Exposé

Author: Maximilian Schaller

University Leipzig - Faculty of Mathematics and Computer Science

Investigating the Feasibility of Training an Agent to Drive a Vehicle in a Simulated Environment Using Reinforcement Learning with Visual Camera Input

Contents

1 Motivation

Nowadays, machine learning (ML) occupies an indispensable place in the lives of all people and influences almost every situation in everyday life. One of the currently most popular areas within artificial intelligence (AI) research is to design fully trustful self-driving vehicles that can safely navigate through different environments to drive side-by-side with humans in everyday traffic. AI-optimized driving behavior would revolutionize transportation in the history of mankind as this change would make it possible to completely optimize traffic and make it more efficient, safe, and less polluting. As this is a very complex target with a bunch of challenging micro-tasks, there is still a lot of fundamental research to be done. A particular itemized challenge is to investigate the feasibility of different AI algorithms to drive vehicles using different inputs such as sensor or camera inputs. Since investigating and testing in the real world would be unfeasible expensive, in research, it is established to construct a dedicated simulation in order to perform certain experiments, to which this thesis aims to contribute.

2 Central Task And Research Goals

This thesis challenges one of those micro-tasks and investigates if reinforcement learning (RL) is suitable for training an agent to drive a vehicle in a simulated environment that contains generated red and blue gates that the vehicle has to pass by using visual inputs from a camera. The generated goals should result in a map similar to a ski slalom. The vehicle then has to drive from the beginning to the end of the map and must pass all of the generated gates in the correct order as fast as possible. Additionally, the car and the environment are modeled as close as possible to reality. As model serves a constructed arena which is located at the scads and as a vehicle a JetBot is used. However, all this ends up in a very complex challenge that the AI has to accomplish. To successfully design a reinforcement learning algorithm that is capable to train an agent to fulfill this task, the challenge has to be broken down into three major research questions which this thesis seeks to answer:

- How can the problem be modeled in the context of Reinforcement Learning? As described by the authors in [1], to fulfill this, it is crucial to represent all parts of the reinforcement learning cycle as shown in Figure 1. This includes the following parts:

  Environment The environment represents the surroundings, and, therewith the actual state, in which the agent is at a certain time point in the simulation. This could be i.e. the input of the camera, an input from the camera that has been pre-processed, like the coordinates of the obstacles, or additionally the vehicles rotation or position in the map.

  Agent In this case the agent will be the controller of the vehicle. Therewith the actions will probably be accelerating the left wheel and accelerating the right wheel. Thus, there will be two continuous actions.

Reward Modelling the reward is the biggest challenge to getting the RL algorithm working. This is a very fine-tuning part and could only be figured out during experimentation. Reasonable ideas are giving big long-term rewards for completing a map and some short-term rewards like rewards in form of the speed to push the vehicle to drive faster. Additionally, it should be considered to give punishments in form of negative rewards like a punishment for hitting an obstacle, being slow, or getting stuck on the map.

- How can the camera input be processed to feed it into the RL algorithm and achieve learning success?

  Description As explained above, the camera input will be an essential component of representing the environment in the reinforcement learning context. To ensure that the agent is able to learn in a reasonable amount of time it is probable that the camera input has to be processed to decrease the size of the state space. The smaller the state space is the less it has to be explored to find good actions by the agent and the probability for the agent to find good actions will rise.

- How suitable is Reinforcement Learning in solving the problem?

  Capability This thesis evaluates different configurations of the RL algorithm, i.e. different models of the RL environment or reward, and evaluates its performance on maps with different difficulties by using different evaluation parameters, as described in Section 5. The main objective here, is to find configurations where the agent is **capable** to maneuver the car completely through maps.

  Adaptability and robustness Additionally, this work seeks to determine how **adaptive** and **robust** the overall approach in solving the problem regarding different parkours and conditions such as light, arena settings or the vehicle.

In order to answer these questions a simulated environment is built to model the given context. Furthermore, a software architecture is designed that is capable of easily and quickly exchanging the algorithms, parameters, and surroundings to deliver reproducible results. In addition, the implemented system has to provide a possibility to evaluate, analyze and visualize the results of the approaches, i.e. in terms of training time, time for completing a track, or number of successfully passed a map. Finally, it is important that the trained agent can easily be exported to be used in a python script that can be executed on the JetBot to allow further experimentation in real-life. As a bonus, this thesis delivers an instruction on how to use the trained agent on the JetBot. Therefore, it is additionally important that the agent consumes a low amount of resources and is able to take its action in real time when it is used in inference.

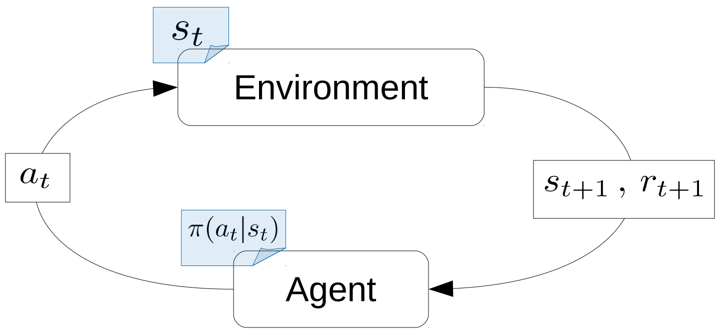

Fig. 1: Illustration of an RL Cycle at timestamp t. The agent takes an action a_t in state s_t by following the policy π. The environment responds by return the reward r_{t+1} corresponding to the action and the following state s_{t+1}

3 Related Work

As Reinforcement Learning has become one of the most popular domains in machine learning, there has already been done a lot of fundamental work and research that is related to this work. In [1] delivers a lot of basic theory about RL in general which can be applied to any kind of reinforcement learning problem. In [2] the authors did similar experimentation. They created a simulation and modeled a *Donkey Car*, which is a robot similar to the JetBot, but with a different motor engine principle. However, they were able to process the camera input by using convolutional neural networks to model the environment's state and then applied a deep reinforcement learning approach to train the agent to control the car in order to follow a simple street by accelerating and steering the donkey car. In [3] the authors created a simulation in unity and trained an agent to maneuver a vehicle on a street and additionally make it avoid obstacles which comes very close to the problem that is investigated in this thesis. They achieved create success by using Proxy Policy Optimization (PPO), which is considered as a state-of-the-art reinforcement learning algorithm as i.e. is explained in [4] by the OpenAI researchers. Thus, PPO seems to be a very promising approach that should be considered in the work of this thesis. In another work in [5] they trained an agent to control a vehicle to successfully switch lanes in an, in unity simulated traffic, so that the agent has to steer and accelerate a car and avoid a collision. For the training they used PPO and another promising reinforcement learning approach called Soft Actor-Critic (SAC), and compared them against each other. Both algorithms were able to successfully fulfill the control task and showed a fast learning success, whereby PPO was able to outperform the SAC approach in training time. Moreover, some relevant fundamental research on this topic has already been done in two previous bachelor theses. Where in one of the theses in a similar setting in a simulation in unity the camera input of a simulated vehicle was used and an evolutionary algorithm was trained to let the vehicle pass a map with obstacles. Here, the findings of the processing of the camera input are particularly interesting, whereby the library OpenCV[6] was used to extract the coordinates of the obstacles from the image and to continue working on them. The other thesis researched on getting the simulation to reality and constructed a robot to executed the trained evolutionary algorithm. This thesis provides me with information of the robot that is used to stick as close to reality as possible.

4 Methods

4.1 Simulation

To create a realistic simulation, Unity [7] is used as a state-of-the-art real-time development platform to create simulations for this kind of task. In order to create a proper simulation this thesis attempts to stick as closely as possible to reality. This is done by recreating a real existing arena, which is available in Scads for real-world experimentation. In addition, the simulated vehicle is based on a real constructed JetBot, and the entire environment including the arena, and all objects are given physical properties. To be able to reuse the results in reality, it is especially important to model the JetBot accurately. This will be done by using the exact same angle, field of view, position, and resolution of the camera as input like the JetBot. Furthermore, the same drive technology is used for the Jetbot, which means the vehicle in the simulation has a single motor for each of the two wheels that can accelerate between -1 and 1. Additionally, the wheels should have the same size and distance from each other. However, to keep the model as simple as possible, linear motors are used and additional factors and forces, such as friction or weight force can be disregarded as well. Furthermore, the gates consist of either two red or two blue posts, in the style of a ski slalom. The striking colors red and blue should ensure that the camera is able to detect the obstacles well. An example of how a map, that the vehicle has to pass, looks like, can be seen in Figure 2 . The environment should provide the possibility to easily generate different maps, i.e. different arrangements of the gates, to simplify the training process on the one hand and to prevent over-fitting to one map on the other hand. The agent should be able to cross every possible obstacle course of gates without errors at the end.

4.2 Model the Reinforcement Learning Cycle

To achieve the stated goals, this thesis starts by reviewing more literature about reinforcement learning, particularly in scenarios where there has been an accomplished success in certain control and driving tasks when using RL with visual inputs, to gather more confidence in the topic. Subsequently, several methods for processing the camera inputs and different RL algorithms are examined for their suitability in this environment. As explained in Section 3 the reinforcement learning approach PPO delivered promising results in similar research and

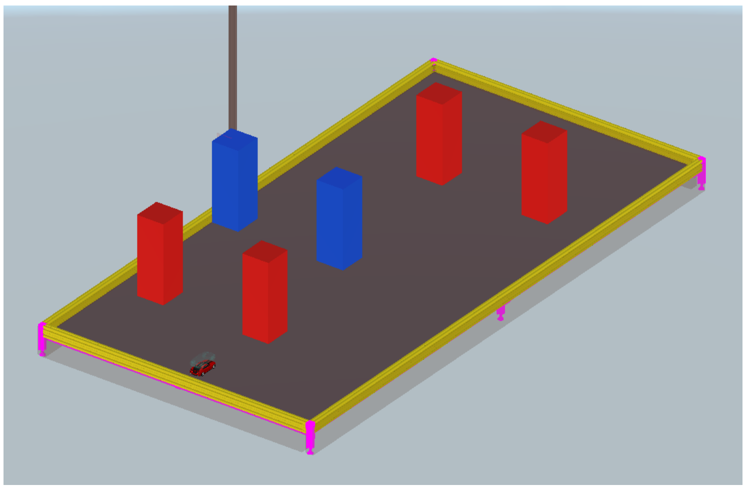

Fig. 2: Example of a generated map in Unity. The car should drive through the gates to the end of the map.

will be considered as a favorite choice, followed by SAC. Following, the goal is to apply the reinforcement learning cycle to the problem of this thesis. Thus, an environment is needed that provides a state and calculates a reward in each training iteration. This will be represented by the simulated environment, which has access to all important parameters like speed, steering angle, position, and camera picture of the vehicle, and therefore is capable of modeling the state in different ways. Additionally, it can track certain events like passing or missing a goal and finishing a map as well as it can observe the world time which is important for calculating the reward. To have fast access to all parameters of the simulated world and be able to fast adapt the state model or reward to giving problem is very important as it is very likely that during experimentation there will occur certain problems that will need to change the model. One problem could be that in certain positions, i.e. the vehicle drives by a goalpost and only sees one post anymore. This view of the vehicle can be seen in Figure 3. This might not be a problem as the RL algorithm might be able to solve this internally, but another solution could be to implement kind of a memory and let the agent i.e. remember the last three pictures that in this case, he knows that he just passed a post on the right.

In the work cited in Section 3 there are several ideas to model the reward and state in similar experiments. Those will be considered and a set of promising configurations will be evaluated in the experiments. Finally, the reinforcement learning agent is represented by the agent that is controlling the car and the actions that the agent can take will be cut down to accelerating the left and right wheel, which concludes in an action set of two continuous actions.

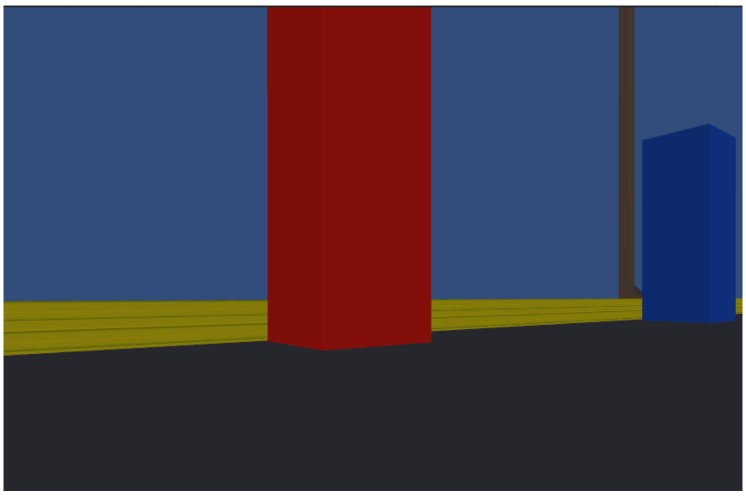

Fig. 3: View of the vehicle. You can see only one post of the goal that the vehicle has to pass. If the agent has only this picture as the input he doesn't probably know that he just has to turn right because in this view it is not apparent where the second post is

5 Experiments and Evaluation

5.1 Training

After examining various ideas, a bunch of configurations consisting of different models for the environment's state, the reward or AI algorithms, will be selected to be researched in the experimentation. An example configuration could be: using PPO as RL algorithm, modeling the raw picture of the vehicle as a state, and giving a reward of 1000 always when the agent successfully passes a map. The training should be done on different maps to prevent over-fitting i.e. by memorizing one map. By reviewing the training process, on the one hand by observing the behavior of the agents in the environment while training, and on the other hand by reviewing certain evaluation parameters like the increasing reward, it is possible to detect if the agent gets stuck in a local maximum. In this case, the training should be aborted and the configuration can be declared as unfeasible to solve the task. Following, every not aborted training results in a trained agent that would be more or less suitable to challenge the task.

5.2 Evaluation

The trained agents, that are representing the different configurations, are evaluated on a small set of constant, unchanging maps with different difficulties. To evaluate them, at least, the following parameters are measured for every agent on every map:

1. error rate of not completing the map by i.e. missing a goal or getting stuck on the map

2. average time they need to complete the map

3. frequency the vehicles hit obstacles

4. training iterations and training time

The error rate measures the percentage of how often an agent was able to pass a map of a certain difficulty. Therewith, it is possible to make a conclusion about which configurations can complete maps in general and are able to fulfill the task. Whereas measures 1 and 2 indicate how well the configurations performed if they completed a map. 4. shows how fast the training was. The training iterations specify the number of RL cycle iterations and are computing time independent, whereas the training time will be measured in seconds and indicate the time the algorithms took to learn its behavior.

Subsequently, the different configurations of the AI algorithms are compared against each other to figure out which approach is the best. Finally, based on the results a conclusion will be drawn that attempts to answer the research question stated at the beginning and to explain the results.

Finally, the configuration that showed the overall best performance is used and examined for its adaptability to new environmental conditions. Thus, the trained agent has to drive the vehicle through different parcours, in different light settings with different camera settings or changes to the vehicle itself like a changed acceleration or steering. Therefore, one of the degrees of freedom is changed at a time, while the others remain constant in order to determine degrees of freedom to which the configuration reacts particularly sensitively or with an unsuccessful driving behavior and which degrees of freedom have little influence on the JetBot. To stay within the scope of a master thesis, only a small set of extreme values of a degree of freedom is examined. In the case of light, for example, this could be 2 different positions, each with a bright and a dark light source.

6 Task Schedule

1. 1 month - Reviewing the literature and deciding for suitable RL algorithms and pre-processing methods that should be examined in this thesis.

2. 1 month - Design and implement the software for training and evaluation.

3. 1.5 month - Training and evaluation. (Experimenting)

4. 2.5 month - Writing the thesis.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[2] Q. Zhang, T. Du, and C. Tian, “Self-driving scale car trained by deep reinforcement learning,” *arXiv preprint arXiv:1909.03467*, 2019.

[3] R. Emuna, A. Borowsky, and A. Biess, “Deep reinforcement learning for human-like driving policies in collision avoidance tasks of self-driving cars,” *arXiv preprint arXiv:2006.04218*, 2020.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.

[5] A. J. M. Muzahid, S. F. Kamarulzaman, M. A. Rahman, and A. H. Alenezi, “Deep reinforcement learning-based driving strategy for avoidance of chain collisions and its safety efficiency analysis in autonomous vehicles,” *IEEE Access*, vol. 10, pp. 43303–43319, 2022.

[6] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[7] U. Technologies, “Unity.” ”https://unity.com”, 2023. Accessed: 2023-01-21.