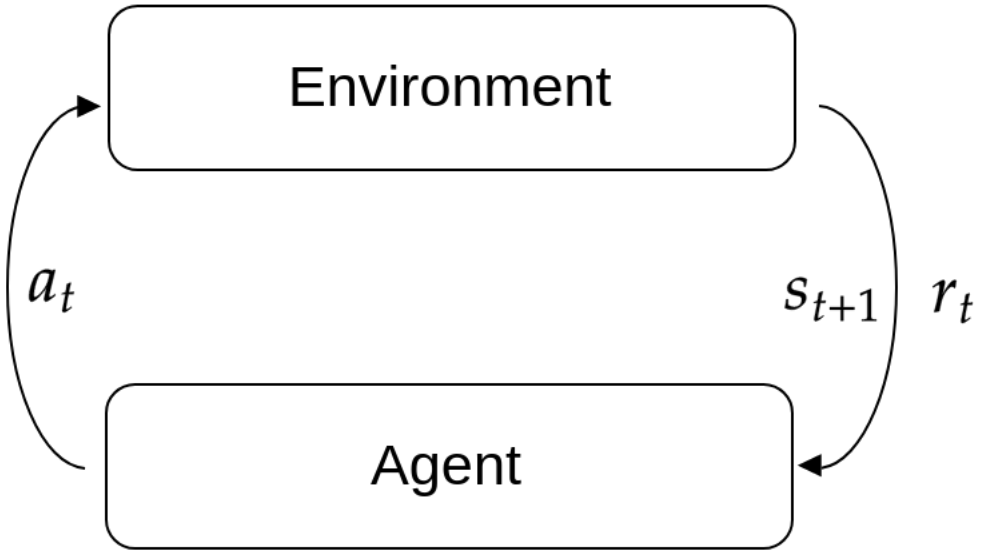
- Create a training environment to model vehicle-pedestrian interactions
- Train a safe autonomous driving software via reinforcement learning
- Evaluate the safety of the driving software with crash metrics

02

REINFORCEMENT LEARNING

Reinforcement Learning

Markov Decision Process (MDP)



Dynamics Model:

$$\rho : S \times A \mapsto S \times R$$

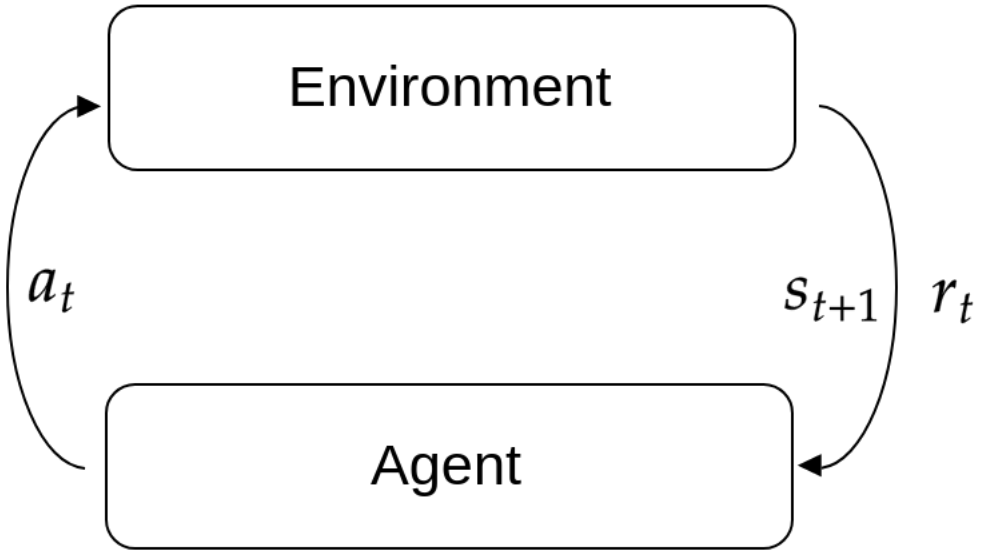
Goal: maximize the expected return $E[G]$

$$E[G] = \sum_{\tau \in T} P(\tau, \theta) \cdot R(\tau)$$

$$R(\tau) = \sum_{t=0}^{t_{\text{term}}} r_t \cdot \gamma^t$$

Reinforcement Learning

Markov Decision Process (MDP)



$$V : S \mapsto R$$

Problem: only for simple problems

$$Q : S \times A \mapsto R$$

Problem: unstable, no model of certainty

$$\pi : S \mapsto A \times [0, 1]$$

Problem: requires a lot of training data

Reinforcement Learning

Learning Policy π_θ with Actor-Critic Methods

- Maximize $E[G_t]$ by preferring profitable trajectories $\tau \in T$
  - Increase probability of actions leading into profitable trajectories
  - Decrease probability of actions leading into unprofitable trajectories
- Advantage measures the quality of an action in comparison to alternatives

Actor-Critic Algorithm

- Sample trajectories $\{\tau_1 ... \tau_m\}$ from training environment
- Estimate advantage A_t of selected action over alternative actions using V or Q
- Update the policy using empirically sampled policy gradients

Reinforcement Learning

Learning Policy π_θ via Proximal Policy Optimization with Generalized Advantage Estimation

$$\underset{\theta}{\operatorname{argmax}}\, E[\hat{A}_t] \longrightarrow \underset{\theta}{\operatorname{argmax}}\, E_{\tau\sim\theta_{old}}\left[\hat{A}_t r_t(\theta)\right], r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

$$L_t{}^{CLIP}(\theta) = min(\hat{A}_t \cdot r_t(\theta), \hat{A}_t \cdot r_t{}^{CLIP}(\theta))$$

$$r_t{}^{CLIP}(\theta) = clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$$

$$\underset{\phi}{\operatorname{argmin}}\, \frac{1}{N}\sum_{n=1}^{N}\|V_\phi(s_n) - \hat{V}_n\|^2$$

$$\hat{A}_t^{GAE(\gamma,\lambda)} := \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l}^V \qquad \delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$$

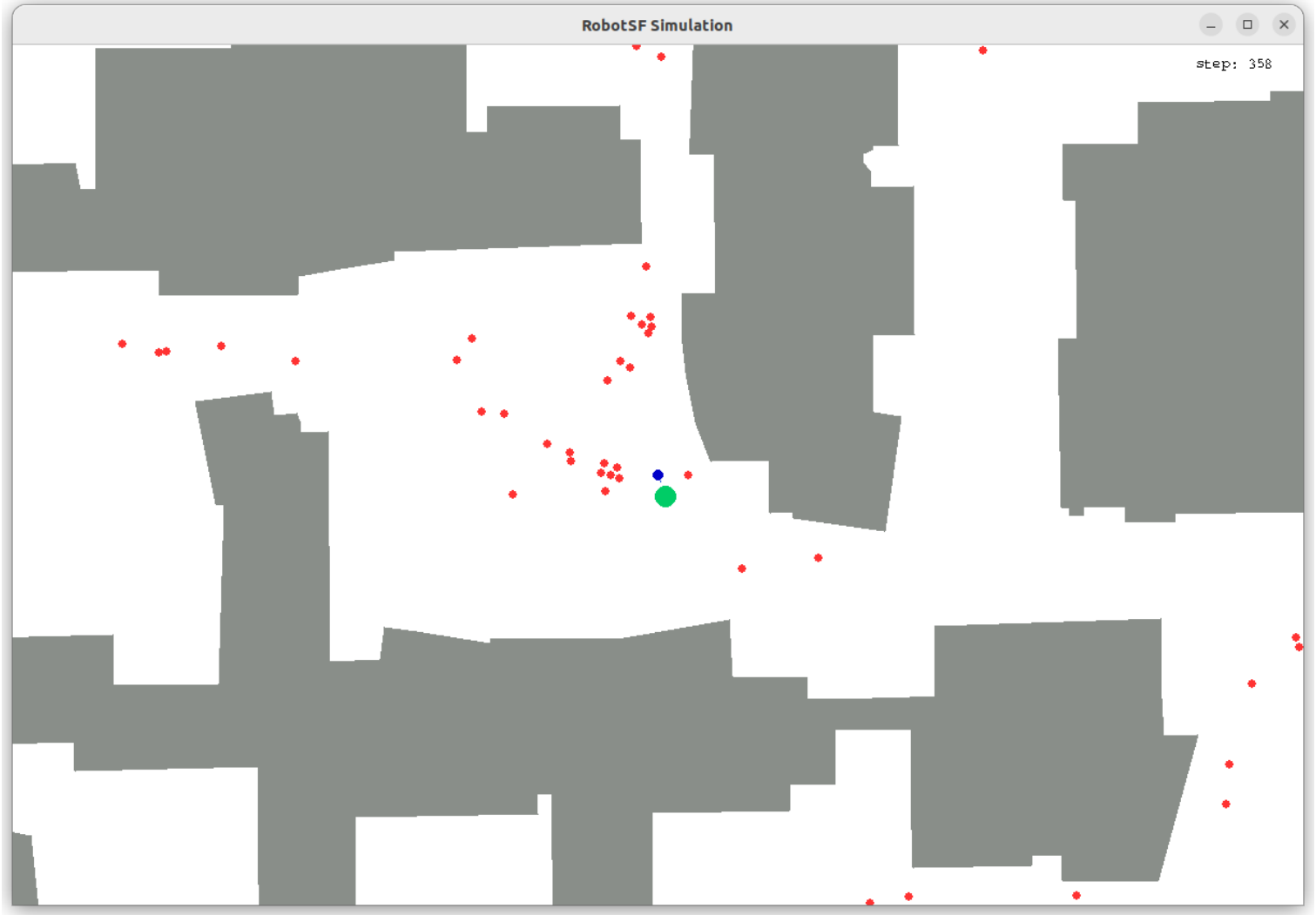
03

SIMULATION ENVIRONMENT

Simulation Environment

Simulated Entities

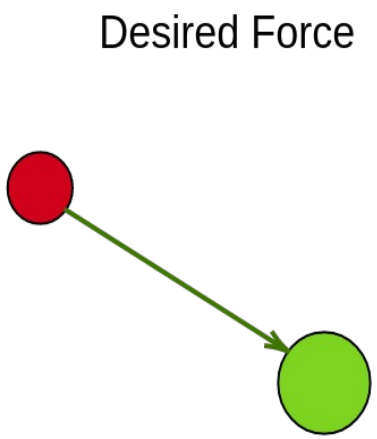
- Vehicle: blue circle
- Pedestrian: red circle
- Waypoint: green circle
- Obstacles: gray polygons

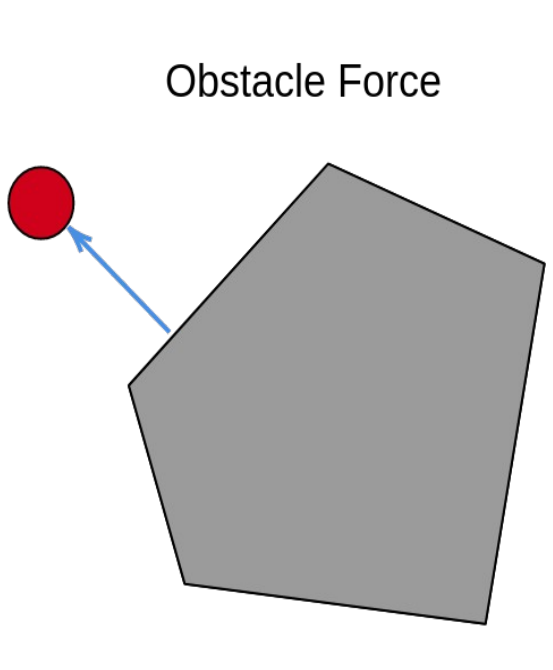


Simulation Environment

Classical Social Force

- Desired Force
- Obstacle Force
- Social Force
- Random Variance (optional)

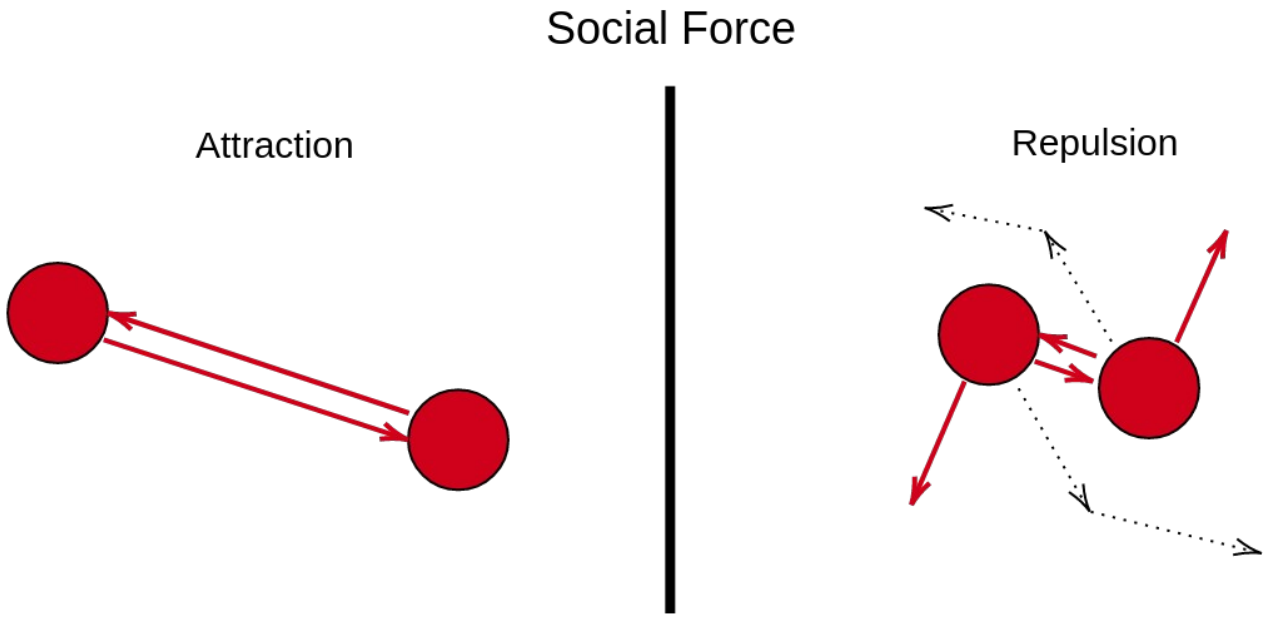




Simulation Environment

Classical Social Force

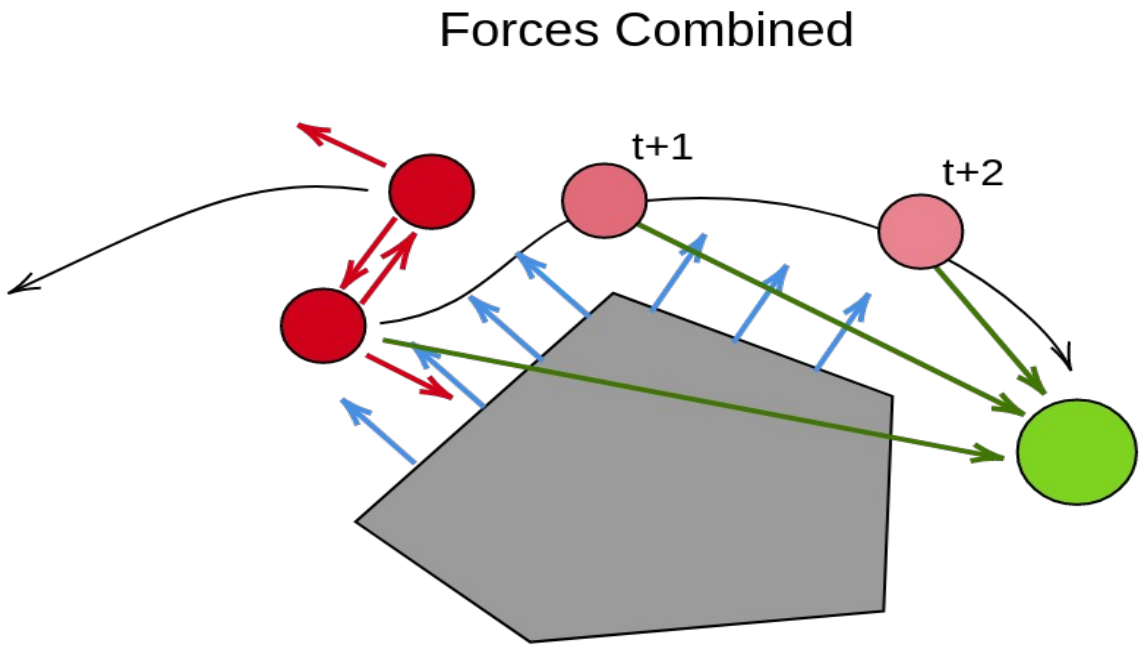
- Desired Force
- Obstacle Force
- Social Force
- Random Variance (optional)



Simulation Environment

Classical Social Force

- Desired Force
- Obstacle Force
- Social Force
- Random Variance (optional)

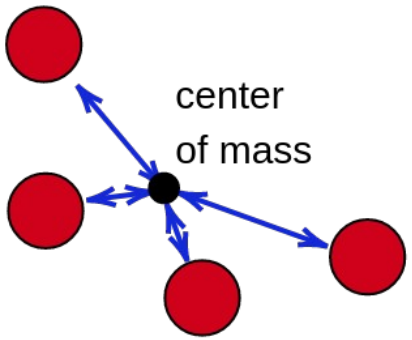


Simulation Environment

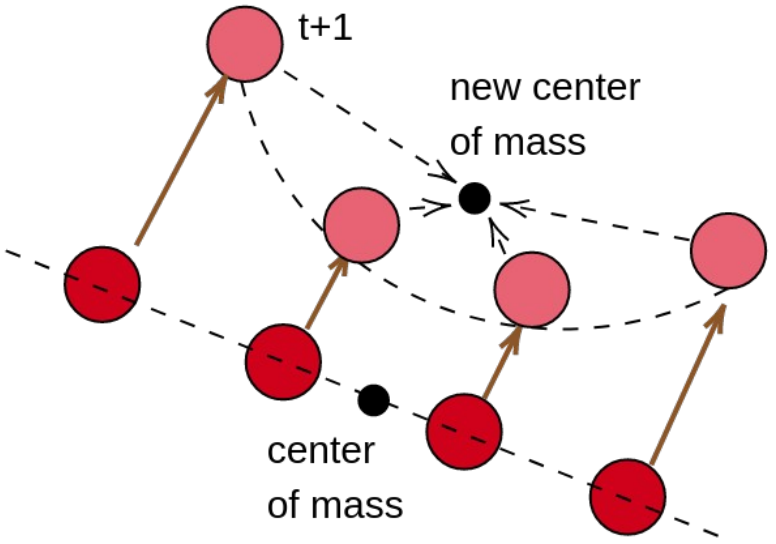
Extended Social Force

- Desired Force
- Obstacle Force
- Social Force
- Group Repulsive Force
- Group Coherence Force
- Group Gaze Force
- Ped-Robot Force
- Random Variance (optional)

Repulsive / Coherence Force



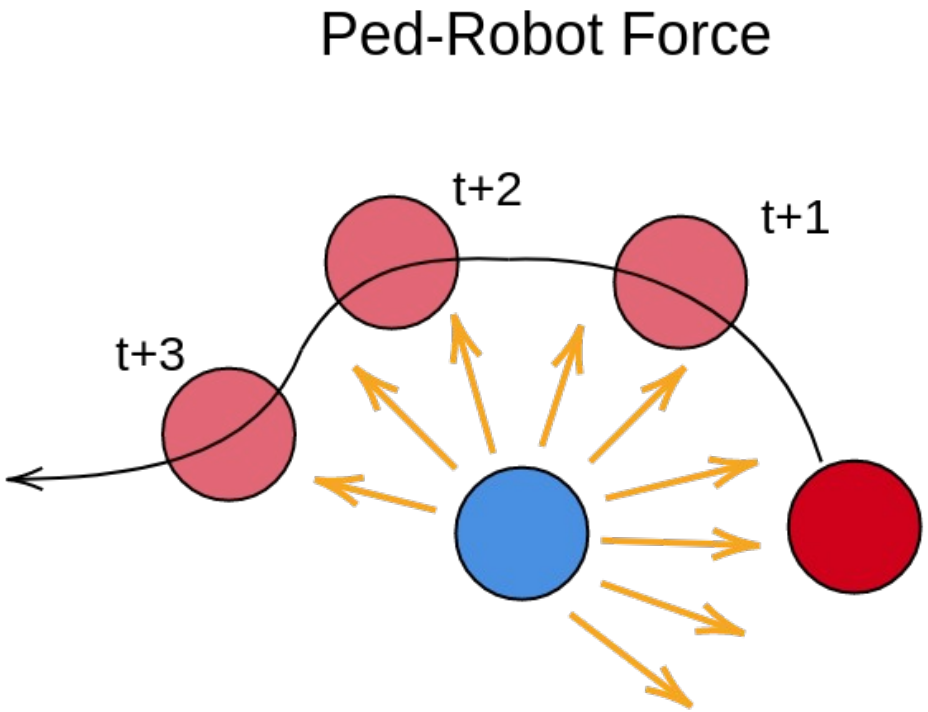
Gaze Force



Simulation Environment

Extended Social Force

- Desired Force
- Obstacle Force
- Social Force
- Group Repulsive Force
- Group Coherence Force
- Group Gaze Force
- Ped-Robot Force
- Random Variance (optional)

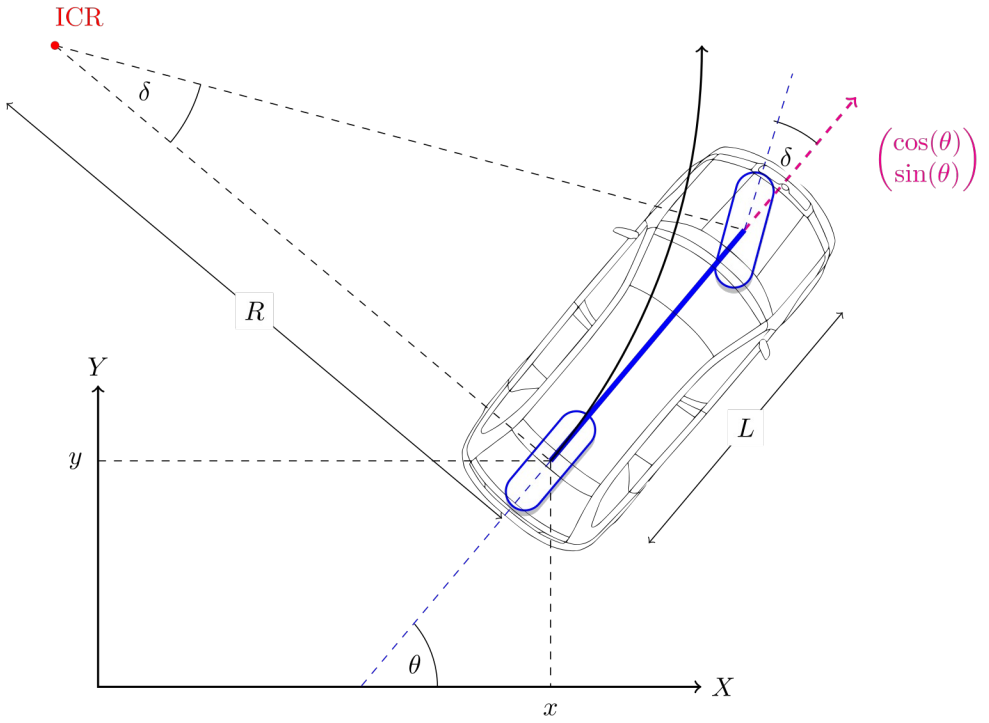


Simulation Environment

Vehicle: Kinematic Bicycle Model

- Action Space: $[\delta_{\min}, \delta_{\max}], [a_{\min}, a_{\max}]$
- Obs. Space: $[\theta_{\min}, \theta_{\max}], [v_{\min}, v_{\max}]$
- Vehicle State: $[\theta, v]$
- Vehicle Config: $[L, \delta_{\min}, \delta_{\max}, v_{\min}, v_{\max}, a_{\min}, a_{\max}]$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \cdot cos(\theta) \\ v \cdot sin(\theta) \\ v \cdot tan(\delta)/L \\ a \end{bmatrix}$$



Sketch of Bicycle Model, Source: Thomas Fermi https://thomasfermi.github.io/Algorithms-for-Automated-Driving/_images/BicycleModel_x_y_theta.svg

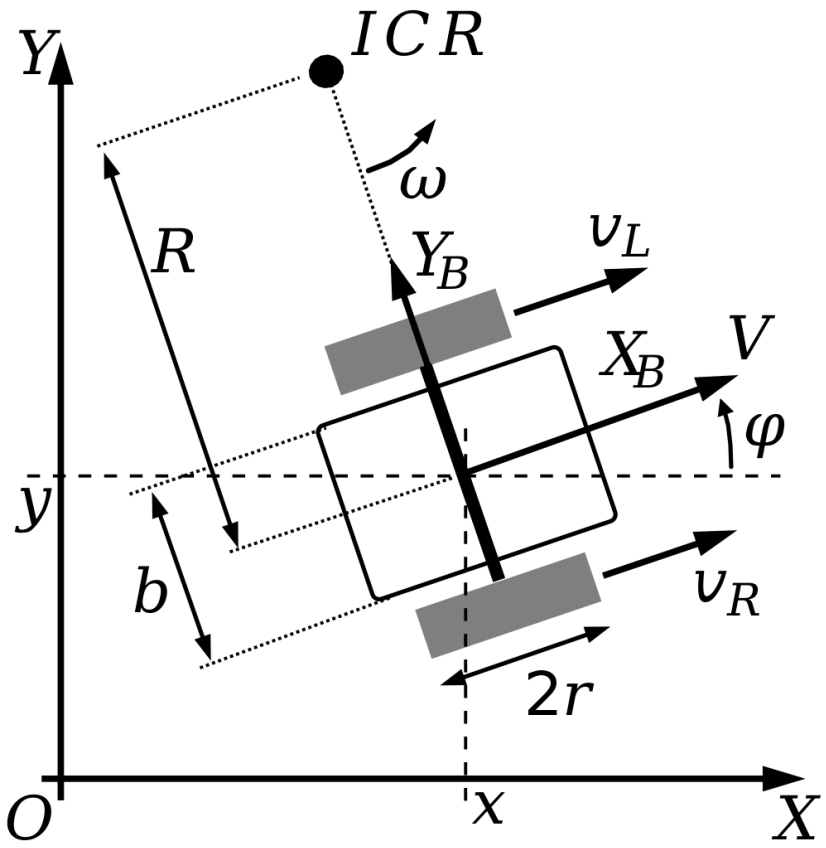
Simulation Environment

Vehicle: Differential Drive

- Action Space: $[\omega_{\min}, \omega_{\max}], [V_{\min}, V_{\max}]$
- Obs. Space: $[\varphi_{\min}, \varphi_{\max}], [V_{\min}, V_{\max}]$
- Vehicle State: $[\omega_R, \omega_L, \varphi]$
- Vehicle Config: $[r, b, V_{\min}, V_{\max}, \omega_{\min}, \omega_{\max}]$

$$\begin{bmatrix} \dot{x}_B \\ \dot{y}_B \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} v \cdot x_B \\ v \cdot y_B \\ \omega \end{bmatrix} \overset{v=r\omega}{=} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ -\frac{r}{b} & \frac{r}{b} \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos(\varphi) & 0 \\ sin(\varphi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix}$$

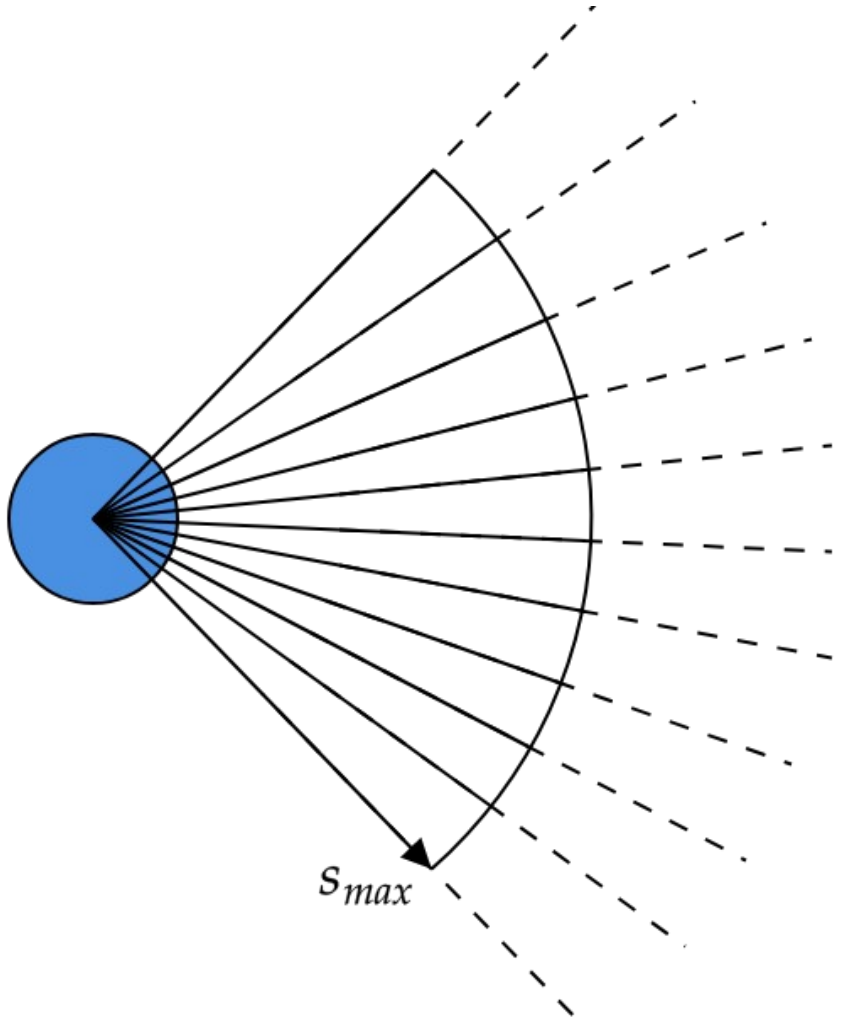


Sketch of Differential Drive, Source: Wikimedia https://commons.wikimedia.org/wiki/File:Differential_Drive_Kinematics_of_a_Wheeled_Mobile_Robot.svg

Simulation Environment

Vehicle: LiDAR Sensor

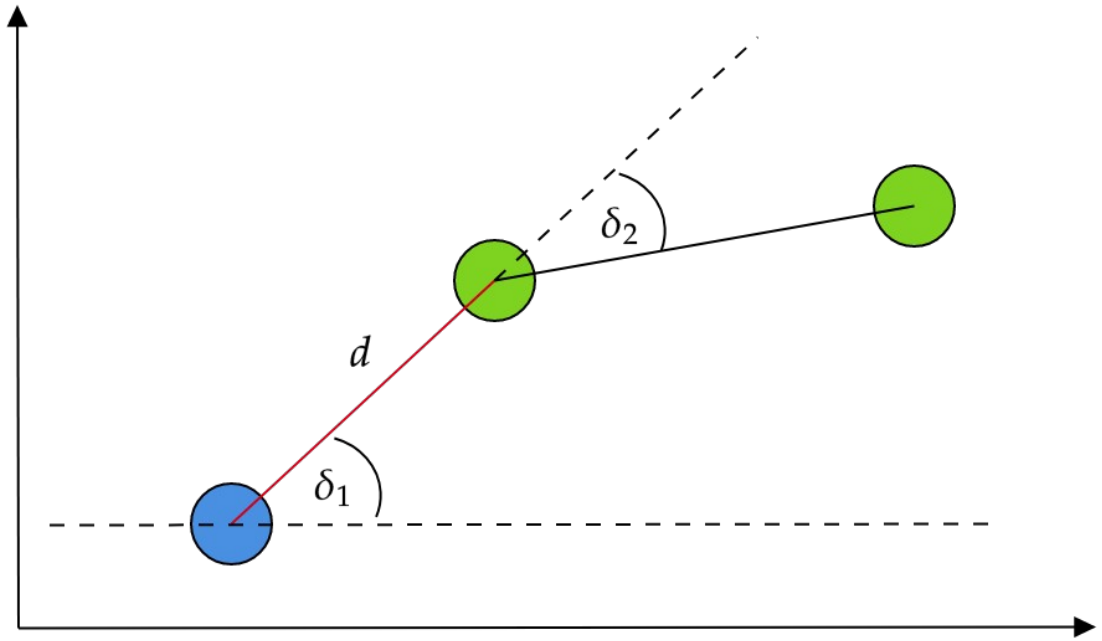
- Obs. Space: $[0, s_{\max}]^n$
- LiDAR Config: $[\varphi, s_{\max}, n]$



Simulation Environment

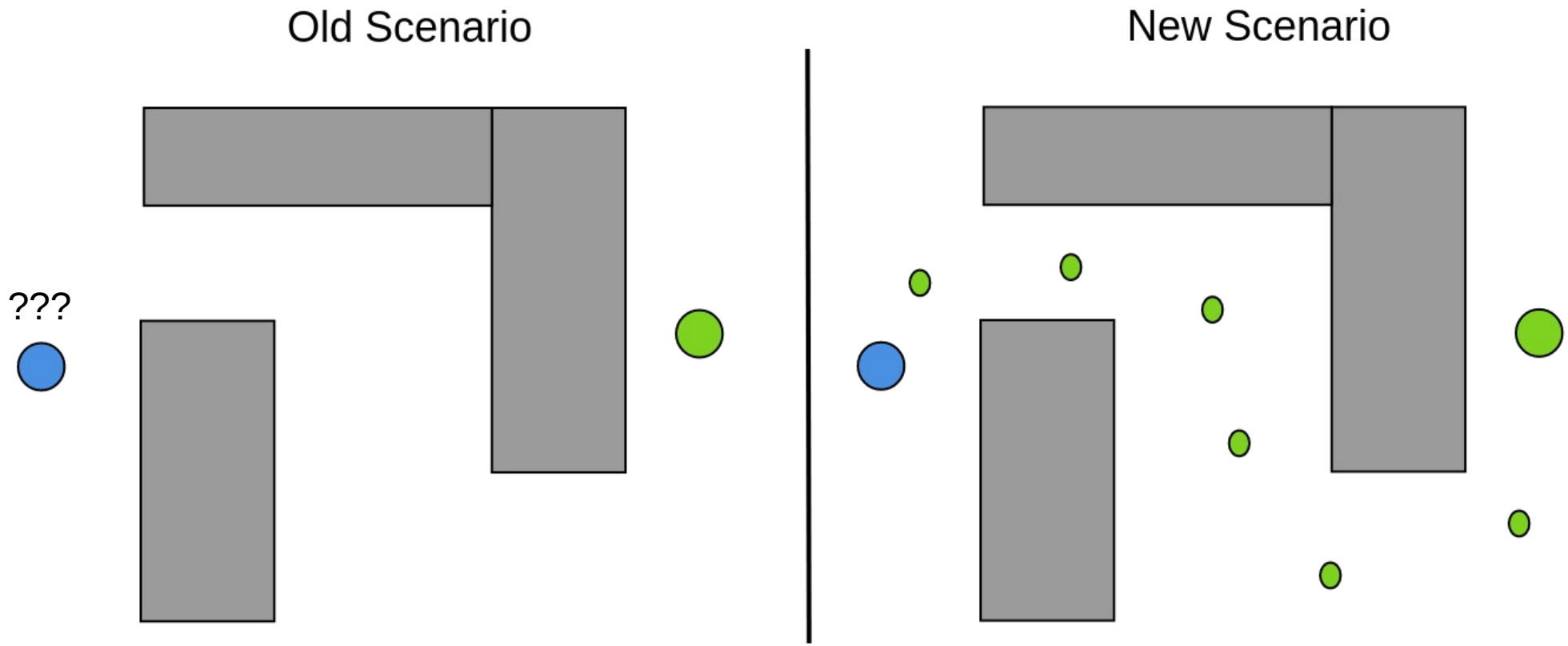
Vehicle: Goal Sensor

- Obs. Space: $[0, d_{\max}], [-\pi, \pi], [-\pi, \pi]$
- Goal Config: $[d_{\max}]$



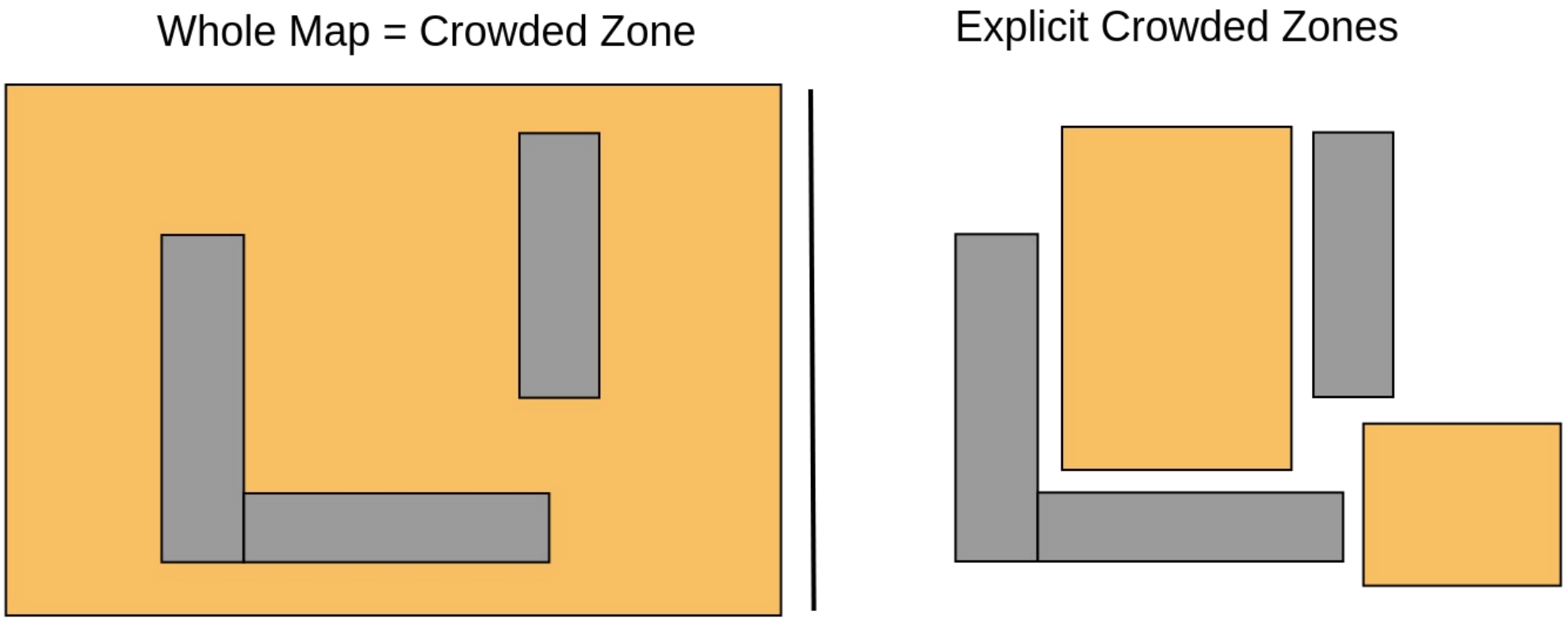
Simulation Environment

Navigation Task: Vehicle



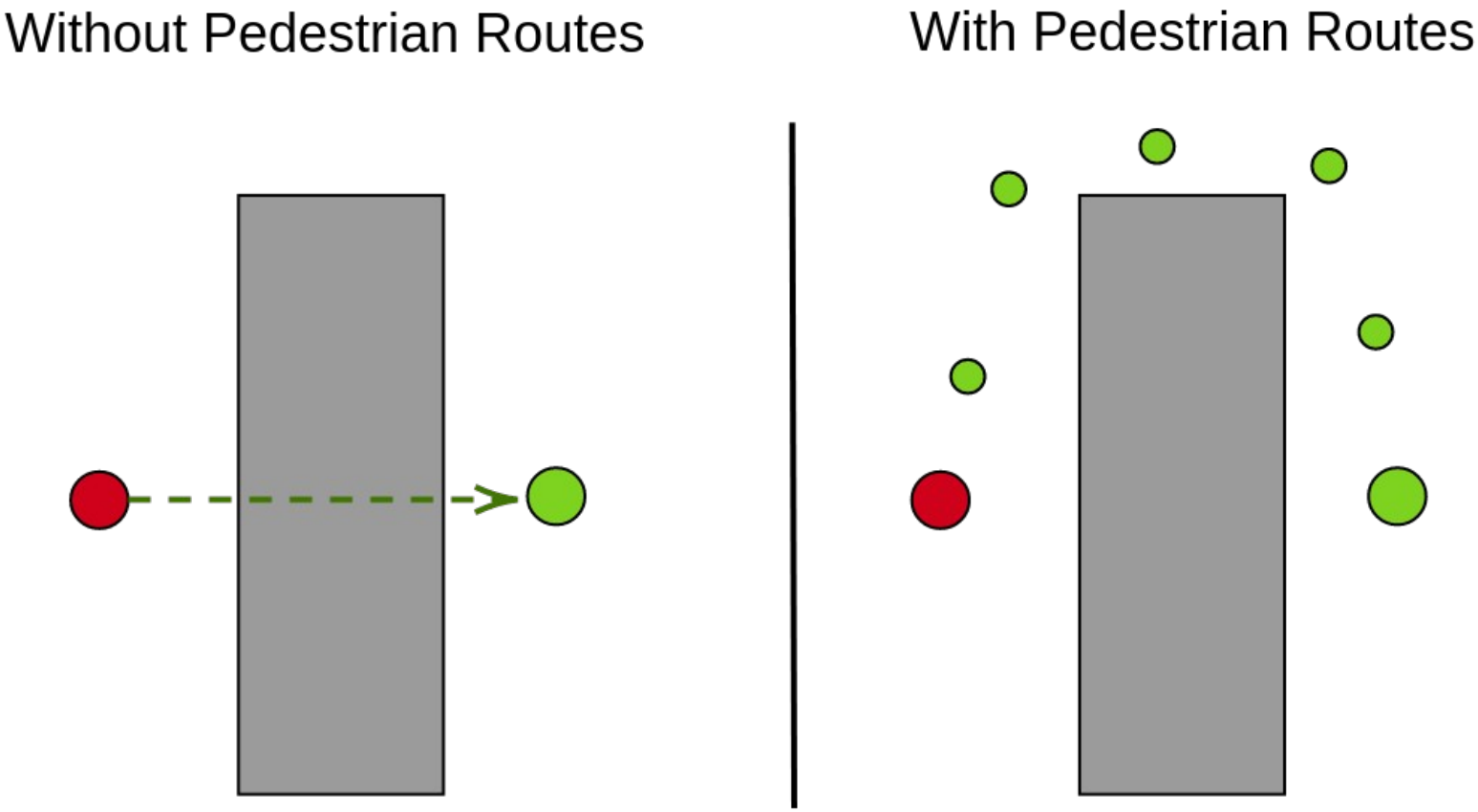
Simulation Environment

Navigation Task: Pedestrians



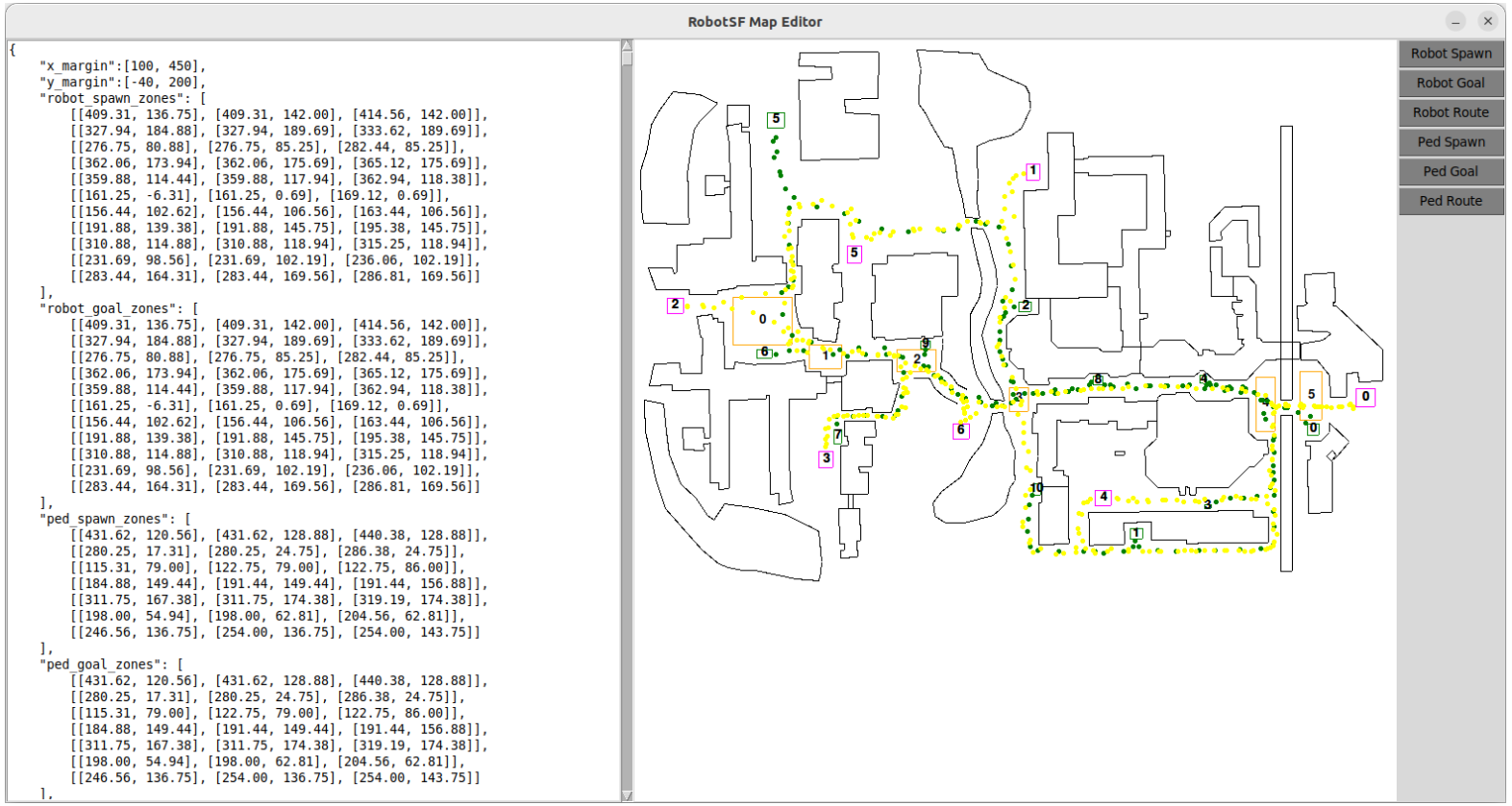
Simulation Environment

Navigation Task: Pedestrians



Simulation Environment

Map Editor: Create Scenarios



Simulation Environment

	Original	Reworked
Map material	Unrealistic maps	Real-world maps from OpenStreetMap
Navigation task	Includes global navigation, very hard to learn	Just local navigation, fast convergence
Simulation performance	Very slow, training takes more than 1 month	19x speedup, training takes less than day

04

TRAINING & RESULTS

Training and Results

Reward Function

$$r(s_t) = \frac{-0.1}{\text{max\_steps}} + \begin{cases} 1 & , \text{reached\_waypoint}(s_t) \\ -2 & , \text{is\_collision}(s_t) \\ 0 & , \text{else} \end{cases} \tag{1}$$

Training and Results

Defensive Policy: Drive Through Crowded Zones

- Learn Goals
  - Dive into crowds
  - Wait for pedestrians to pass
  - Find safe passages
- Approach
  - Observation: no timeseries
  - Ped. Density: medium – high
  - Action Rate: low

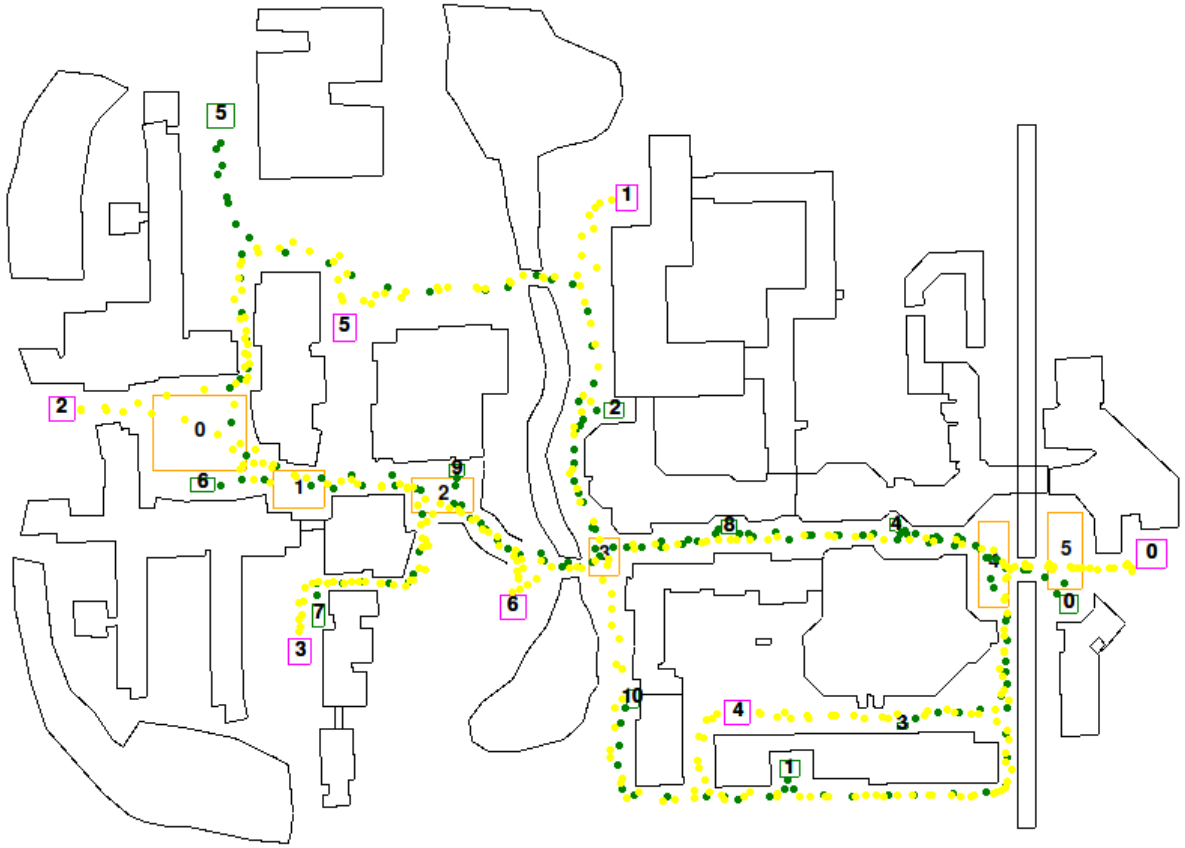
Training and Results

Demo Video: Defensive Policy

Training and Results

Offensive Policy: Learn From Realistic Scenario

- Learn Goals
  - Use real map from OpenStreetMap
  - Model mini scenarios (route design)
  - Overtake on sidewalks safely
- Approach
  - Observation: timeseries (3 steps)
  - Ped. Density: low – medium
  - Action Rate: high



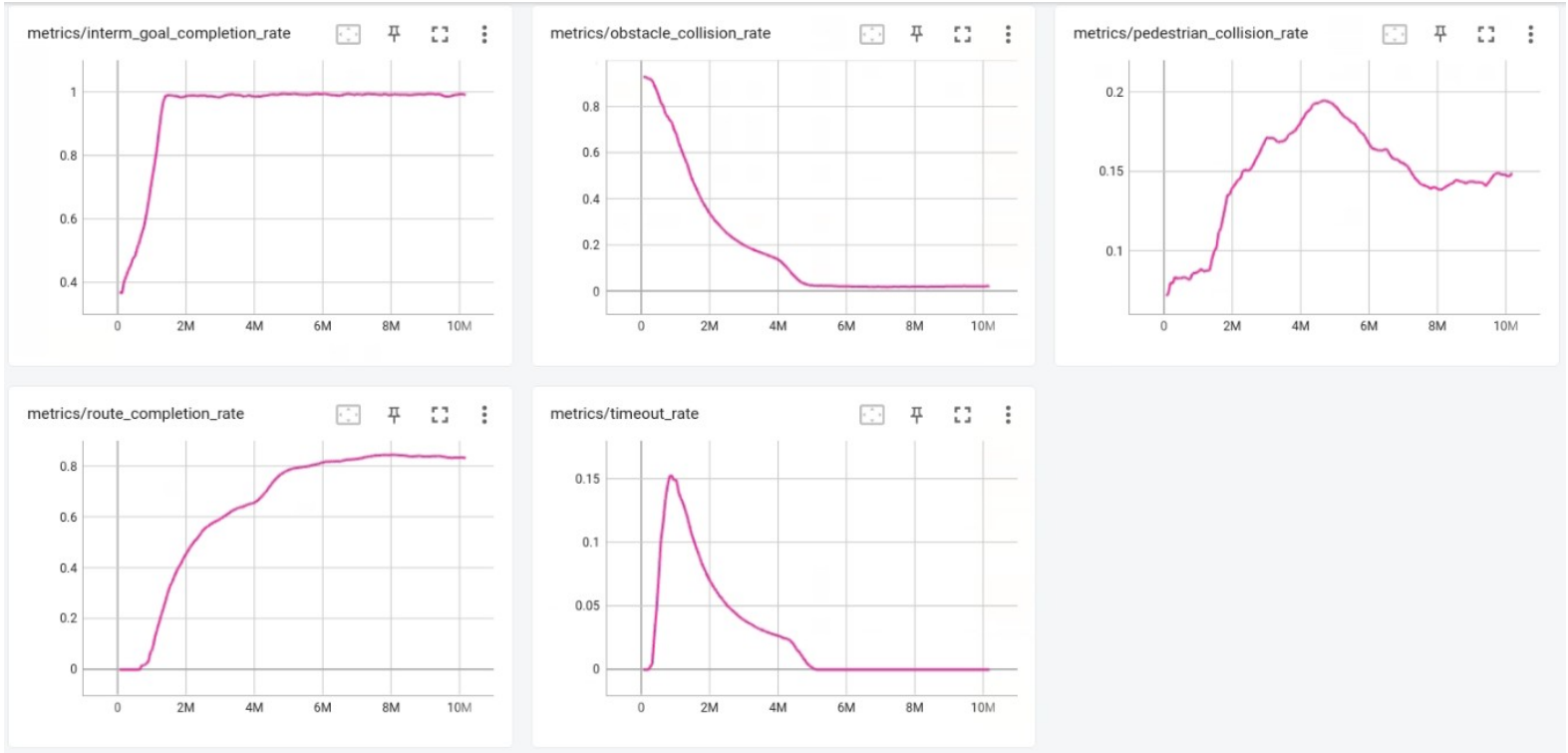
Training and Results

Demo Video: Offensive Policy

Training and Results

Driving Quality Benchmark

- Route Completion Rate
- Crash Rates
  - With pedestrians
  - With obstacles
- Timeout Rate



Training and Results

Approach	Ped. Density	Completion	Obst. Coll.	Ped. Coll.	Timeout
Caruso et. al	0.00 m^2	1.00	0.00	0.00	0.00
Best Offensive		1.00	0.00	0.00	0.00
Best Defensive		0.14	0.00	0.00	0.86
Caruso et. al	0.02 m^2	0.87	0.00	0.13	0.00
Best Offensive		0.95	0.02	0.03	0.00
Best Defensive		0.64	0.00	0.00	0.36
Caruso et. al	0.08 m^2	0.67	0.00	0.32	0.01
Best Offensive		0.60	0.09	0.30	0.01
Best Defensive		0.61	0.04	0.00	0.35
Caruso et. al	0.10 m^2	0.52	0.01	0.47	0.00
Best Offensive		0.50	0.15	0.32	0.03
Best Defensive		0.62	0.03	0.00	0.35

05

SUMMARY & OUTLOOK

Summary and Outlook

Summary: Research Goals

- Create training environment to model vehicle-pedestrian interactions
  - *Extend / enhance existing RobotSF project by Caruso et. al*

- Train a safe autonomous driving software via reinforcement learning
  - *Use PPO algorithm by Schulman et. al to learn a policy* π_θ

- Evaluate the safety of the driving software with crash metrics
  - *Use crash metrics by Caruso et. al*

Summary and Outlook

Summary: Approaches and Results

- Trained agents:
  - Defensive policy handles crowded zones (0% collisions with pedestrians)
  - Offensive policy handles less crowded zones and sidewalks (95% route completions)
  - Very simple reward function suffices to solve the navigation task

- Efficiency improvements:
  - 19x simulation speedup, enables usage of real maps
  - More efficient training algorithm (PPO vs. A3C)
  - Still lots of potential for improvement, e.g. using quad-trees

Summary and Outlook

Outlook: Ideas for Upcoming Research

- Combine RobotSF with Multi-Robot:
  - Simulation of pedestrians uses > 80% of computation
  - Drive multiple routes at once with multiple vehicles
  - Sample 5x - 10x more training data in the same time

- Model-based Reinforcement Learning:
  - Dreamer architecture learns dynamics model ρ
  - While training: replay recorded scenarios
  - Efficiency improvements of 100x - 1000x

Thanks for your attention!

Any Questions?

Master Thesis

Lokale Navigation von Mikro-mobilitätsfahrzeugen mittels Reinforcement Learning

Marco Tröster

28.08.2023

Lehrstuhl für Mechatronik
Universität Augsburg
marco.troester@student.uni-augsburg.de
www.uni-augsburg.de

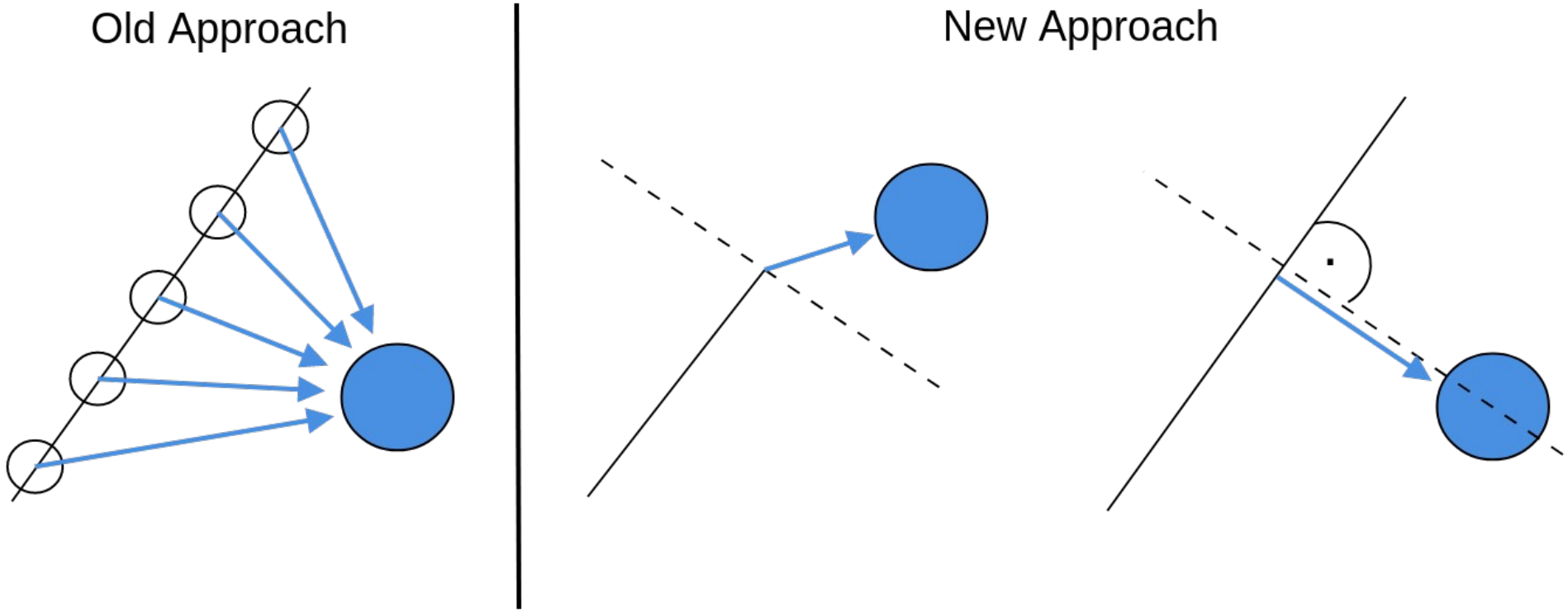
06

APPENDIX

PERFORMANCE OPTIMIZATION

Appendix

Performance Optimization: Obstacle Force



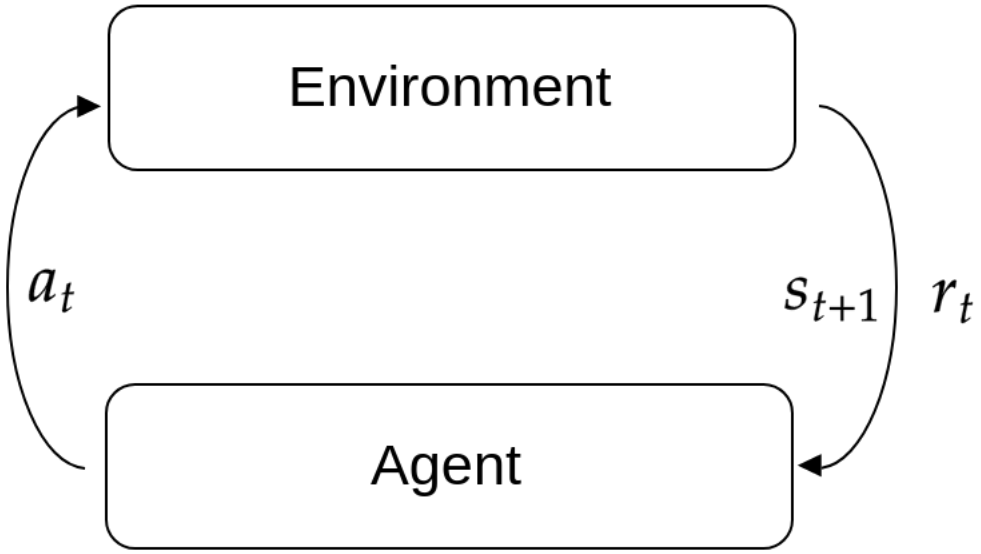
07

APPENDIX

PPO ALGO

Appendix

Markov Decision Process (MDP)



Dynamics Model:

$$\rho : S \times A \mapsto S \times R$$

Goal: maximize the expected return $E[G]$

$$E[G] = \sum_{\tau \in T} P(\tau, \theta) \cdot R(\tau)$$

$$R(\tau) = \sum_{t=0}^{t_{\text{term}}} r_t \cdot \gamma^t$$

Appendix

Learning Policy π_θ via Vanilla Policy Gradient Methods

$$E[G_t] = \sum_{\tau \epsilon T} R(\tau) P(\tau, \theta) \qquad P(\tau, \theta) = \prod_{t=0}^{t_{\text{term}}} \pi_\theta(a_t|s_t) \cdot P(s_{t+1}|s_t, a_t)$$

Approach: Maximize the expected return $E[G_t]$ by adjusting the policy $\pi_\theta(a_t \mid s_t)$ to select more favorable actions

$$\underset{\theta}{\text{argmax}}\, E[G_t]$$

$$\nabla_\theta E[G_t] = \sum_{\tau \in T} P(\tau, \theta) R(\tau) \nabla_\theta P(\tau, \theta) \qquad \nabla_\theta P(\tau, \theta) = \sum_{t=0}^{t_{\text{term}}} \nabla_\theta log(\pi_\theta(a_t|s_t))$$

Appendix

Learning Policy π_θ via Vanilla Policy Gradient Methods

$$\nabla_\theta E[G_t] = \sum_{\tau \in T} P(\tau, \theta) R(\tau) \nabla_\theta P(\tau, \theta)$$

Problem: Dynamics model ρ is unknown; cannot compute $P(\tau, \theta)$

Solution: Approximate $\nabla_\theta E[G_t]$ using an empirical sample over $\{\tau_1 \dots \tau_m\}$

$$\nabla_\theta E[G_t] \approx \frac{1}{m} \sum_{\tau \in \{\tau_1 \dots \tau_m\}} R(\tau) \sum_{t=0}^{t_{\text{term}}} \nabla_\theta log(\pi_\theta(a_t|s_t))$$

Appendix

Learning Policy π_θ via Vanilla Policy Gradient Methods

$$\nabla_\theta E[G_t] \approx \frac{1}{m} \sum_{\tau \in \{\tau_1 \dots \tau_m\}} R(\tau) \sum_{t=0}^{t_{\text{term}}} \nabla_\theta log(\pi_\theta(a_t|s_t))$$

Simplifcation: Assuming the policy is optimal for all following steps, only the next action matters to lead into good trajectories. All following steps can be seen as subtrajectories.

$$\nabla_\theta E[G_t] \approx \frac{1}{m} \sum_{\tau \in \{\tau_1 \dots \tau_m\}} R(\tau^{(t)}) \nabla_\theta log(\pi_\theta(a_t|s_t))$$

Problem:

$$\nabla_\theta log(\pi_\theta(a_t \,|\, s_t)) > 0$$

Sign of $R(\tau^{(t)})$ determines whether $\pi_\theta(a_t \,|\, s_t)$ increases or decreases

Appendix

Learning Policy π_θ via Advantage Actor-Critic (A2C) with Generalized Advantage Estimation (GAE)

$$\nabla_\theta E[G_t] \approx \frac{1}{m} \sum_{\tau \in \{\tau_1 \dots \tau_m\}} R(\tau^{(t)}) \nabla_\theta log(\pi_\theta(a_t|s_t))$$

Problem: Sign of $R(\tau^{(t)})$ determines whether $\pi_\theta(a_t \mid s_t)$ increases or decreases

Solution: Replace $R(\tau^{(t)})$ with an estimation of the advantage $A_t = Q(s_t, a_t) - V(s_t)$

Problem:

Training uses trajectories only once

Very inefficient

$$\hat{A}_t^{GAE(\gamma,\lambda)} := \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \qquad \delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Appendix

Learning Policy π_θ via Proximal Policy Optimization (PPO) with GAE

$$\underset{\theta}{\operatorname{argmax}}\, E[\hat{A}_t] \longrightarrow \underset{\theta}{\operatorname{argmax}}\, E_{\tau\sim\theta_{old}}\left[\hat{A}_t r_t(\theta)\right], r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

Approach: Use trajectories sampled with θ_{old} multiple times
Problem: Policy ratio $r_t(\theta)$ becomes too small / big, unstable training
Solution: Clip the ratio using objective function L^{CLIP}

$$L_t{}^{CLIP}(\theta) = min(\hat{A}_t \cdot r_t(\theta), \hat{A}_t \cdot r_t{}^{CLIP}(\theta))$$

$$r_t{}^{CLIP}(\theta) = clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$$

Appendix

Learning Policy π_θ via Proximal Policy Optimization (PPO) with GAE

$$\nabla_\theta E[{L_t}^{CPI}(\theta)] = \nabla_\theta E[\hat{A}_t r_t(\theta)] = E[\hat{A}_t r_t(\theta) \nabla_\theta log(\pi_\theta(a_t|s_t))]$$

$$\nabla_\theta E[{L_t}^{CLIP}(\theta)] = \nabla_\theta E[\hat{A}_t {r_t}^{CLIP}(\theta)] = E[\hat{A}_t \nabla_\theta {r_t}^{CLIP}(\theta)]$$

$$r_t(\theta) \notin [1-\epsilon, 1+\epsilon] \implies \nabla_\theta {r_t}^{CLIP}(\theta) = 0$$

Result: Only training examples yielding policy ratios within ϵ bound participate in the policy update -> trajectories can be re-used

08

APPENDIX

MODEL STRUCTURE

Appendix

Model Structure: Neural Network

- Input Layer
  - LiDAR rays (272)
  - Target sensor (3)
  - Drive state (2)
- Output Layer
  - Actuators: acceleration and steering (4)
  - Per actuator: sample normal distributed value from predicted σ and μ