Applying Deep Reinforcement Learning in the Navigation of Mobile Robots in Static and Dynamic Environments

Ronja Güldenring

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

16. April 2019

Outline

Table of Contents

Motivation

Related Work

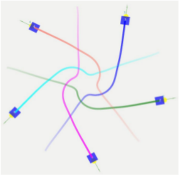


Figure 1: Multi robot scenario [1]

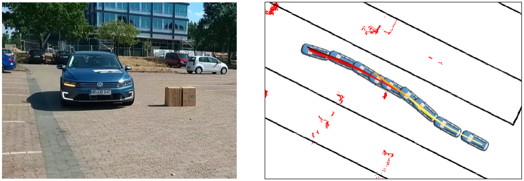


Figure 2: Self-driving car [2]

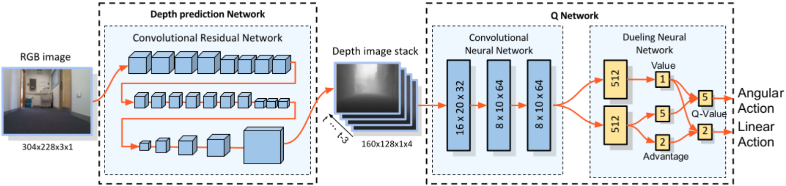


Figure 3: Dueling-Double DQN applied to very noisy depth images. [3]

Table of Contents

MiR100 robot

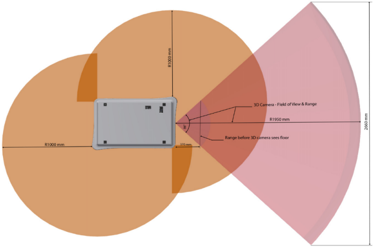


Figure 4: MiR100 robot of the company Mobile Industrial Robots ApS [1].

navigation: global planner + local planner

[1] accessed 2019-01-27: http://www.mobile-industrial-robots.com/de/products/mir100/

Reinforcement Learning (RL)

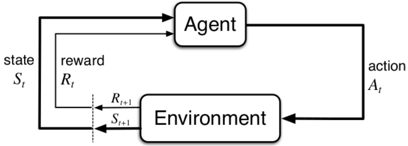


Figure 5: Reinforcement Learning Loop.[5]

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad (1)$$

- Policy: Agent samples action from probability distribution $\pi(a|s)$.
- Value function $v_\pi(s)$: estimate of how good it is for the agent to be in state s.
- Action-value function $q_\pi(s, a)$: estimate of how good it is to take action a in state s.

RL – Q-Learning

Data: $\pi, \alpha \in (0,1]$
Initialize Q(s), for all $s \in \mathcal{S}$ arbitrarily;
for *each episode* **do**
    Initialize S_t **do**
        $A_t \leftarrow$ action given by π for S_t;
        Take action A_t, observe R_{t+1} and S_{t+1};
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
        $S_t \leftarrow S_{t+1}$
    while *S is not terminal*;
end

$\rightarrow$ Deep RL (DRL): replace table Q(s,a) with function approximator

DRL – Proximal Policy Optimization (PPO) [7]

- Policy Gradient Method
  - optimization of the policy $\pi(a|s,\theta)$ directly
  - Actor-Critic Architecture
- builds on TRPO [6].
- learns relatively quickly/stable
- easy to tune

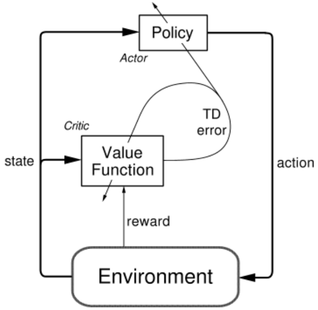


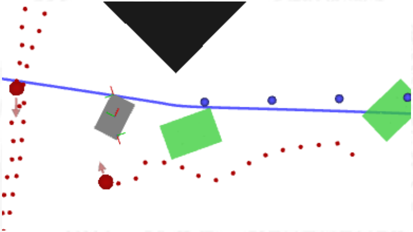
Figure 6: Actor-Critic Architecture.[5]

- **Clipped Surrogate Objective**
  - restricting the update size from one policy to another
  - stable updates
  - prevents optimization overshooting the maximum

Table of Contents

Simulation Environment

- Restricted to 2D-problem
- $\rightarrow$ 2D laser scanner as sensor source
  - + data approximates real world more realistically
  - + less computational expensive
  - – provides less features
- Flatland as base simulator [10]
- Pedsim for crowd simulation [9]
- three different obstacle types:
  - global static obstacle
  - local static obstacle
  - dynamic obstacle (pedestrian)



PedSim Crowd Simulator [9]

- Helbing's Social Force Model [8]
  - **Desired Force** f_{des}
  - **Pedestrian Force** f_{ij}
  - **Wall Force** f_{iW}
  - **Robot Force** f_r

$$\boxed{F_{sum} = f_{des} + \sum_j f_{ij} + \sum_W f_{iW} \; (+f_r)} \qquad (2)$$

- Semi-polite pedestrian

- Pedestrian-plugin:
  synchronises the pedestrian state of the PedSim simulator with the Flatland simulator

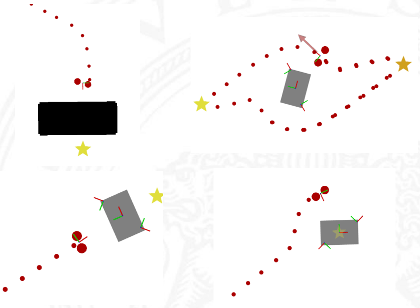
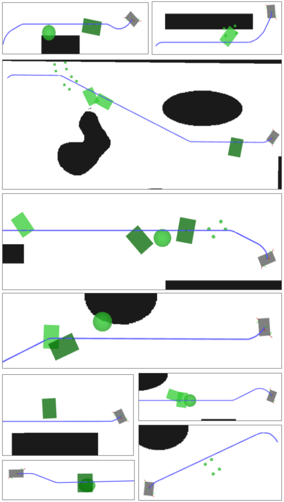
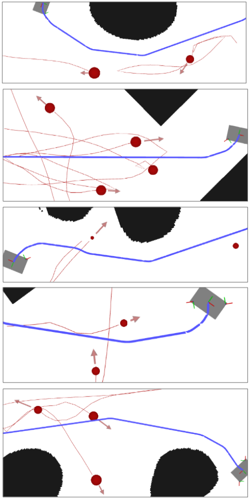


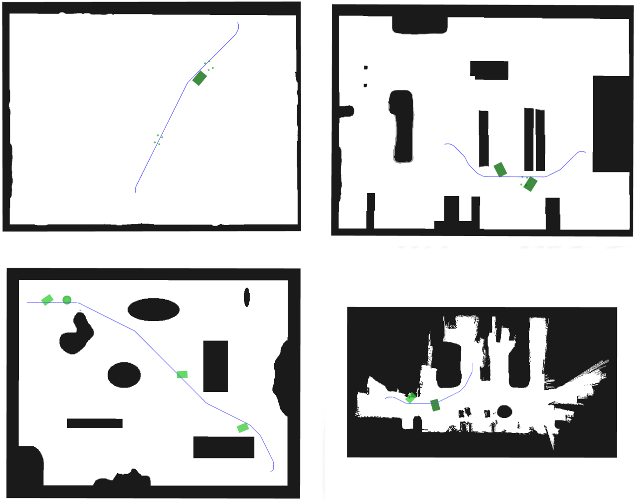
Table of Contents

Task setup


(a) Static Setup


(b) Dynamic Setup

Global world setup



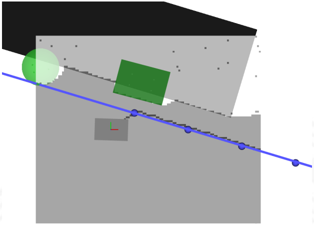
RL-agent setup

- RL-agent replaces traditional local planner
- Proximal Policy Optimization
  - PPO1/PPO2 implementation stable baselines library [11]
  - Tensorflow
- Wrapper class *Ros_env*
  - implements *gym.Env*-interface.
  - communicates with ROS side.

Observation and action space

Observation Space

- Raw Data Representation
- X-Image Representation
- X-Image Speed Representation



Action Space

- 6 discrete actions as combination of tranlational and rotational velocity.
  $[0, -\omega_{max}]$,
  $[v_{max}, 0]$,
  $[0, \omega_{max}]$,
  $[v_{max}, \omega_{max}/2]$,
  $[v_{max}, -\omega_{max}/2]$,
  $[0, 0]$,
  $([0.09, 0])$

Reward function 1

$$r_t = r_t(wp) + r_t(o) + r_t(g) \tag{3}$$

$$r_t(g) = \begin{cases} R_g & \text{if } d(p_{r,t}, p_g) < D_g \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$r_t(o) = \begin{cases} -R_o & \text{if collision with an obstacle } \in O \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$r_t(wp) = \begin{cases} 0 & \text{if } \min\limits_{o_i \in O}(d(p_{o_i,t}, p_{r,t})) < D_o \\ r'_t(wp) & \text{otherwise} \end{cases} \tag{6}$$

$$r'_t(wp) = r_{1t}(wp) + r_{2t}(wp) + r_{3t}(wp) \tag{7}$$

$$\text{diff}(p_{r,t}, p_{wp,t}) = d(p_{r,t-1}, p_{wp,t-1}) - d(p_{r,t}, p_{wp,t}) \tag{8}$$

$$r_{1t}(wp) = \begin{cases} w_1 \cdot \text{diff}(p_{r,t}, p_{wp,t}) & \text{if diff}(p_{r,t}, p_{wp,t}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$r_{2t}(wp) = \begin{cases} w_2 \cdot \text{diff}(p_{r,t}, p_{wp,t}) & \text{if diff}(p_{r,t}, p_{wp,t}) < 0 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$r_{3t}(wp) = \begin{cases} R_{wp} & \text{if } d(p_{r,t}, p_{wp,t}) < D_{wp} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Reward function 2

$$r_{t,2} = r_t(wp) + r_{t,2}(o) + r_t(g) + r_t(vel) \tag{12}$$

$$r_{t,2}(o) = \min(r_t(so), r_t(ped)) \tag{13}$$

$$r_t(so) = \begin{cases} -R_{so} & \text{if collision with a static obstacle } \in SO \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$r_t(ped) = \begin{cases} 0 & \text{if } \min\limits_{ped_i \in PED}(d(p_{ped_i,t}, p_{r,t})) > D_{ped} \\ & \textbf{or } v \leq v_{reaction,max} \text{ for a duration of } t_{reaction} \\ -R_{ped} & \text{otherwise} \end{cases} \tag{15}$$

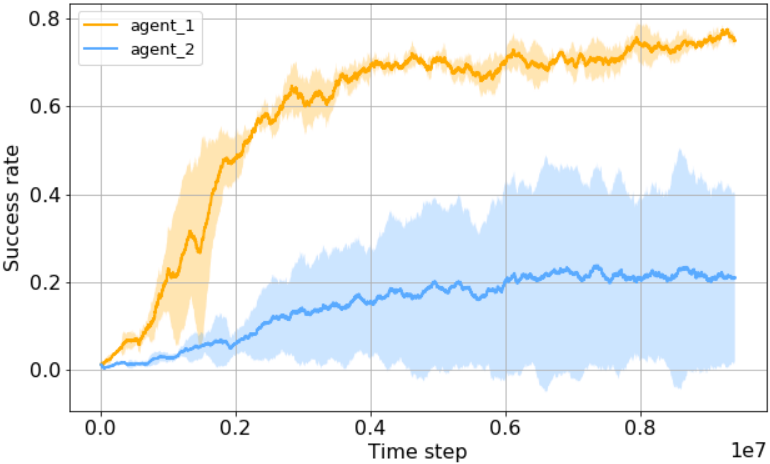
$$r_t(vel) = \begin{cases} -R_{vel1} & \text{if } v_t = 0 \text{ and } \omega_t = 0 \\ -R_{vel2} & \text{if } v_t = 0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Table of Contents

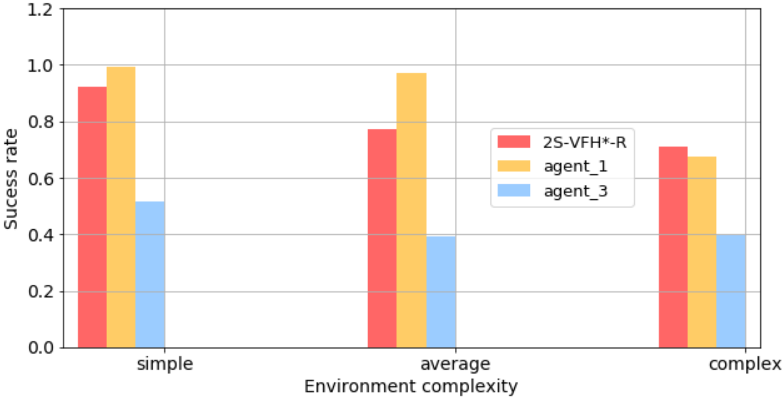
Static agents

	agent_1	**agent_3**
Action Space	discrete $v_{max} = 0.5$ $\omega_{max} = 0.5$	
State Input	1-Image Representation	Raw Data Representation
Network architecture	4-layered 2D-CNN	1D-CNN
Reward function	reward function 1	
Reward function parameters	table 1	

Static Agents – training results



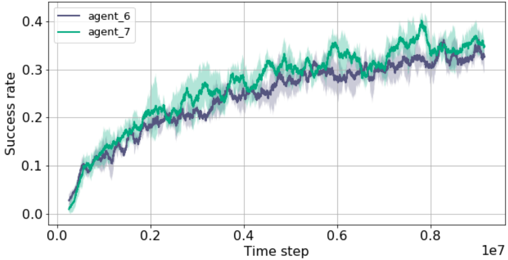
Static agents – test results

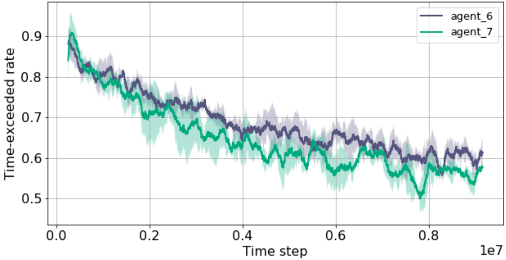


Dynamic agents

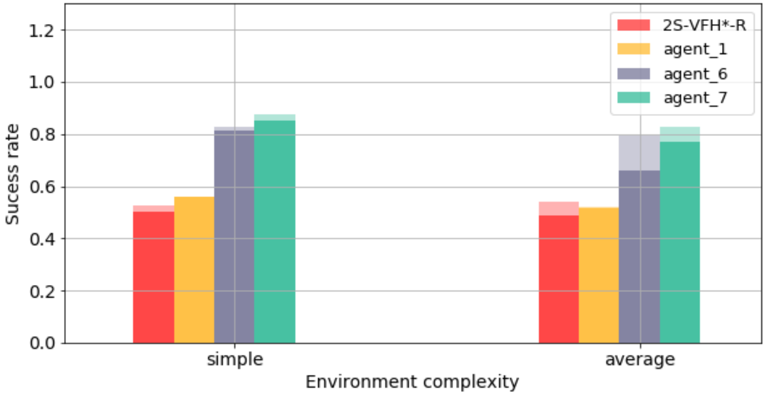
	agent_6	agent_7
Reward function	reward function 2	
Reward function parameters	table 2	table 3
Action Space	discrete $v_{max} = 0.5$ $\omega_{max} = 0.7$	discrete $v_{max} = 0.5$ $\omega_{max} = 0.7$ $+ [0.09, 0]$
State Input	4-Image Speed Representation	
Network architecture	6-layered 2D-CNN	

Dynamic agents – training results





Dynamic agents – test results



Video

https://www.youtube.com/watch?v=laGrLaMaeT4

Table of Contents

Conclusion

Implementation

- successful integration of the DRL-library stable-baselines [11] in the ROS navigation stack.
- fusion of the Flatland simulator with the PedSim crowd simulator.

Static Training

- Image Representation and discrete action space generates the best results.
- stable avoidance of static objects.

Dynamic Setup

- different reasonable policies were trained.
- high potential for improvements, but "proof-of-concept" is fulfilled.

Future Work

- Increase success rate and improve learned policy of agent_6 and agent_7.
- Train in a more complex and realistic dynamic (and static) setup.
  - Apply normal walking speed of the pedestrians.
  - Train in more complex maps with more clutter and narrow corridors.
  - Train with more complex pedestrian behaviors $\rightarrow$ learning social behavior.
- Fusion of the traditional local planner with the rl-local planner. The rl-local planner is triggered, when moving objects are detected.

Questions?

Appendix: reward parameter sets

Parameter	Value
R_g	10
R_o	15
D_o	0.96
w_1	2.5
w_2	3.5
R_{wp}	1.0
D_{wp}	0.2

Table 1: Parameter set for Reward Function 1.

Parameter	Value
D_{ped}	0.85
D_o	0.66
D_{wp}	0.2
R_g	10
R_{ped}	7
R_{so}	15
R_{vel1}	0.001
R_{vel2}	0.01
R_{wp}	0.3
$t_{reaction}$	0.8
$v_{reaction,max}$	0.0
w_1	4.5
w_2	5.5

Table 2: Parameter set 1 for Reward Function 2.

Parameter	Value
D_{ped}	0.85
D_o	0.66
D_{wp}	0.2
R_g	10
R_{ped}	7
R_{so}	15
R_{vel1}	0
R_{vel2}	0
R_{wp}	0.3
$t_{reaction}$	0.8
$v_{reaction,max}$	0.1
w_1	4.5
w_2	5.5

Table 3: Parameter set 2 for Reward Function 2.

Appendix: Neural Network architectures (1)

Layer	Type	Activation	Size	Filter Size	Filter Stride
1	Convolution	ReLu	32 Filter	$[5 \times 1]$	$[2 \times 0]$
2	Convolution	ReLu	32 Filter	$[3 \times 1]$	$[2 \times 0]$
3	Fully-Connected	ReLu	256 Neurons	-	-
4	Fully-Connected	ReLu	128 Neurons	-	-
5	Fully-Connected	Linear	Output Size	-	-

Table 4: 1D-Convolutional Neural Network

Layer	Type	Activation	Size	Filter Size	Filter Stride
1	Convolution	ReLu	32 Filter	$[8 \times 8]$	$[4 \times 4]$
2	Convolution	ReLu	64 Filter	$[4 \times 4]$	$[2 \times 2]$
3	Convolution	ReLu	64 Filter	$[3 \times 3]$	$[1 \times 1]$
4	Fully-Connected	ReLu	512 Neurons	-	-
5	Fully-Connected	Linear	Output Size	-	-

Table 5: 4-layered 2D-Convolutional Neural Network

Appendix: Neural Network architectures (2)

Layer	Type	Activation	Size	Filter Size	Filter Stride
1	Convolution	ReLu	64 Filter	$[8 \times 8]$	$[4 \times 4]$
2	Convolution	ReLu	64 Filter	$[4 \times 4]$	$[2 \times 2]$
3	Convolution	ReLu	32 Filter	$[3 \times 3]$	$[1 \times 1]$
4	Convolution	ReLu	32 Filter	$[2 \times 2]$	$[1 \times 1]$
5	Fully-Connected	ReLu	512 Neurons	-	-
6	Fully-Connected	ReLu	216 Neurons	-	-
7	Fully-Connected	Linear	Output Size	-	-

Table 6: 6-layered 2D-Convolutional Neural Network

[1] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," *Computing Research Repository (CoRR)*, vol. abs/1709.10082, 2017. [Online]. Available: http://arxiv.org/abs/1709.10082

[2] A. Folkers, "Steuerung eines autonomen Fahrzeugs durch Deep Reinforcement Learning," Master's thesis, University Bremen, Bremen, Germany, 2018.

[3] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," *Computing Research Repository (CoRR)*, vol. abs/1706.09829, 2017. [Online]. Available: http://arxiv.org/abs/1706.09829

[4] I. Ulrich and J. Borenstein, "Vfh*: Local obstacle avoidance with look-ahead verification," 2000.

[5] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Networks*, vol. 9, no. 5, pp.

1054–1054, 1998. [Online]. Available:
https://doi.org/10.1109/TNN.1998.712192

[6] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1889–1897. [Online]. Available:
http://proceedings.mlr.press/v37/schulman15.html

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *Computing Research Repository (CoRR)*, vol. abs/1707.06347, 2017. [Online]. Available:
http://arxiv.org/abs/1707.06347

[8] P. M. Dirk Helbing, "Social force model for pedestrian dynamics," 1998. [Online]. Available:
https://arxiv.org/abs/cond-mat/9805244

[9] B. Okal, T. Linder, D. Vasquez, and L. P. Sven Wehner, Omar Islas, “pedsim_ros,” https://github.com/srl-freiburg/pedsim_ros, 2018.

[10] Avidbots, “Flatland,” https://github.com/avidbots/flatland, 2018.

[11] A. Hill, A. Raffin, M. Ernestus, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “Stable baselines,” https://github.com/hill-a/stable-baselines, 2018.