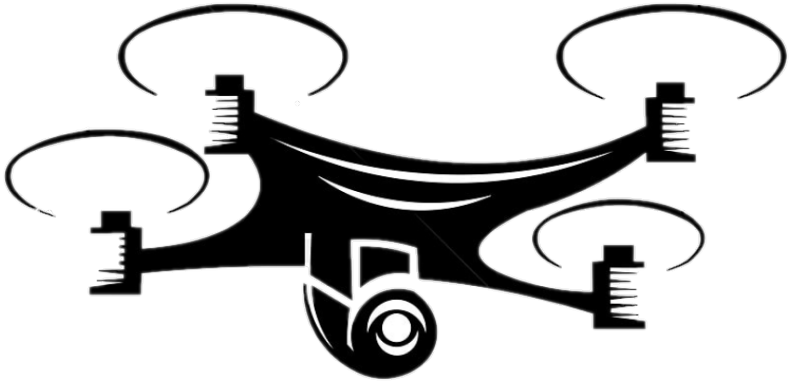
Studienarbeit
Documentation

Training a simulated drone with deep reinforcement learning

Semester: Fall 2023



Version: 1.0
Date: 2023-12-22 12:21:41Z
Git Version: Not available

Project Team: Andri Joos
Project Advisor: Marco Lehmann

School of Computer Science
OST Eastern Switzerland University of Applied Sciences

Contents

Part I

Abstract

Chapter 1

Abstract

This report details the comprehensive planning, execution, and evaluation of a solo-developed project in the field of artificial intelligence (AI) and software engineering. The project, focused on implementing a drone capable of autonomous navigation and delivery, follows the Agile Scrum methodology for project management. The report covers various aspects, including risk management, iterative development processes, and the application of the Soft Actor-Critic (SAC) algorithm for AI training.

A key aspect of the report is the detailed description of the SAC algorithm, which outlines the mathematical principles of this algorithm in an understandable way.

The development lifecycle is outlined through a series of sprints, each encompassing planning, review, and retrospective meetings. The report underscores the importance of risk management, categorizing and addressing potential challenges throughout the project.

Detailed insights into the project's structure, from backlogs to sprints, provide a transparent view of the iterative development process. The challenges faced, such as lack of experience with AI and Unreal Engine, are mitigated through proactive measures, including research and adaptation.

The long-term plan, presented in a Gantt chart, highlights key milestones, phases, and features. Primary and secondary features, along with epics and milestones, are meticulously defined, allowing for a clear understanding of project progress.

The report concludes with a thorough retrospective, highlighting successes, areas for improvement, and personal reflections. Key achievements include overcoming the hyperparameter problem and implementing the SAC algorithm to control the drone.

The project code can be found in this repository.

Part II

Management Summary

Chapter 2

Management Summary

2.1 Problem

As described in the initial project description, there are many systems out there, which are capable of solving much more complex tasks. However, these systems require specific hardware and complex algorithms and the work is done by specialized research teams.

This project explores the other side of the scale: How far can we get in a small project, using commodity hardware and building on an existing RL framework?

2.2 Techniques

As one of the first steps, the team needed to agree on some working methods known in software development. Since I already had experience in Scrum to iteratively create a working product, this was the selected short-term project management framework. To ensure that the project is completed in the given timeframe, a long-term management framework was needed. Since I already had experience with RUP was suggested, I used this framework.

To allow that several tasks can be processed simultaneously,

2.3 Results

The main result of this project two policies, capable of controlling a drone, specifically taking off, landing and navigating to a target point. These policies are described in detail in the policies section.

However, I have also created a framework to create policies for various other tasks. This is especially helpful, when the project will be developed further. This framework is described in detail in the environment section.

2.4 Outlook

Already during the planning of the project, I quickly realized that the project has great potential for many extensions. Some extensions I could already declare as secondary features, because I realized that the time during the semester will not be enough to implement these features to the full extent.

This project focused on the drone controlling in a small city. However, it can be easily extended to work world-wide (or even at greater scales), with a few adjustments to the environments.

Also, the framework written generalizes the problems, allowing to implement and solve much more complex tasks.

2.5 Conclusion

I have now spent a whole semester in this project, producing a good result, which I am very proud of. I learned a lot about AI, especially reinforcement learning and the Soft-Actor-Critic algorithm. It was very nice to finally use the theoretical stuff learned at the university in a productive environment and creating a product out of it.

One key point I will take away from this project is, how important it is to create a thorough plan at the beginning of the project. Also, the sprint reviews and retrospective were very useful to review the process in the project an mitigate new risks early, even though I was the sole team member.

Part III

Product Documentation

Chapter 3

Requirements

The requirements chapter in the documentation outlines the functional and non-functional requirements of the software solution, serving as a guide for development.

3.1 Functional Requirements

The functional requirements specify the specific features and functionality that the software solution must provide.

3.1.1 Actors

There is only a single actor in the system. This actor can be a company, a single person or everything in between. He wants to deliver goods from one place to another using this system and is called "deliverer".

3.1.2 User Stories

Actions a deliverer wants to perform are defined by the following User Stories. They are marked with US-X, where X is the number corresponding to the User Story.

All User Stories will be given a priority of development, as well as rough estimation of development effort needed. The Fibonacci scale is a common measure in the scrum process and reflects the relative effort of a task in the project (Story Points). Additionally, the MoSCoW method will be used to prioritize the User Stories.

US-1: Pickup

The deliverer wants the Drone to pick up the goods. Therefore the Drone needs to take off while holding a position.

Estimation 5

Prioritization Must have

US-2: Dropoff

The deliverer wants the Drone to drop off the goods. Therefore the Drone needs to land while holding a position.

Estimation 8

Prioritization Must have

US-3: Navigation

The deliverer wants the Drone to navigate to the dropoff location. Therefore the Drone needs to navigate to the dropoff location by herself.

Estimation 21

Prioritization Must have

US-4: Stop

The deliverer wants to be able to stop the Drone manually in case he feels something bad is about to happen.

Estimation 1

Prioritization Must have

3.2 Non-Functional Requirements

The non-functional requirements outline the characteristics and constraints that the software solution must satisfy, such as performance, security, and scalability.

3.2.1 Fulfillment-Check Procedure

Each NFR has a "Fulfillment Check" row which describes who has the responsibility to check if the NFR is fulfilled and how this is checked.

3.2.2 Performance

ID	NFR-1
Subject	Model action time
Priority	Medium
Measures	Model evaluation of an action must not take more than 100ms
Fulfillment check	A test that evaluates the execution time of the model

ID	NFR-2
Subject	Model size
Priority	Low
Measures	When the model is loaded, it must not be bigger than 1GB in the RAM
Fulfillment check	A test that loads the model and evaluates the size of the model

3.2.3 Compatibility

ID	NFR-3
Subject	Python version
Priority	Medium
Measures	• The code must support the Python version 3.10 and above • When the project is created, Python 3.10 is selected as Python version
Fulfillment check	The architect is responsible to check the Python version used after the project setup

ID	NFR-4
Subject	Tensorflow version
Priority	Medium
Measures	• The code must support the Tensorflow version 2.13 • When the project is created, Tensorflow 2.13 is installed
Fulfillment check	• The architect must check the requirements.txt if the Tensorflow version is set to 2.13 • The architect is responsible to check the Tensorflow version used after the project setup

3.2.4 Maintainability

ID	NFR-5
Subject	Modularity
Priority	High
Measures	• To ensure modularity, the division of Actor, Policy, Network and Environment must be respected • The TF-Agents standard implementations or derivations from them must be used
Fulfillment check	This requirement is given by using TF-Agents and needs therefore no further fulfillment check

Chapter 4

Architecture

This chapter explores the main structure of the project.

4.1 C4 diagrams

4.1.1 System Context diagram

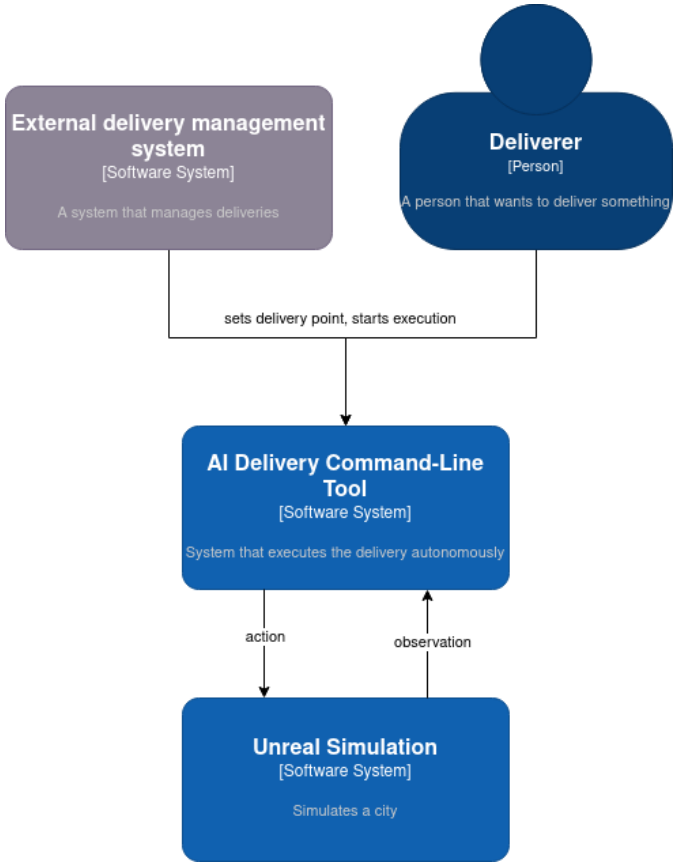


Figure 4.1: System Context diagram

4.1.2 Component diagram

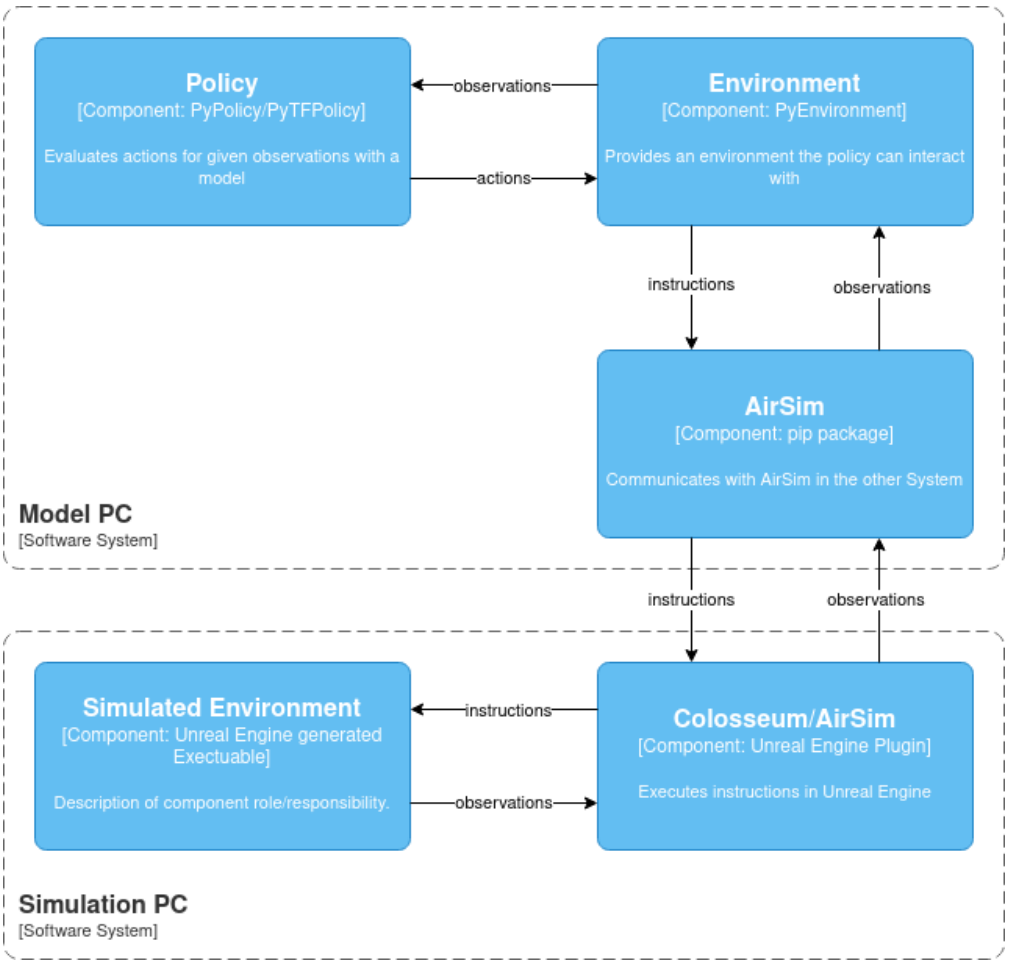


Figure 4.2: Component diagram

This physical separation of the Model PC and the Simulation PC allows multiple architectures in production. If desired, the Model can run at a centralized location, for example the headquarter, while the drone is moving. Then, only a reliable connection to the drone is needed, and the drone must be able to interpret the AirSim commands.

Another possibility is, that the drone completely maneuvers itself without any supervision from a centralized location. Therefore, the Model PC is also located in the drone, for example as a Raspberry mounted on the drone or embedded in the drone software.

Note, that the Model PC shows the simplified view of the evaluation process, as this is the process that runs in production. For training however, all of the components shown in Figure 4.3 are also part of the Model PC system.

4.2 Project architecture

4.2.1 TF agents

The intended architecture of TF agents is of course respected throughout the whole project and is as follows

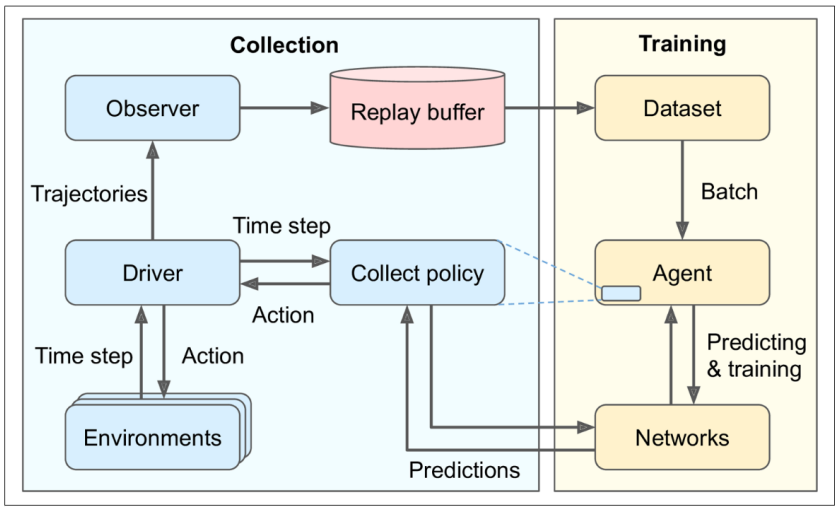


Figure 4.3: TF agents architecture [4]

In summary, this means, that the policy, regardless if it is the collect policy or evaluation policy, and the environment interact heavily. Each step, the observations from the environment are fed into the policy, which then evaluates the actions to take. These actions in turn are fed back into the environment and the process starts again.

Everything else in the diagram serves the purpose of gathering and storing data to train the model. The much simpler evaluation process therefore is as follows

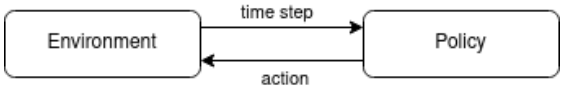


Figure 4.4: Evaluation process

4.2.2 Environments

Since the environments must be abstracted to avoid huge duplication of code, a BaseEnvironment was introduced. This environment provides the basic functionality to communicate with the AirSim API in the Unreal Engine environment.

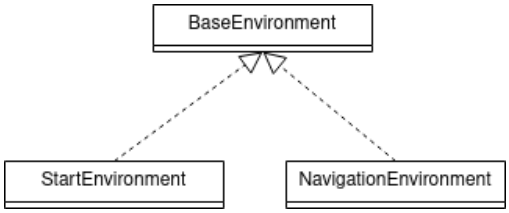


Figure 4.5: Environments hierarchy

4.2.3 Project structure

The structure from the simplified process is also shown in the project structure.

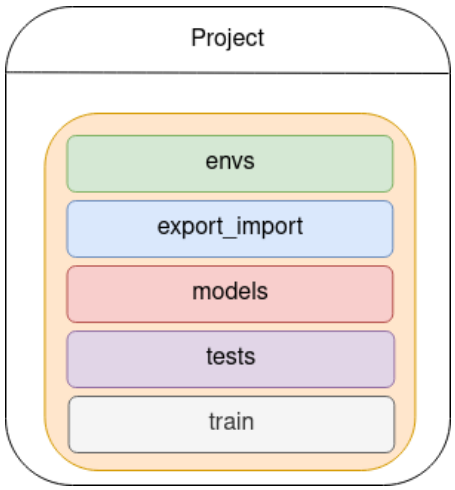


Figure 4.6: Project structure

- In the folder envs are all environments, as well as an abstract BaseEnvironment, from which all environments must derive
- For a simple export and import of the generated models, the folder export_import provides some scripts
- All models, alongside with checkpoints to continue training in the future if needed, are exported into the models folder
- The test folder contains the tests for the project
- In the train folder, the scripts for training the models are located

4.3 Toolstack

The following tools and patterns are employed and implemented in the software development process:

4.3.1 Development Process

The source code is centrally stored on a Gitlab server, which offers integrated issue management and merge request capabilities, streamlining the development process and allows for better teamwork. I chose Gitlab, because I am very familiar with the Gitlab ecosystem, since the used Gitlab server is a instance hosted by myself.

4.3.2 Simulation

Unreal Engine

To provide a simulated environment where the drone can safely be trained, Unreal Engine is used. Specifically, a slightly modified version of the City Sample environment is used. This environment has already built-in autonomous traffic and pedestrians, as well as many other features which makes it a very good simulation of a real city.

Colosseum/AirSim

Colosseum is a fork of Microsofts AirSim. It is installed into the Unreal environment so the drone can be controlled. Additionally, the drone is also provided by Colosseum.

4.3.3 Frameworks

TF Agents

TF Agents is the TensorFlow framework to train a model with reinforcement learning. It provides many standard and abstract implementations, which significantly reduce the effort to create a working model.

AirSim

AirSim also provides a pip package, which is needed to control the drone from the Python code.

Testing

The pip package pytest is used to execute the tests.

4.3.4 Documentation

The documentation is written in LaTeX, which makes collaboration and source control easy. Additionally, it allows to generate a pdf, which is the preferred format for reports.

Chapter 5

Soft Actor-Critic

This chapter describes the Soft Actor-Critic (SAC) algorithm in an understandable way.

5.1 Algorithm evaluation

Since there exist a vast amount of diffrent algorithms for reinforcement learning, each with its advantages and disadvantages, they had to be evaluated carefully.

5.1.1 On-policy

On-policy methods involve learning the policy based on the agent's current policy. The agent interacts with the environment using its current policy, collects experiences, and updates its policy based on these experiences. Therefore, the agent learns from its own experiences and updates the policy with experience collected with the same policy.

Advantages

- They often have good convergence properties and can be more stable in learning.
- They are well-suited for situations where you cannot collect a large amount of data from the environment.

Disadvantages

- They are generally sample-inefficient since the data used to update the policy is collected by the suboptimal policy before the update.
- They might struggle to explore the state space effectively because they are constrained by the current policy.

5.1.2 Off-policy

Off-policy methods decouple the data collection from the policy being learned. This means, that the policy is learned from data collected by another policy, which might be more exploratory and noisy.

Additionally, importance sampling is a key feature of off-policy algorithms. This estimates the reward based on the collected experience. A more detailed description can be found at the SAC critic network.

Advantages

- Often more sample efficient, since collected data can be reused even after a policy update.
- Often more stable in production, since more options are explored during data collection.

Disadvantages

- Can be more challenging to stabilize and require careful parameter tuning.
- Importance sampling introduces variance, which can make learning more unstable in practice.

5.1.3 On-policy vs. off-policy

Since the data collection is cheap in the simulated Unreal Engine environment and a stable production usage, where edge cases are covered better, is preferred, the off-policy approach was selected. This decision also implies, that the need of time for tuning the hyperparameters must be considered in the Sprint Planning.

5.1.4 Off-policy algorithm evaluation

There is still a huge number of off-policy algorithms and it would take too much resources to list all of them with their corresponding advantages and disadvantages. Of course, this evaluation was conducted carefully, by considering the advantages and disadvantages.

Only the algorithms A2C/A3C, SAC and TD3 will be evaluated.

Advantage Actor-Critic (A2C) & Asynchronous Advantage Actor-Critic (A3C)

A3C is basically A2C with the ability to train in parallel. This in turn also means, that there must exist multiple instances of the environment. As only one Unreal Engine environment instance can run at a time, training in parallel is of no use and A3C loses its advantage over A2C.

But there is also a second problem. A2C and A3C are designed to run at discrete action spaces, while the actions for the drone are continuous, which makes them unusable for this project.

Twin Delayed Deep Deterministic Policy Gradients (TD3) vs. Soft Actor-Critic(SAC)

The key difference between these two algorithms is, that TD3 uses a deterministic policy, while SAC uses a stochastic policy.

Deterministic Policy A deterministic policy is a rule that maps a specific state to a single, deterministic action.

$$\pi : \begin{cases} S \to A \\ s \mapsto a \end{cases}$$

where S is the set of possible states and A is the set of possible actions. This means, there is a clear one-to-one mapping between states and actions.

But there also lies the disadvantage of a deterministic policy. It doesn't perform well in uncertain environments, as it maps one input state to exactly one output state.

Stochastic Policy In contrast to a deterministic policy, a stochastic policy maps the input state to a a probability distribution over actions.

$$\pi : \begin{cases} S \times A \to [0,1] \\ a|s \in [0,1] \end{cases}$$

where S is the set of states and A is the set of possible actions.

Instead of $\pi(s) \mapsto a$, as the deterministic policy evaluates actions, the stochastic policy calculates $\pi(a|s) \in [0,1]$. In words, the stochastic policy calculates the probability of taking action a given state s.

This behaviour also allows to react better in uncertain environments.

TD3 vs. SAC Back to the TD3 and SAC evaluation. Because of the better handling of uncertain environments, I chose SAC in the end. This because it must be always considered, that this application may be used in a real-world scenario, even though not during this project. And the real world has many uncertain factors, such as sudden wind changes, air holes and many more. Therefore, the SAC was the better option.

5.2 Soft Actor-Critic

This section gives an in-depth view on how the SAC Agent works.

This section is heavily based on OpenAI's SAC explanation [3] and the official SAC paper [1].

5.2.1 The RL problem

In reinforcement learning, the objective is to choose a policy that maximizes the expected rewards when the agent follows that policy.

Probability distribution

In order to unserstand and calculate the expected reward, the probability distribution over trajectories must be clarified.

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \tag{5.1}$$

[2]

The notation $P(\tau|\pi)$ represents the probability of observing a trajectory τ given a policy π.

$\rho_0(s_0)$ is the probability of the agent being in state s_0 at the beginning of a trajectory. In most cases, the beginning of a trajectory is the start of an episode.

$P(s_{t+1}|s_t, a_t)$ calculates the probability landing in state s_{t+1} given the current state s_t as well as the action a_t that is going to be taken. This describes how the agent's actions affect state transitions.

$\pi(a_t|s_t)$ describes the probability of taking action a_t given the state s_t under the policy π. This is already known from the stochastic policy.

$\prod_{t=0}^{T-1}$ indicates, that the joint probability of the states and actions across all time steps is used.

Expected reward

The expected return $J(\pi)$ is then

$$J(\pi) = \int_{\tau} P(\tau|\pi)R(\tau) = \underset{\tau\sim\pi}{\mathrm{E}}[R(\tau)] \tag{5.2}$$

[2]

As already discussed, $P(\tau|\pi)$ is the probability distribution of observing a trajectory τ when following the policy π.

$R(\tau)$ is the cumulative reward the agent receives over all time steps.

$\underset{\tau\sim\pi}{\mathrm{E}}[R(\tau)]$ calculates the average total reward that the agent can expect to receive when following policy π over various possible trajectories. In other words, since each trajectory represents a unique sequence of states and actions, $\underset{\tau\sim\pi}{E}[R(\tau)]$ calculates the average total reward $R(\tau)$ obtained by the agent across all these possible trajectories that occur when the agent follows policy π.

Reward optimization

The optimal policy π^*, which gets the maximum reward, is therefore

$$\pi^* = \arg\max_{\pi} J(\pi) \tag{5.3}$$

[2]

In other words, the optimal policy π^* is the policy π, that maximizes the expected cumulative reward $J(\pi)$.

5.2.2 Entropy-regularized reinforcement learning

To understand SAC, we must first have a look at entropy-regularized reinforcement learning, as SAC is based on this principle.

The concept of entropy relates to the level of randomness encountered in a random variable. For instance, if a coin is biased towards landing on heads and rarely lands on tails, then it has low entropy. However, if a coin has an equal chance of landing on either side, its entropy is high.

Entropy

The entropy of a variable probability density function P of a variable x is calculated as follows

$$H(P) = \underset{x\sim P}{\mathrm{E}}[-\log P(x)] \tag{5.4}$$

[3]

where $\underset{x\sim P}{\mathrm{E}}$ represents the average over the random variable x sampled from the probability distribution P. This means, the average of $-\log P(x)$ is calculated over all possible values of x weighted by their respective probabilities as defined by $P(x)$.

The RL problem

"In entropy-regularized reinforcement learning, the agent gets a bonus reward at each time step proportional to the entropy of the policy at that timestep." [3]

The standard RL problem is changed accordingly

$$\pi^* = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(s_t,a_t,s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\right] \tag{5.5}$$

[3]

A lot of this is already explained in the standard RL problem and will not be explained again.

γ is a discount factor and therefore, γ^t is a discount factor that weighs future rewards. It ensures, that future rewards have less impact than immediate rewards.

$R(s_t, a_t, s_{t+1})$ represents the reward obtained when transitioning from s_t to s_{t+1} by taking action a_t

$\alpha H\left(\pi(\cdot|s_t)\right)$ is the entropy of the policy's action distribution given state s_t. The term α scales the influence of the entropy regularization in the overall objective. α determines the balance between the reward and the entropy.

Reasoning for including entropy When an agent receives a reward for being exploratory, the agent is encouraged to explore diffrent actions, which are not considered otherwise. Therefore, the policy becomes less deterministic, which is useful when the environment is complex, uncertain, or the best strategy has not been found yet.

Without entropy regularization, the policy can converge quickly to a deterministic policy, where the currently known high-reward actions will be used. This leads to high near-term rewards, but can hinder the agent's ability to react to changing conditions or discover more rewarding actions in the long run.

State-action function

The state-action function Q^π calculates the expected cumulative future reward, given the current state s and the current action a, when future actions follow a policy π.

$$Q^\pi(s,a) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1}) + \alpha\sum_{t=1}^{\infty}\gamma^t H\left(\pi(\cdot|s_t)\right)\middle| s_0 = s, a_0 = a\right] \tag{5.6}$$

This is very similar to the value function. The key diffrence however is, that it calculates the cumulative reward based on the current state s as well as the current action a.

Note, that the entropy is used from the second time step ($t = 1$) onwards, but not for the first time step $t = 0$. As OpenAI mentioned, this definition may vary in diffrent papers.

Value function

Value function are used to evaluate, which policy π of all policies is going to yield the highest reward and therefore is going to be used, given s as initial state s_0. In other words, it calculates which policy yields the highest cumulative reward starting at the current state s. Past rewards are therefore omitted.

The standard value function is changed to include the entropy.

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\Big(R(s_t,a_t,s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\,\middle|\, s_0 = s\right] \tag{5.7}$$

[3]

As most parts of this equation have already been covered in the entropy-based RL problem.

However, the last part $|s_0 = s$ means, that the value function is only calculated based on the current state s as initial state s_0. Of course, all following states have still an influence, as visible in the $\sum$, but the value function is not calculated for them.

Given the state-action function, we can now connect V^{π} and Q^{π}

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{a\sim\pi}\left[Q^{\pi}(s,a)\right] + \alpha H\left(\pi(\cdot|s)\right) \tag{5.8}$$

[3]

The last part of the equation $\alpha H\left(\pi(\cdot|s)\right)$ is needed, because in the state-action function Q^{π}, the entropy of the current time step is omitted.

Bellman equations

"The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next." [2]

Therefore, the standard Bellmann equations for V^{π} and Q^{π} are

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\substack{a\sim\pi\\ s'\sim P}}\left[R(s,a) + \gamma V^{\pi}(s')\right] \tag{5.9}$$

[2]

$$Q^{\pi}(s,a) = \mathop{\mathrm{E}}_{s'\sim P}\left[R(s,a) + \gamma \mathop{\mathrm{E}}_{a'\sim\pi}\left[Q^{\pi}(s',a')\right]\right] \tag{5.10}$$

[2]

Given the current state s as initial state (and a as initial action in the state-action function), the bellman equations just sum up future expected rewards iteratively and apply a discount factor γ for each iteration. Therefore, near-term rewards have a greater

influence than long-term rewards, since long-term rewards are reduced by applying the discount factor γ with every iteration.

$s' \sim P$ states, that the next state s' is drawn from the distribution P (the transition rules), which represents the probabilities of transitioning from state s to state s' by taking action a. This can be described as $s' \sim P(\cdot|s,a)$.

$a \sim \pi$ means that action a is chosen following the policy π. This can be described as $a \sim \pi(\cdot|s)$. The same applies for $a' \sim \pi$, which is shorthand for $a' \sim \pi(\cdot|s')$.

With this information, we can now extend the bellman equation for Q^π to fit the previously discussed state-action function.

$$\begin{aligned} Q^\pi(s,a) &= \underset{\substack{a' \sim \pi \\ s' \sim P}}{\mathrm{E}} \left[R(s,a,s') + \gamma (Q^\pi(s',a') + \alpha H(\pi(\cdot|s'))) \right] \\ &= \underset{s' \sim P}{\mathrm{E}} \left[R(s,a,s') + \gamma V^\pi(s') \right] \end{aligned} \tag{5.11}$$

[2]

5.2.3 Soft-Actor Critic

SAC learns a policy π_θ as well as two Q-networks Q_{ϕ_1} and Q_{ϕ_2}. Additionally, the entropy regularization coefficient α is learned.

Reasoning for two Q-networks

Q-learning algorithms in RL tend to overestimate the Q-values. This overestimation can result in suboptimal and unstable policies, because the agent might become to optimistic.

Therefore, the loss is calculated based on the minimum of the two Q-network.

Loss

SAC uses mean squared bellman error (MSBE).

$$L(\phi_i, \mathcal{D}) = \underset{(s,a,r,s',d) \sim \mathcal{D}}{\mathrm{E}} \left[\left(Q_{\phi_i}(s,a) - y(r,s',d) \right)^2 \right] \tag{5.12}$$

[3]

$\mathcal{D}$ is the gathered experience in the replay buffer.

$Q_{\phi_i}(s,a)$ refers to the Q-value predicted by the Q-network with the parameters ϕ_i

$L(\phi_i, \mathcal{D})$ measures the error between the Q-values predicted by the Q-network $Q_{\phi_i}(s,a)$ and the target values $y(r,s',d)$.

The target y is given by

$$y(r, s', d) = r + \gamma(1 - d) \left(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s') \quad (5.13)$$

[3]

r represents the immediate reward receiving when transitioning to state s'.

d is called the termination indicator. It is 1 if the episode terminates after the current transition. This ensures, that future rewards are considered only when the episode is ongoing ($d = 0$)

$\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s', \tilde{a}')$ calculates the minimum Q-value of the two Q-networks. As discussed before, the minimum is taken since the agent may become to optimistic. This is called the clipped double-Q trick.

It is worth mentioning, that variables with a tilde, such as $\tilde{a}$, are sampled from the policy while variables without a tilde, such as s' are retrieved from the replay buffer.

Learning the policy

In each state, the policy should take actions that aim to maximize the expected future reward plus the expected future entropy. Therefore, it should maximize V^π.

This is achieved by employing the reparametrization trick. The main idea behind the reparametrization trick is to separate the randomness in the stochastic part of a model from the deterministic part, allowing backpropagation of gradients through stochastic layers and therefore for gradient-based optimization.

As a squashed Gaussian policy is the most common type of action selection process and it's used in the OpenAI paper as well as in the SAC paper, this will be discussed.

$$\tilde{a}_\theta(s, \xi) = \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \xi\right), \quad \xi \sim \mathcal{N}(0, I). \quad (5.14)$$

[3]

$\mu_\theta(s)$ denotes the mean of the action distribution. It is determined by the policy's network parameterized by θ and is dependent on the current state s. This is the deterministic part of the function.

$\sigma_\theta(s)$ represents the standard deviation of the action distribution. Like μ, it is determined by the policy's network parameterized by θ and is dependent on the current state s.

ξ is random noise sampled from a standard multivariate Gaussian distribution with a mean of 0 and identity covariance matrix I. This distribution is basically the standard normal distribution with multiple variables.

tanh is the squashing function that squashes the values into the range $[-1, 1]$.

$\tilde{a}_\theta(s, \xi)$ represents the stochastic action, which is produced based on the policy parameterized by θ. It is generated by taking the deterministic part $\mu_\theta(s)$, adding some noise ξ, which is scaled by $\sigma_\theta(s)$. This value is then finally squashed by tanh.

The reparametrization trick allows to rewrite the expectation over actions into an expectation over noise, which doesn't depend on the policy parameters anymore. As pointed out by OpenAI, this removes the pain point, that the expectation distribution depends on the policy parameters.

$$\underset{a\sim\pi_\theta}{\mathrm{E}}[Q^{\pi_\theta}(s,a) - \alpha \log \pi_\theta(a|s)] = \underset{\xi\sim\mathcal{N}}{\mathrm{E}}[Q^{\pi_\theta}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s)] \tag{5.15}$$

[3]

The same can also be applied to the loss function.

$$\max_\theta \underset{\substack{s\sim\mathcal{D}\\ \xi\sim\mathcal{N}}}{\mathrm{E}}\left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right], \tag{5.16}$$

[3]

5.2.4 SAC components

Since the matemathical background is now known, we can dive into the components of SAC.

Actor network

The actor network in SAC is responsible for determining the policy for selecting actions in different states. It outputs a probability distribution over the action space, which is μ and σ for a gaussian distribution.

The actor network therefore learns the policy (strategy) π.

Critic network

The critic network, on the other hand, is responsible for estimating the value of states or state-action pairs. It provides a measure of the expected return for being in a particular state and taking a particular action.

The critic network therefore learns the Q function.

Networks and target networks

SAC and many other algorithms have use diffrent networks to learn (just called network) and to make predictions (target network). The changes in the network slowly

get forwarded to the target network by using the parameters `target_update_tau` τ and `target_update_period`. This ensures stability during training, as the target network doesn't vary as much as the network, which gets trained.

$$\theta_{\text{target}} \leftarrow \tau \cdot \theta_{\text{learned}} + (1 - \tau) \cdot \theta_{\text{target}} \tag{5.17}$$

The `target_update_period` specifies the interval of doing this update.

Chapter 6

Implementation

This chapter describes how certain key technologies are implemented and used.

6.1 Environment

The environments provide a system for a task, which the agent tries to optimize. For this, the observation and reward have an important role, as the agent gets the observation as input and tries to optimize the reward.

6.1.1 BaseEnvironment

As mentioned in the architecture of the environments, the BaseEnvironment provides the basic functionality needed in all of the environments used to train the agent. Most important, it handles the controls of the drone with the AirSim function moveByRollPitchYawrateThrottleAsync.

This function was chosen, because it takes roll, pitch, yaw rate and throttle as inputs, which is basically what a remote controller emits.

6.1.2 StartEnvironment

Actions

The StartEnvironment only takes a single action, because it only needs to move on the z-axis (up and down). Therefore, only the throttle needs to be learned.

Observations

First, there is the difference between the target height and the actual height of the drone. A negative height delta means, that the drone is currently below the target height, a positive delta means it is above.

The z delta as sole observation caused the drone to oscillate heavily around the optimal point on the z axis.

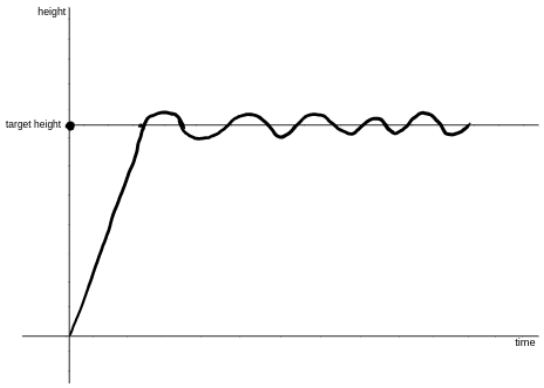


Figure 6.1: Oscillations

These oscillations happened, because the agent tries to get to the target height as fast as possible, which means full throttle. As soon as the target height is reached, the throttle is reduced to it's minimum, but since it has already gained quite a lot momentum, it overshoots the target height. Since the drone is now above the target height the throttle is not activated until it reaches the target height again. Now, full throttle is applied, but the drone has gained momentum again, which causes the drone to fall below the target height. Then the process repeats itself. To counteract this, the velocity of the drone on the z-axis is emited in the observation.

Rewards

The reward is basically the negative absolute height delta. The negation is needed, so the drone tries to go to the target height as fast as possible.

To smoothen the reward function, the height delta is squared.

6.1.3 NavigationEnvironment

Actions

This environment takes 3 actions, the throttle, the yaw rate, and the pitch, allowing the drone to move on all axes.

Observations

Firstly, there are is a delta on each axis in the observations, as well as the current speed on each axis. Additionally, there is also a heading delta, where 0 means the optimal heading and π means the drone facing the wrong way.

In total, there are therefore seven observation values.

Reward

The reward is put together from diffrent components.

As in the StartEnvironment, the delta to the target point is given as a negative reward. The delta can be calculated with the Pythagorean Theorem.

Additionally, the heading delta is added as negative reward to force the drone looking forward while navigating. Without this reward, the drone would be also allowed to reach the target facing the wrong direction of even sideways.

I have also experimented with implementing negative rewards for direction changes, forcing the drone to take smooth actions. This however seemed to have a huge negative impact, as the drone only circled around the starting point. Therefore, this reward is not used.

6.2 Policies

To each environment, there exists a corresponding policy.

6.2.1 Start policy

The start policy handles the takeoff of the drone. However, since the takeoff and landing are very similar tasks and the start policy generalizes quite good, the start policy can also be used for the landing task.

Hyperparameters

```
num_episodes = 10000 # ensures training isn't stopped too early
num_steps_per_episode = 2000

# ensures the policy reaching its target is in the replay
# buffer, 10000 was too little, as only the last 5 episodes
# were kept in the replay buffer
replay_buffer_capacity = 100000

# simple tasks with large learning rate require large
# batch size to prevent loss getting 0
batch_size = 1024

# largest possible value
# larger learning rate causes policy to not solve task
critic_learning_rate = 3e-4
actor_learning_rate = 3e-4
```

```
alpha_learning_rate = 3e-4

# simple task -> discount can be quite large / gamma quite low
# fine-adjusted through trial and error
gamma = 0.9

# simple task -> few neurons
# z velocity requires a second layer
actor_fc_layer_params = (16,16)
critic_joint_fc_layer_params = actor_fc_layer_params


# the following hyperparameters are from utils.py

target_update_tau = 0.005
target_update_period = 1
```

The parameters `target_update_tau` and `target_update_period` are explained in section 5.2.4. They only affect the critic network.

Learning curve

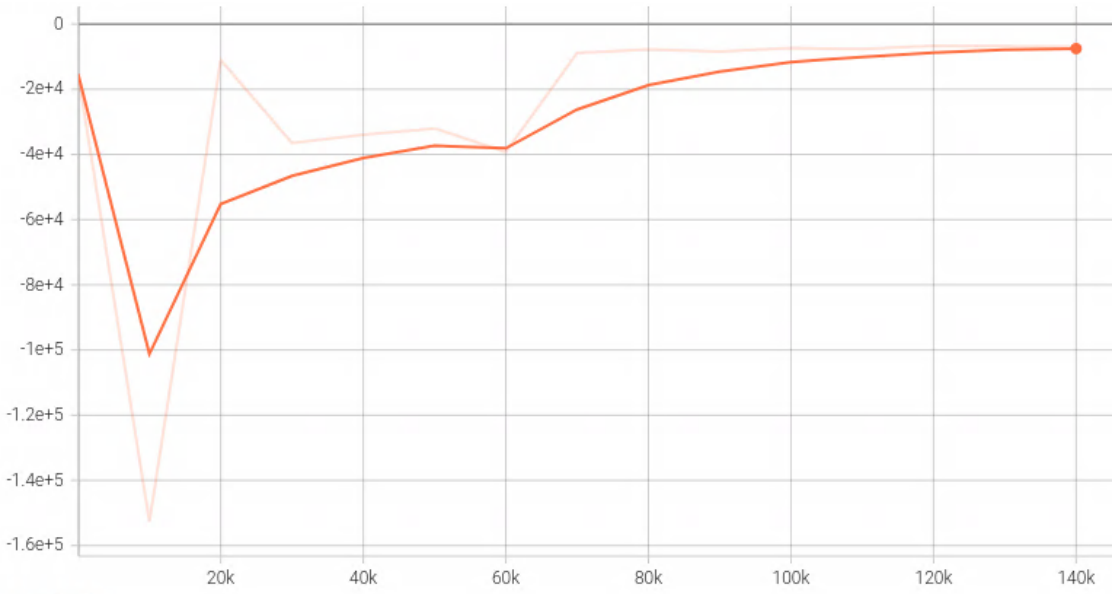


Figure 6.2: Learning curve start policy

In the opaque, smoothened curve, is is clearly visible how the policy approaches the optimum, which is approximately at $-1*10^4$. The partially transparent, not smoothened

curve shows even shows good results at 70K steps, but, while testing the policy, I have noted, that the policy is still not good in some cases. Therefore, the smoothened curve is a better indicator for the policy successfully solving the task.

Demo

A demonstration video of the drone starting is available here. A demonstration video of the drone landing is available here.

6.2.2 Navigation policy

The navigation handles, as the name suggests, the drone while navigation to a target point.

Hyperparameters

```
# navigation training has two training loops to speed up
# training, one parameter set for each
initial_train_episodes = 200
initial_train_steps_per_episode = 1000
initial_eval_steps = 2000

train_episodes = 10000
train_steps_per_episode = 4000
eval_steps = 6000
# replay buffer already has samples, therefore no initial
# samples needed
train_initial_steps = 0

# taken from start policy training
replay_buffer_capacity = 100000

# taken from start policy training, same reasoning
batch_size = 1024

# taken from start policy training, then fine-adjusted
# through trial and error
critic_learning_rate = 1e-4
actor_learning_rate = 1e-4
alpha_learning_rate = 1e-4

# taken from start training, as it is still a relatively
# simple task
gamma = 0.9
```

```
# relatively simple task -> few neurons
# task is more difficult than start -> more neurons needed
# than for start policy.
# I expected also to need more layers, as there are more
# observation and they are not as obviously connected as
# with the start policy. While reducing the number of layers
# it has turned out, that two layers are sufficient.
actor_fc_layer_params = (64,64)
critic_joint_fc_layer_params = actor_fc_layer_params


# the following hyperparameters are from utils.py

target_update_tau = 0.005
target_update_period = 1
```

The parameters `target_update_tau` and `target_update_period` are explained in section 5.2.4. They only affect the critic network.

Learning curve

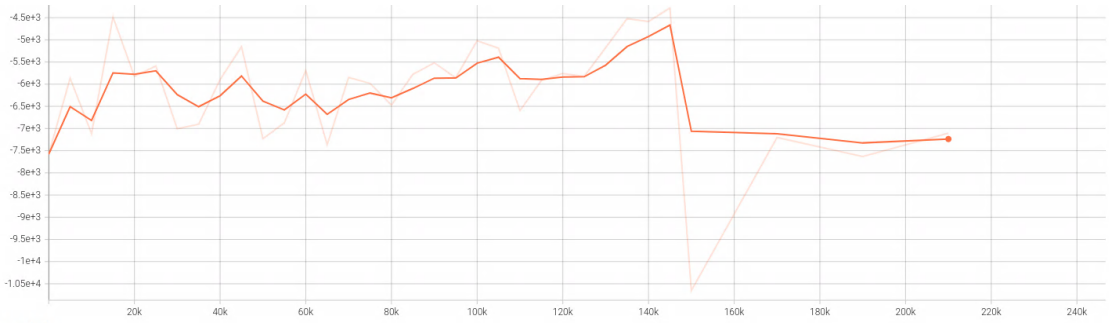


Figure 6.3: Learning curve navigation policy

As visible in the image, there is a huge reward drop after 150K steps. This, because in the first 150K steps, the episode length is very small to speed up training. After the reward is stabilized again, the policy is successfully trained.

Demo

A demonstration video is available here.

Chapter 7

Quality Measures

Quality measures are critical indicators used to evaluate the effectiveness and efficiency of software development, encompassing various metrics that assess the software's functionality, performance, security, maintainability, and other key factors.

7.1 Guidelines

The guidelines ensure the readability of the code as well as clean code.

7.1.1 Latex formatting

I did not use any official latex formatting rules. However, I used my own latex formatting rules to ensure consistent formatting throughout the document, as well as an enhanced readability of the latex content. These rules are:

- Every sentence must be on a new line
- Each line must be 158 columns or less, excluded is text in itemize, enumerate and lstlistings
- The content belonging to the title must be one intendation more than the title itself
- For intendation, spaces are used, not tabs
- One intendation is 4 spaces
- Before a title, there must be an empty line
- Each file ends with an empty line
- Subitems are created with an additional itemize
- The itemize for the subitems must use one intendation more than the corresponding item

- If a file gets too big ($\geq$ 50 lines), it may be worth to outsource and break down the file content into multiple files

7.1.2 Code

The official Python coding guideline is used.

7.1.3 Definition of Done

It is important to define, when a task is truly complete.

- **Acceptance criteria**
  - All defined tasks of the issue have been solved
  - Complies with functional and non-functional requirements
  - The feature is ready to demonstrate
  - Code review by myself
- **Quality**
  - Complies with defined coding standards
- **Documentation**
  - All necessary items have been documented

7.1.4 Git

To allow a clean and fast workflow, some guidelines regarding git are defined.

Branching

Each task must have its own branch, allowing for later traceability of the changes. Each person has its own branch, even if they are working on the same task.

Additionally, it is forbidden to push to the master directly. This is also enforced by the Gitlab policy, making it impossoble to push to the master directly.

Merges

To minimize errors be merged into the master, a merge request must be opened and the code must be carefully examined. Also, the code must be checked if complies with the guidelines.

7.1.5 Sprint retrospective

To reevaluate quality of the project each sprint is ended with a review meeting, which also contains the retrospective.

7.2 Test strategy

7.2.1 Test framework

As it is a de facto standard in python, pytest is used.

To structure the tests, the commonly used AAA pattern is employed.

- **Arrange**: During the arrange phase, the test prepares the necessary prerequisites for the test. This involves creating any required objects, initializing variables, and configuring any dependencies that the code under test relies on.
- **Act**: During the act phase, the test executes the action being tested. This could involve calling a specific method or interacting with an object.
- **Assert**: In this phase, the test verifies that the action performed in the Act phase has produced the expected result. This involves comparing the actual result of the action with the expected result and failing the test if the two do not match.

7.2.2 Acceptance tests

The generated models must checked with various target and starting points, ensuring the ability to generalize and producing an acceptable output, regardless of the starting and end point.

7.2.3 Unit tests

Unit tests are not yet implemented, due to my initial thought, that there is not much to check. At the end of the project, when it became obvious, that unit tests are also necessary in this project, there was not enough time left to implement them. However, I strongly suggest implementing unit tests to check the environments when further developing the product.

Part IV

Project Documentation

Chapter 8

Initial Project Description

8.1 Problem

Controlling a first-person view drone (fpv drone) is a difficult task. It has been demonstrated that deep reinforcement learning algorithms are capable of learning this control task. Current research produces fascinating results, but achieving those results requires specific hardware and complex algorithms. Furthermore the work is often done by specialized research teams.

Here we explore the other end of the scale: How far can we get in a small project, using commodity hardware and building on an existing RL framework?

8.2 Task

The goal is to train a simulated drone. The drone should be able to start, land, and fly to a predefined location. A particular aspect of this project is to scale-down the problem of "Learning to Fly". End-to-End RL needs many iterations and is difficult to tune. Here we explore the options to simplify the task.

8.3 Scope and form of the expected results

The main deliverables are listed in chapter 5.5 "Leitfaden BA/SA".

Additional project specific deliverables:

- Description of the training pipeline: hardware, simulator, RL algorithms, system architecture
- Demonstration of the achieved results: screenshots or video
- Quantitative description of the learning process (e.g. graphs showing reward vs. epoch)

- The final configuration (tools, algorithms, hyperparameters, etc.) should be described such that an interested person can reproduce the results in less than two days.

8.4 SA Team

- **Student**: Andri Joos
- **Advisor**: Marco Lehmann
- **Internal Co-Examiner**: not required
- **External Expert**: not required

8.5 Dates

- **Start Date**: Monday September 18$^{\text{th}}$ 2023
- **Submission Date (Report)**: Friday, December 22$^{\text{nd}}$ 2023, 5pm
- **Intermediate Presentation**: not required
- **Final Presentation**: not required

Chapter 9

Project Plan

The project plan outlines the resources, roles and processes required for the successful completion. It goes into detail which risks I expect and how I have laid out the long term plan.

9.1 Resources

9.1.1 People

- Andri Joos

I am studying Computer Engineering with Data Science & Machine Intelligence as field of study at the OST University of Applied Sciences. This project is carried out within the scope of the course Studienarbeit.

I have basic knowledge of Artificial Intelligence, as it was taught in the courses AI Foundations and AI Applications.

Additionally, I have good knowledge about Git and source control in general. I have gathered some experience in DevOps in a business environment, as well as in many private projects.

Also, I was part of a Scrum team in a business environment as well as in the course Software Engineering Project. In this course, I have gathered also the necessary understanding of project management, project documentation and project execution.

9.1.2 Time

The project started with the kickoff meeting on September 19th 2023, 13:00 and will end on December 22nd 2023, 17:00 with the final submission of the report. This, alongside with the later described milestones, are deadlines which have to be met. For the whole project, 240 hours per person is required. Since the project must be finished by December

22nd 2023, these hours must be met in 14 weeks, which results in approximately 17 hours a week per person.

9.1.3 Cost

As this is a school project, I do not have a budget which can be expressed in money. However, our costs can be expressed as the earlier mentioned 240 hours, that I can use for this project.

Additionally since I am doing a semester abroad while doing this project and I need my PCs, I shipped them to the remote University. These expenses however will be covered by myself.

9.1.4 Tooling

The used IDE for the project and the documentation is Visual Studio Code, as it's free, open source and covers all needs.

Unreal Engine 5 is be used for the simulated environment. The API of Colosseum is used to interact with the simulated environment. This is a fork of Microsofts AirSim.

As version control, Git is used. As UI for Git, gitextensions is used. The server to store the source code and the documentation is a selfhosted instance of Gitlab reachable at `https://git.420joos.dev/`. The Continuous Integration of this server, as well as the issue tracking system of the before mentioned server will be used to keep track of the work and the time. To collect and evaluate the time spent, gtt is used.

The documentation is written in LaTeX.

Microsoft Teams is used for the communication with the Advisor.

9.2 Roles

Since there is only one team member, all of the following roles will be executed by Andri Joos. He is responsible to carry out the project successfully.

- Project Manager
- Developer
- Product Owner (in agreement with the project advisor)
- DevOps-Engineer
- Software Architect

9.3 Processes & Meetings

To achieve an agile project I use Scrum to define processes in this project.

9.3.1 Backlogs

Product Backlog

The Product Backlog is realized in Excel.

Sprint Backlog

The Sprint Backlog is realized in GitLab. Product Backlog items will be refined and transformed into GitLab Issues during Planning.

9.3.2 Sprint

Sprint information:

- Sprint duration: 2 weeks
- Sprint start: Tuesday
- Total amount of Sprints: 7

Planning

Each Sprint is started by a planning meeting.

Planning procedure:

1. Specify what should be achieved during this Sprint
2. Evaluate which Product Backlog items should be put into the Sprint Backlog

Review

Each Sprint end is initiated by a Review. Since it doesn't make sense for such a small team to also do a separate Retrospective meeting, I will do both, the Review and the Retrospective, in one single Meeting.

Review procedure:

1. Progress evaluation of the project
2. Adjust long term plan

Retrospective procedure:

1. Documentation Quality
    - Check if documentation if properly formatted
    - Check for grammatical errors
    - Check if anything is missing or should be improved

2. Time Tracking Quality
    - Check if the estimations for the tasks are good enough
3. Risk Reevaluation
    - Reevaluation of all current risks
    - Check for any new risks
4. Code Quality
    - Check for bad code
5. Git Quality
    - Check for leftover branches
    - Check for open merge requests

Stakeholder meeting

This meeting is about the current state of the project and to discuss any problems. The meeting is attended by the project advisor and myself.

9.4 Risk Management

Risk management is a critical process that involves identifying potential risks and developing strategies to mitigate or eliminate them.

9.4.1 Risk categorization method

To rate the risks, they are categorized and assigned a value using the risk matrix below. The respective values can be found right next to the title of the risk, likelihood will be referred to as "L" and consequence as "C"

		Consequence				
		Negligible 1	Minor 2	Moderate 3	Major 4	Catastrophic 5
Likelihood	5 Almost certain	Moderate 5	High 10	Extreme 15	Extreme 20	Extreme 25
	4 Likely	Moderate 4	High 8	High 12	Extreme 16	Extreme 20
	3 Possible	Low 3	Moderate 6	High 9	High 12	Extreme 15
	2 Unlikely	Low 2	Moderate 4	Moderate 6	High 8	High 10
	1 Rare	Low 1	Low 2	Low 3	Moderate 4	Moderate 5

Figure 9.1: Risk Matrix

9.4.2 Lack of experience with AI |10 (L: 2, C: 5)

Description

Since I have never done a big project in the field of artificial intelligence, it is possible that I don't have needed experience for this project.

Mitigation

When something is unclear, I will try to figure it out by doing research in the first place. If it's still not clear, the project advisor will be contacted.

9.4.3 Lack of experience with Unreal Engine |6 (L: 2, C: 3)

Description

I have never used Unreal Engine before this project.

Mitigation

As mitigation before the project started, I did some research and tried some Plugins to see if this project is feasible.

9.4.4 AI training needs too much time |9 (L: 3, C: 3)

Description

The AI model has to get trained in order to produce acceptable results. This process however can take much time depending on the task it has to solve.

Mitigation

I have two powerful PCs, one to train the AI equipped with a RTX 3090 and the other to run the simulation equipped with a RTX 3060 and a AMD Ryzen 7 5800X.

9.4.5 Hyperparameter tuning takes too much time |6 (L: 2, C: 3)

Description

Since the chosen Agent has an off-policy algorithm, the hyperparameters must be chose carefully.

Mitigation

There are no mitigation strategies. Of course, some kind of automation, such as grid and random hyperparameter evaluation, can be employed, but in the end, it will still take some time and manual tuning to figure out good hyperparameters.

This needs to be considered in the Sprint planning.

9.4.6 Gitlab failure |2 (L: 1, C: 2)

Description

Since a selfhosted Gitlab is used, there is a chance that the server is unavailable for some reason.

Mitigation

The server runs on a Kubernetes cluster and therefore should be restarted as soon as a crash happens. Additionally, accessing the network of the server remotely is also possible in case the server needs to be restarted manually.

9.4.7 Specification of requirements/features is inaccurate |9 (L: 3, C: 3)

Description

At the start of the project it's hard to perfectly define all requirements as it's unclear how the application will look like in the end.

Mitigation

I am using the agile methodology Scrum to assure that the requirements will be regularly adjusted in the sprint planning and sprint review meetings.

9.4.8 Time management (Not enough time) |12 (L: 3, C: 4)

Description

Due to many other assignments that have to be accomplished, it is possible that I am unable to invest the required time each sprint.

Mitigation

At the sprint planning, this must be noted. The sprint planning is done accordingly.

9.4.9 Changes History

Risk	Value change	Reason for change	Date
AI training needs too much time	9 → 6	The experienced training times were very low	17.10.2023
Time management (Not enough time)	12 → 8	The progress is more than expected	17.10.2023
AI training needs too much time	6 → 9	Training with controller inputs instead of velocity takes more time	31.10.2023
Hyperparameter tuning takes too much time	→ 9	Off-policy algorithm needs accurate hyperparameters	31.10.2023
Lack of experience with AI	15 → 10	much experience gathered already	31.10.2023
Lack of experience with Unreal Engine	10 → 6	Unreal environment runs fine, no big changes expected anymore	14.11.2023
Hyperparameter tuning takes too much time	9 → 6	Base hyperparameters have been established, only fine-tuning needed	14.11.2023
Time management (Not enough time)	9 → 6	Hyperparameter problem delayed implementation of navigation	14.11.2023

9.5 Long-term Plan

All the listed items are presented in a Gantt chart. This is attached at the end of this section.

9.5.1 Phases

I will be working according to the Rational Unified Process (RUP). This includes following phases:

- Inception (19.09.2023 - 03.10.2023)
- Elaboration (03.10.2023 - 17.10.2023)
- Construction (17.09.2023 - 05.12.2023)
- Transition (05.12.2023 - 22.12.2023)

9.5.2 Features

Primary Features

Features which are guaranteed to be implemented in the available time of the project.

- Drone starts by itself
- Drone lands by itself
- Drone can fly a short distance from a single point to another (without obsticals)

Secondary Features

Bonus features which will be implemented when there is some time left.

- Drone detects obstructions
- Drone takes obstructions into consideration
- Done safely stops if there is an obstical in it's way, that cannot be bypassed
- Drone can fly from any point to any other point
- Asynchronity, so call to drone is non blocking (involves predicting the position when taking the next action)

9.5.3 Epics

There is only a single epic, since the the actor only wants to execute a single action.

- Deliver goods

9.5.4 Milestones

- M1 - Initial Project setup (03.10.2023)
  - Creation of project documentation
  - Setup of build pipelines for project documentation
  - Setup of time tracking

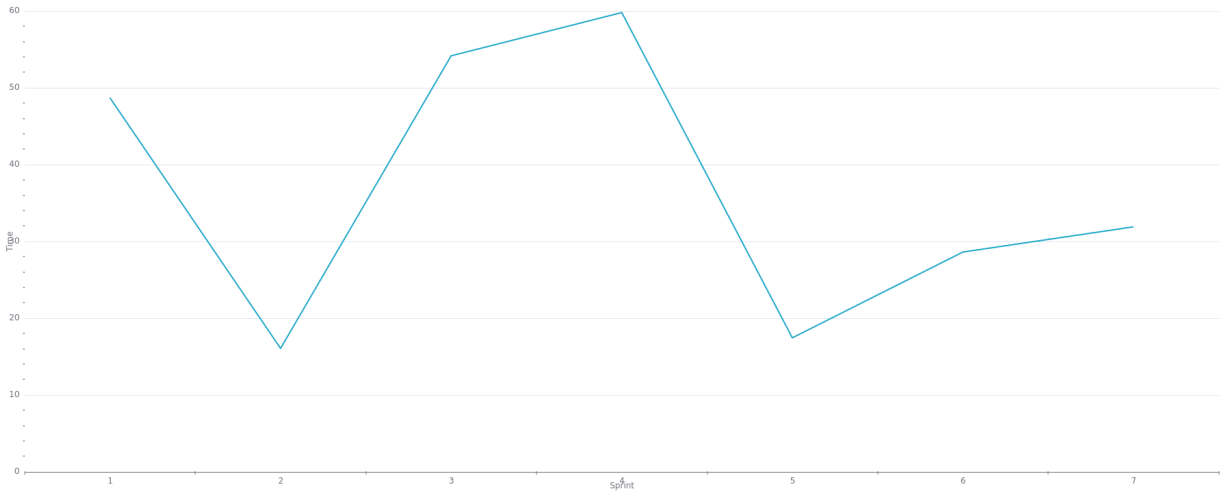
  - Setup of Issue tracking
  - Defining collaboration, roles, responsibilities and meetings
  - Creation of a long-term plan
  - Risk-Manangement
  - Define requirements
    * Functional requirements
    * Non-functional requirements
- M2 - End of elaboration (17.10.2023)
  - Phase of Elaboration is completed and documented
  - Refined requirements are sorted according to priority
  - Architecture is defined
  - Creation of prototype is done
  - All preparations for implementation are completed
  - Quality measures are defined
- M3 - Alpha version (28.11.2023)
  - All primary features are implemented rudimentary
  - Documentation is up-to-date with the technical solution
- M4 - Beta version (12.12.2023)
  - Severe bugs are mitigated
- M5 - Final Submission (22.12.2023)
  - Finished documentation
  - Finished application

Timeplan

		Sprint	Sprint #1		Sprint #2		Sprint #3		Sprint #4		Sprint #5		Sprint #6		Sprint #7	
		SW	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Inception	Setup of Issue tracking / Time tracking	Estimate														
		Actual														
	Setup of Build pipelines	Estimate														
		Actual														
	Risk-Management	Estimate														
		Actual														
	Defining collaboration, roles, processes and meetings	Estimate														
		Actual														
	Creation of long-term plan	Estimate		M1												
		Actual		M1												
Elaboration	Requirement Analysis	Estimate														
		Actual														
	Refine Requirements (Prioritize)	Estimate														
		Actual														
	Define initial architecture	Estimate														
		Actual														
	POC / Prototype (Take off & hold position)	Estimate														
		Actual														
	Test concept & quality measures	Estimate				M2										
		Actual				M2										
Construction	Dropoff	Estimate										M3		M4		
		Actual														
	Navigation	Estimate														
		Actual														
Transition	Improve model	Estimate														M5
		Actual														
	Finalize Documentation & Product	Estimate														
		Actual														
Legend: Estimated time, Actual time, Milestones		SW	1	2	3	4	5	6	7	8	9	10	11	12	13	14
		Sprint	Sprint #1		Sprint #2		Sprint #3		Sprint #4		Sprint #5		Sprint #6		Sprint #7	

Chapter 10

Time Tracking Report



This results in a total time of

256.35

Chapter 11

Personal Report

11.1 What did go well

The communication with the project advisor was always respectful, open and clear. Both parties could contribute their knowledge and findings, creating an environment allowing to develop knowledge. In addition, the defined processes were always adhered to, even though they could not always be checked.

Also, there was no time lost in additional meetings, as I was the sole team member, allowing faster reactions to mistakes and areas to improve.

From my point of view, the priorization in general, but also focused on clean code and architecture, was always well done and also adhered to.

The time estimations were mostly accurate, which is a huge improvement compared to the previous projects.

11.2 Areas to improve

Although the time estimations were accurate for the most tasks, there were some big tasks, which were estimate too low. This is mostly due the fact, that these tasks were to big to estimate correctly. Therefore, these task should have been broken down into smaller tasks, allowing a more precise estimation of the effort needed to complete them.

Due to my lack of knowledge in the area of AI, some tasks, such as hyperparameter tuning, needed a lot of time, while a expert would probably guess quite good hyperparameters and then just do the fine-tuning. In contrast, I had to try out everything, from big to low hyperparameter values and back again. This, however, will improve when gaining knowledge in this area, I guess.

Apart from this, the project went quite smoothly.

11.3 Personal highlights

Firstly, I have never done a large project in the area of AI. So successfully conducting a project in this area was of course a highlight.

Because I was the sole team member, I was very free in organising my time. This was especially nice, since I did a exchange semester in Norway, allowing me to do nice trips to see this beatiful country. The time lost there I just needed to catch up beforehand or after the trip.

Part V

Appendix

Bibliography

[1] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. https://arxiv.org/abs/1801.01290, Accessed: 05.11.2023. 2018.

[2] OpenAI. *Key Concepts in RL*. https://spinningup.openai.com/en/latest/spinningup/rl_intro.html. Accessed: 05.11.2023. 2018.

[3] OpenAI. *Soft Actor-Critic*. https://spinningup.openai.com/en/latest/algorithms/sac.html. Accessed: 05.11.2023. 2018.

[4] Suraj Regmi. *CartPole Problem Using TF-Agents — Build Your First Reinforcement Learning Application*. https://towardsdatascience.com/cartpole-problem-using-tf-agents-build-your-first-reinforcement-learning-application-3e6006adeba7. Accessed: 21.10.2023. 2020.