DATA-DRIVEN DRONES: AN ANALYSIS OF LARGE LANGUAGE MODELS IN REINFORCEMENT LEARNING FOR DRONES

Ng Shi Qing Eugenia[1], Vera Ong Liwen[2], Kuan Qi Heng Benson[3]
[1]River Valley High School, 6 Boon Lay Avenue, Singapore 649961
[2]Raffles Institution (Junior College), 1 Raffles Institution Lane, Singapore 575954
[3]DSO National Laboratories, 12 Science Park Drive, Singapore 118225

1. Introduction

With Artificial Intelligence (AI) being such an integral part of modern technological advancements, many have applied it onto drone navigation systems using deep learning. One of these branches in particular, Reinforcement Learning (RL), involves an agent learning to make decisions by interacting with an environment. With RL, the environment that an agent is trained in is usually a simulator that allows for numerous instances of training without incurring high cost. Through the optimisation of RL models for drone navigation, drones used for surveillance, pursuit, and otherwise can be trained to perform complex tasks with an accuracy that is not achievable by traditional approaches. However, one of the challenges of RL is the crafting of the rewards functions that the agent uses to improve its sequence of decision making. In this project, the usage of Large Language Model (LLM) is being explored to help improve on the RL model's rewards function that is usually crafted manually by hand. In summary, this project aims to investigate the potential of using LLMs in RL drone models for simple navigation.

2. Literature Review

2.1 Autonomy in Drone Navigation

There have been many advancements in the fields of AI and drone development in recent years, which includes the use of deep learning to optimise drone navigation. By utilising AI, drones account for its surroundings in enclosed environments by detecting objects to clearly map out its environment. This is important in other implementations like navigation in enclosed spaces autonomously (i.e. without the need for human involvement), collision avoidance, and automatic takeoff and landing. Other uses of AI in drones include the optimisation of trajectories and paths in foreign environments, distinguishing environments based on geographical features, and planning three-dimensional, non-planar movement (Lee et al., 2021). With drones increasing in popularity over the past decade, firms have invested efforts into AI implementations, with a notable example of Near-Earth Autonomy and National Aeronautics and Space Administration's (NASA) breakthroughs in self-piloted unmanned drones and autonomous systems, that have largely reduced the reliance on Global Positioning Systems (GPS) for drone navigation (NASA, 2020).

2.2 Reinforcement Learning (RL)

RL is a branch of deep learning that involves a series of decisions based on exploration and exploitation by the agent, where "correct" actions, which brings the agent a step closer to achieving its goals, are rewarded and "incorrect" actions, which brings the agent a step further to achieving its goals, are penalised. By interacting with the given environment, the agent can recognise "correct" and "incorrect" actions through the rewards gained from the rewards functions, and thereafter continue choosing to act on the sequences of actions that provide the greatest overall reward and minimise overall penalties. Such rewards would act as reinforcement for the agent to approach optimal behaviour. (Kaelbling, 1996).

To speed up RL model training, episodic learning models were used, where agents leverage on past observations and actions to further learn, rather than starting from scratch each time. This

"learn by learnt" approach allows for greater RL model optimisation due to a more efficient trial-and-error approach (Botvinick et al., 2019). However, another approach that can be taken to optimise RL models would be to optimise the rewards function dictating the behaviour of agents. Much of today's efforts to maximise the efficiency of such functions are done manually through fine-tuning.

2.3 Reinforcement Learning in Real-World Contexts and Drone Navigation

As RL is most advantageous in contexts where the agent must take in inputs from its external environment such as feedback and sensory inputs and requires great amounts of data to be implemented effectively (Haque et al., 2023), applying RL into today's drone navigation developments can allow drones to be adapted to many functions.

The use of autonomous high-speed drones includes search and rescue in an unknown environment. Using sensors like the camera, the drone will be able to visualise the environment in the form of a map that allows it to plan safe trajectories to a desired goal state. (Karatzas et al., 2022) The RL process for drone navigation can be illustrated by the flowchart below (AlMahamid et al, n.d.):

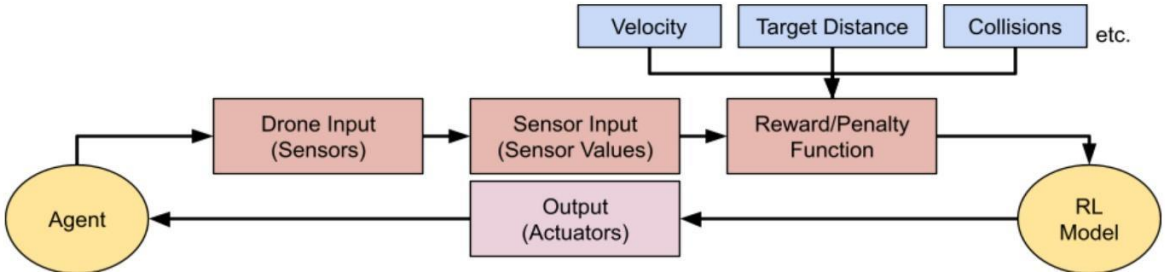


Figure 2.1 Flowchart visualising RL system

Such technology has been used for drone racing as well, where RL modelling is used to plan minimum-time trajectory and paths from waypoint to waypoint while ensuring obstacle avoidance in a randomised map. These trajectories are often high-velocity and aggressive movements that accurately and precisely utilise actuators to navigate sharp turns and messy terrains, which pushes drone controls to its maximum efficiency (Song et al., 2021). Even in situations where a gentler trajectory is required, RL can allow drones to follow paths or landmarks through object distinction to navigate areas (Jacob et al., 2022).

Furthermore, RL can be used to program surveillance or pursuit drones that are designed to follow a specified moving object, such as a vehicle or person, through a system similar to Figure 2.1 but with a reward function centred around distance from target and orientation of drone compared to bearing of target (Darwish et al., 2021). This function, when combined with high speed movement in random terrain mentioned above, will yield a drone that is efficient and able to pursue and survey autonomously.

Lastly, RL is integral to delivery drones, where safe and stable flight regardless of environment is essential to its function. By using RL to path find and determine the safest or fastest route to its destination, as well as to ensure stable flight, the drone will then be able to deliver cargo efficiently and effectively (Munoz et al., 2019).

However, as a start to the research into the effectiveness of implementing LLMs into drones, we decided to just train it to hover rather than getting it to avoid obstacles and plan its path.

3. Methodology

3.1 Repository Implementation and Modifications on Linux OS

With the pre-existing gym-pybullet-drones repository (utiasDSL, 2023), the first step would be to ensure that the code environment is correctly implemented and tested before any further modifications.

First, the repository was downloaded onto an Ubuntu laptop running Linux OS, as python libraries such as gymnasium, as well as high-level coding, are best supported on Linux. Once all setups had been completed, the original source code was ensured to run as indicated on the repository notes before further modifying the code as will be explained below.

3.1.1 Continuous Learning

For the model to build on its past training and restore its training progress each time the code was run, continuous learning was implemented. We did this by saving the best model into a zip file each time the training was finished and loading the parameters into the model before training it during the next run of the code. Continuous learning allows us to set shorter training iterations to better visualise the progress of the drone. We can then more accurately determine when the drone has been sufficiently trained to reach the target position. The specific number of training iterations during each run of the code will be further discussed in section 3.3.

3.2 Modifying of Reward Function

For the human-crafted rewards function, we decided to implement a simple formula which penalises the model based on its euclidean distance away from the target position:

$$reward = max\ (0, 2z - d^4)$$

where z is the Euclidean distance between target and origin,
d is the Euclidean distance between current coordinates and target coordinates

The further it is from the target position, the higher the penalty. The euclidean distance for two points in the 3-dimensional environment was calculated using the following formula (Tabak, 2008):

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

where d is the Euclidean distance,
p is target position,
q is the current position,
n is the dimension of the coordinate

By rewarding the model when its euclidean distance from its target position is low and penalising the model when its euclidean distance from its target position is high the movement of the model becomes smoother and less abrupt after multiple iterations of training. The formula allows it to be used for different target coordinates and the Euclidean distance between current coordinates and target coordinates to the fourth power increases the rewards given exponentially as the drone gets closer to the target coordinates.

3.3 Training Models in Iterations

To provide an initial estimate of how many iterations would be sufficient to train a model in general, a large, randomised number of timesteps were used to train the model with a given target position of (0,0,1). Through numerical analysis, we determined that $5x10^5$ timesteps is

generally sufficient to train the model to reach this target position. If too little timesteps like $1x10^3$ were used, there would be barely any improvements observed. Therefore, all models in this project were trained in $5x10^4$ timestep iterations, until the drone was sufficiently trained, to reduce any chances of overfitting or overtraining, and visualise the training progress between iterations. This also allows us to compare between models by comparing the number of timesteps required to train the drone to navigate to a given position, efficiently comparing the effectiveness of any modifications.

3.4 Implementation of OpenAI's ChatGPT API

To implement the LLM-crafted rewards function, we integrated OpenAI's GPT model into the code itself.

3.4.1 Version of the GPT Model

The version of the GPT Model we used was GPT-4, the latest model in the GPT series. We chose this model as it was the latest and most advanced model currently. It is more creative and able to handle much more nuanced instructions than GPT-3.5, which we felt would increase the reliability and accuracy of the reward function crafted by it. Furthermore, GPT-4 performed better as compared to the current State-of-the-Art (SOTA) model and GPT3.5 model when evaluated using HumanEval (OpenAI, 2023), which evaluates code generated by LLMs (OpenAI, 2021). Our project is reliant on the rewards function generated by the LLM and therefore we felt that the increase in accuracy of the generation of code by GPT-4 could help us yield better results compared to GPT-3.5 and therefore decided to use this model.

3.4.2 Memory of conversation

Unfortunately, the GPT model does not retain past conversation history and in turn, the context from the previous queries and responses.

However, the GPT model does take in a parameter called 'messages' where we can input in past queries and outputs as contexts for its future outputs. Therefore, each past query and response was saved into a list of dictionaries and parsed through each time the GPT model was being used (OpenAI, n.d.). This allows the model to access its past reward functions and modify the parts of it that were not successful based on the query from the most recent evaluation.

3 4.3 Modification of context, input and output of the LLM

To increase the autonomy of the feedback input and implementation of the LLM-crafted rewards function, we had to give the model context to modify its output given our input.

For the context, we added the code for the environment's action and observation space so that the model can better understand how the rewards function affects the control of the drone. We also gave it specific instructions on what the format of the model's output should be. For example, to get the model to generate a rewards function that could be immediately executed, the context we provided was to "generate a python code (not in a code block) with no other commentary. create a function called generate_rewards that can only take in 6 positional arguments, x, y, z, targetx, targety, targetz".

To implement the LLM crafted reward function, we queried in the final distance the drone was at the end of its final evaluation. The LLM outputs an updated version of its rewards function based on the distance away from the target position the drone was during the last evaluation. This output will then be saved into a text file and executed using the exec() function whenever

the environment's reward function is called. After each run of the code, the distance will be calculated and will be queried into the LLM automatically

4. Results

4.1 Human-crafted Rewards Function

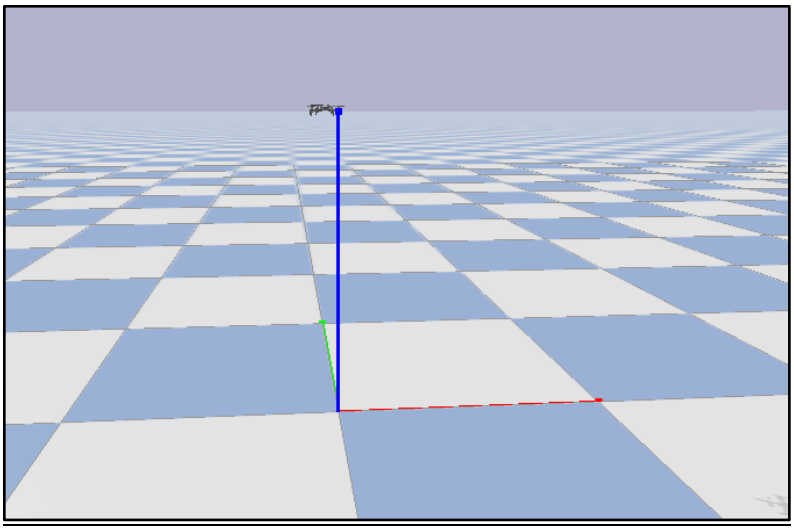


Figure 4.1 Drone hovering at (0,0,1) in the simulation

Using the rewards function mentioned in section 3.2, the drone was able to achieve stable flight to the coordinates (0,0,1) within $2x10^5$ timesteps. The drone was able to achieve stable flight to the coordinates (0,0,2) within $3x10^5$ timesteps. Figure 4.1 shows the drone hovering stably after training. The graphs of the drone's dynamics during the final evaluation. Annex A and B shows the graphs of the drone's dynamics during the final evaluation and the mean rewards throughout the training period for coordinates (0,0,1) and (0,0,2) respectively.

Other hand-crafted rewards functions that were tested included exponentially increasing rewards as the distance between target and drone decreases, as well as rewarding when the drone is hovering at a steady height and penalising when the drone overflies past the target position to streamline flight. Though it had reduced training timesteps slightly, the overall complexity of the rewards function paired with the negligible change in results made this modification unimportant to the overall rewards function. Therefore, the original function was restored for use in the model.

4.2 LLM-crafted Rewards Function

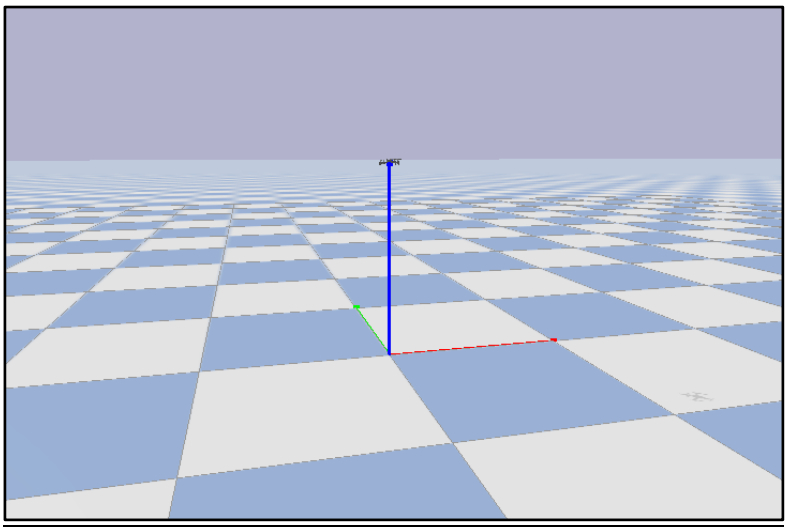


Figure 4.2 Drone hovering at (0,0,1) in the simulation

The drone trained using the LLM-crafted rewards function managed to hover stably at the target position of (0,0,1) after $1x10^5$ timesteps and could hover stably at the target position of (0,0,2) after $2x10^5$ timesteps. Annex C and D shows the graphs of the drone's dynamics during the final evaluation and the mean rewards throughout the training period for the target coordinates (0,0,1) and (0,0,2) respectively.

5. Discussion

5.1 Potential of LLMs

Fine tuning and adjusting the rewards function could be extremely time-consuming, especially when the effectiveness of the training in RL is so heavily dependent on it. However, if we were to use LLMs to automate this process, we can not only save time but can also reduce the human biases that come with human-crafted reward functions. Furthermore, LLMs can better understand the semantics of the query and can output a reward function that can be used to encourage or reduce certain behaviours that might arise from a normal human-crafted reward function. Its effectiveness can be seen in our results where the LLM-crafted rewards function managed to achieve a higher accuracy and had a nearer final coordinates to our target coordinate as compared to the hand-crafted rewards function.

5.2 Reinforcement Learning with Human Feedback (RLHF)

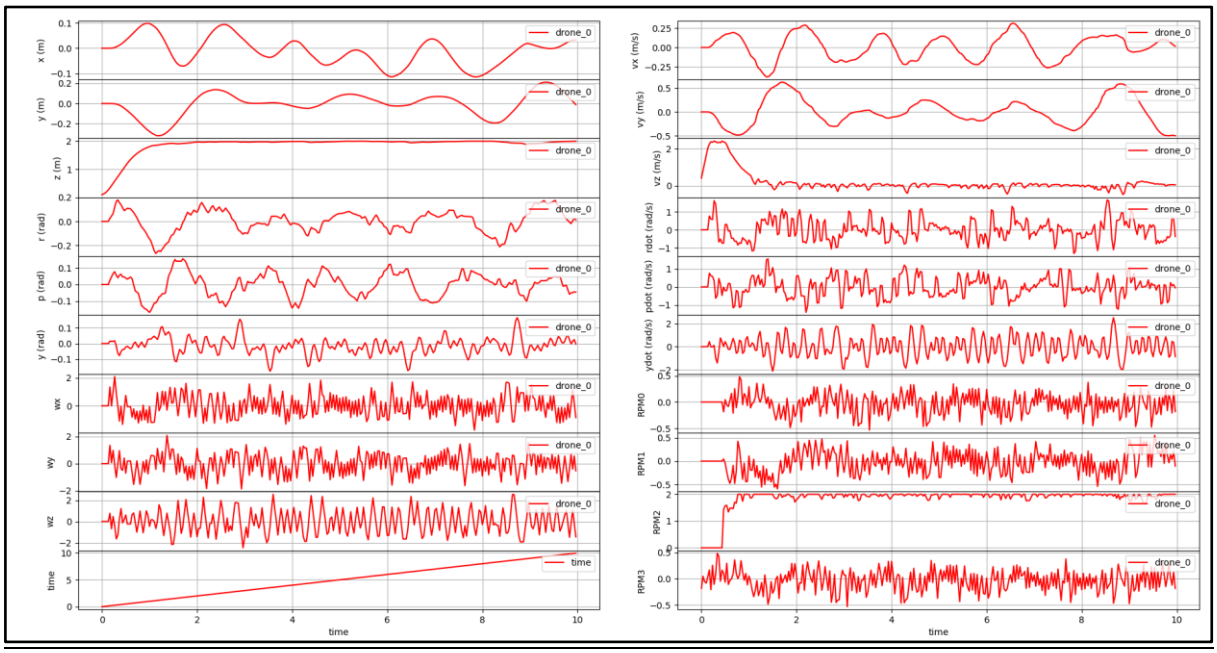


Figure 5.1 Graphs of drone's dynamics during final evaluation with target position (0,0,2)

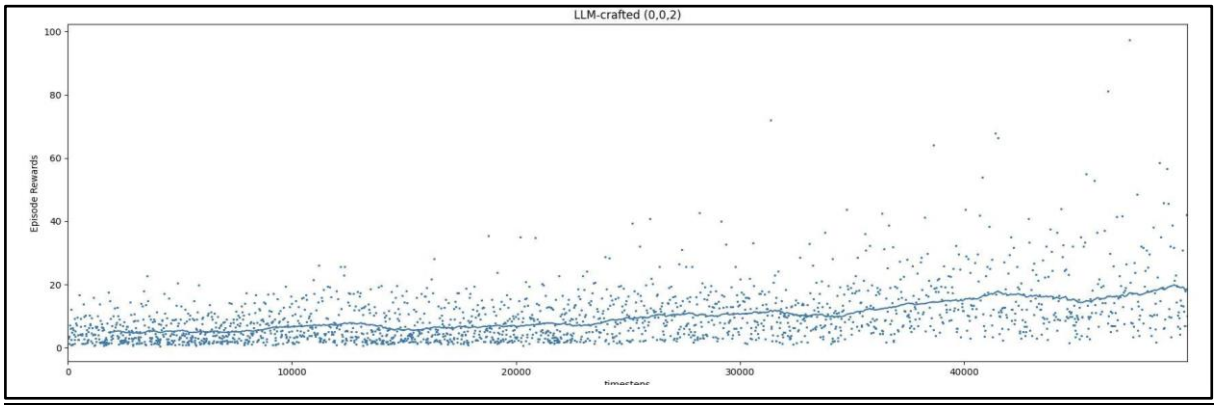


Figure 5.2 Graph of mean rewards during training

Other research projects such as Eureka (Eureka Research, 2023) have integrated human feedback into their LLM inputs to improve the efficacy of the reinforcement learning training and therefore we wanted to test if the usage of RLHF would benefit the training of drones. To test this, we queried the human feedback in words after the end of each $5x10^4$ interval for the LLM to take into consideration before modifying its rewards function. The training timesteps to train the model to hover stably at the target position of (0,0,2) was further shortened to 10^5. Figure 5.1 shows the x, y and z-coordinates of the drone and the z-coordinate is relatively stable throughout. Figure 5.2 shows the increasing trend of the mean rewards after its training.

5.3 Evaluation of Rewards Functions

Based on the results seen in section 4, using LLMs to craft and tweak the rewards functions can evidently increase the efficiency and effectiveness of the model's training. The timesteps needed to train the model to hover at (0,0,1) decreased from $2x10^5$ to $1x10^5$ using the human-crafted and LLM-crafted rewards function respectively, which is a 50% decrease in the time taken. The timesteps needed to train the model to hover at (0,0,2) also decreased from $3x10^5$ to $2x10^5$ using the human-crafted and LLM-crafted rewards function respectively, which is a 33% decrease in the time taken. However, as seen in section 5.2, the drone trained using a combination of LLM-crafted rewards function and RLHF used the least number of timesteps, $1x10^5$, to train the drone to hover stably.

5.4 Limitations

The integration of LLMs into the RL training that we tested only showed results for relatively simple tasks such as the hovering of a drone. For more complex tasks such as object avoidance or zero-shot drone flying and hovering, the integration of LLMs may not be as effective and may not have impactful changes to the time taken to train the model.

6. Future Improvements

Improving on this project in the future would entail integrating LLMs into an end to end RL pipeline which allows for a drone to output control commands directly from sensory inputs that avoids obstacles when moving to a desired target goal as mentioned in section 5.4. Beyond that, LLM training has potential to branch into real-life drones, through implementing LLM-generated reward functions into real-life waypoint navigation and collision avoidance for use in different contexts as previously mentioned.

7. Conclusion

Through the analysis of the use of LLM to optimise rewards functions, it is concluded that LLM greatly reduces training required for RL drone navigation models to reach its intended target coordinates and has significant potential to be applied onto real-life drone platforms.

8. References

Autonomous Drone Racing with Deep Reinforcement Learning - Researchgate. (n.d.-a). https://www.researchgate.net/publication/350104776_Autonomous_Drone_Racing_with_Deep_Reinforcement_Learning

Autonomous Unmanned Aerial Vehicle Navigation using Reinforcement Learning: A Systematic Review - arxiv.org. (n.d.-b). https://arxiv.org/pdf/2208.12328

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., & Hassabis, D. (2019, April 16). Reinforcement learning, fast and slow. Trends in Cognitive Sciences. https://www.sciencedirect.com/science/article/pii/S1364661319300610

Drone Navigation and Target Interception Using Deep Reinforcement Learning: A Cascade Reward Approach - IEEE xplore. (n.d.-c). https://ieeexplore.ieee.org/document/10323488/

Exploring the benefits of reinforcement learning for autonomous drone navigation and control (n.d.-d). https://www.researchgate.net/publication/372987000_Exploring_the_Benefits_of_Reinforcement_Learning_for_Autonomous_Drone_Navigation_and_Control

Eureka-Research. (2023). Eureka-Research/Eureka: Official Repository for "eureka: Human-level reward design via coding large language models." GitHub. https://github.com/eureka-research/Eureka

Google. (n.d.). Geometry: The Language of Space and Form. Google Books. https://books.google.com.sg/books?id=r0HuPiexnYwC&pg=PA150&redir_esc=y#v=onepage&q&f=false

GPT base, GPT-3.5 Turbo & GPT-4: What's the difference? Pluralsight. (n.d.). https://www.pluralsight.com/resources/blog/data/ai-gpt-models-differences

Jacob, B., Kaushik, A., & Velavan, P. (1970, January 1). Autonomous Navigation of drones using reinforcement learning. SpringerLink. https://link.springer.com/chapter/10.1007/978-981-16-7220-0_10

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996, May 1). Reinforcement learning: A survey. arXiv.org. https://arxiv.org/abs/cs/9605103

Karatzas, A., Karras, A., Karras, C., Giotopoulos, K. C., Oikonomou, K., & Sioutas, S. (1970, January 1). On Autonomous Drone Navigation using Deep Learning and an intelligent rainbow DQN agent. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-031-21753-1_14

Lee, T., Mckeever, S., & Courtney, J. (2021, June 17). Flying free: A research overview of deep learning in drone navigation autonomy. MDPI. https://www.mdpi.com/2504-446X/5/2/52

Muñoz, G., Barrado, C., Çetin, E., & Salami, E. (2019, September 10). Deep reinforcement learning for drone delivery. MDPI. https://www.mdpi.com/2504-446X/3/3/72/htm

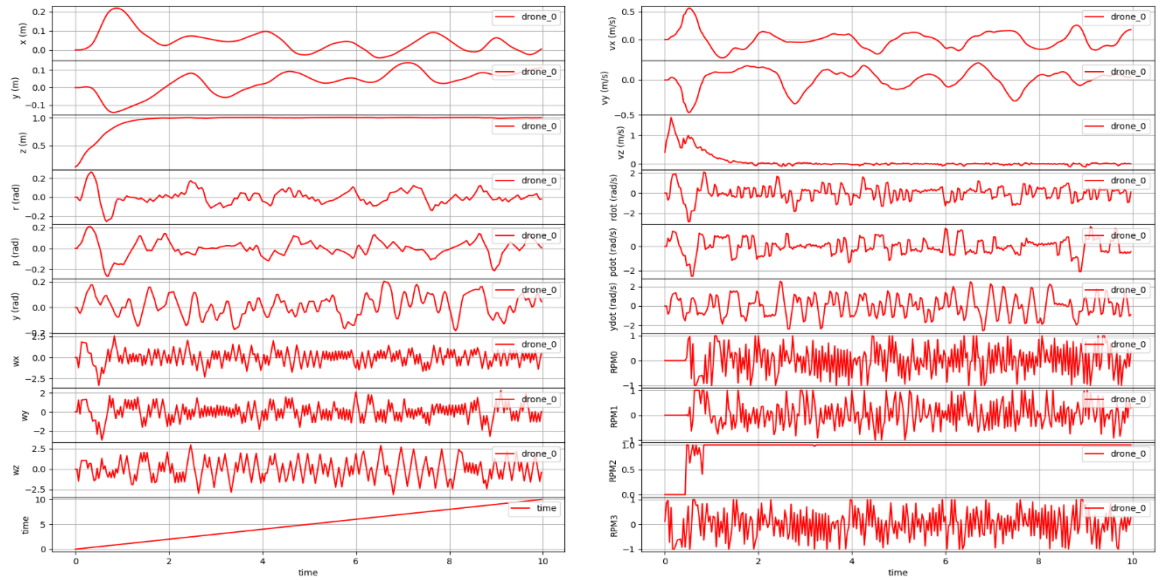
NASA. (n.d.). Autonomous drone navigation system ends reliance on GPS. NASA. https://spinoff.nasa.gov/Spinoff2020/ps_5.html

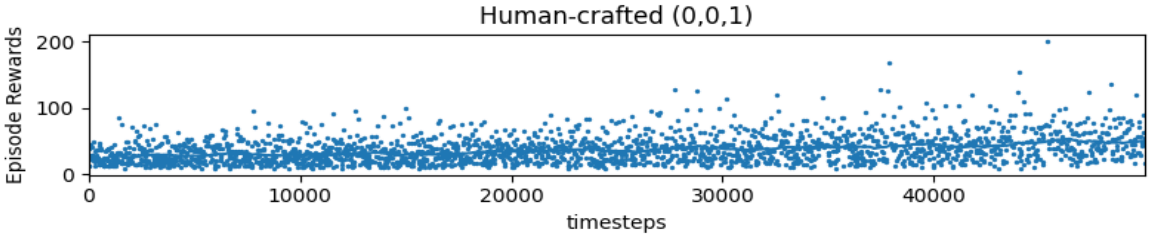
OpenAI platform. (n.d.-e). https://platform.openai.com/docs/api-reference
OpenAI. (2023, December 19). GPT-4 technical report. arXiv.org. https://arxiv.org/abs/2303.08774

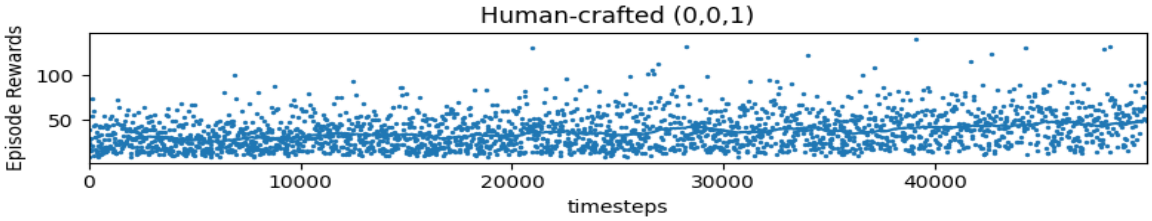
openai. (2021). Openai/human-eval: Code for the paper “evaluating large language models trained on code.” GitHub. https://github.com/openai/human-eval

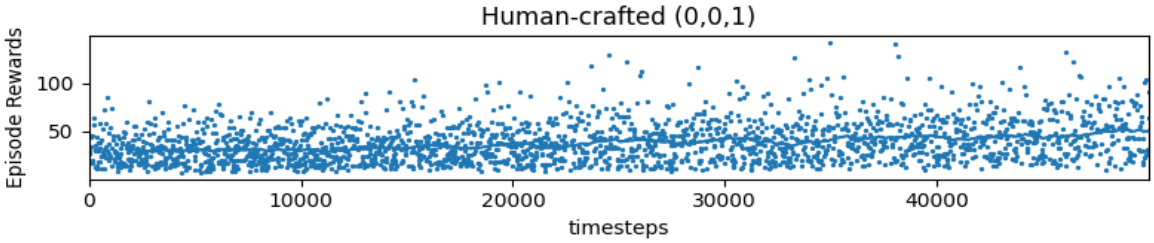
utiasDSL. (n.d.). UTIASDSL/gym-pybullet-drones: Pybullet Gymnasium environments for single and multi-agent reinforcement learning of Quadcopter Control. GitHub. https://github.com/utiasDSL/gym-pybullet-drones

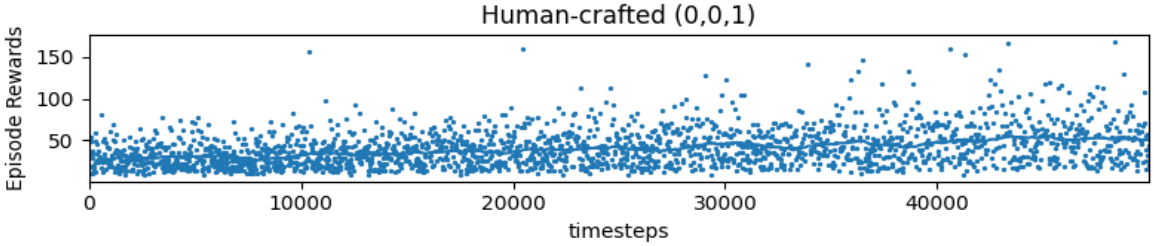
9. Annex


x (m)
y (m)
z (m)
r (rad)
p (rad)
y (rad)
wx
wy
wz
time
drone_0
vx (m/s)
vy (m/s)
vz (m/s)
rdot (rad/s)
pdot (rad/s)
ydot (rad/s)
RPM0
RPM1
RPM2
RPM3

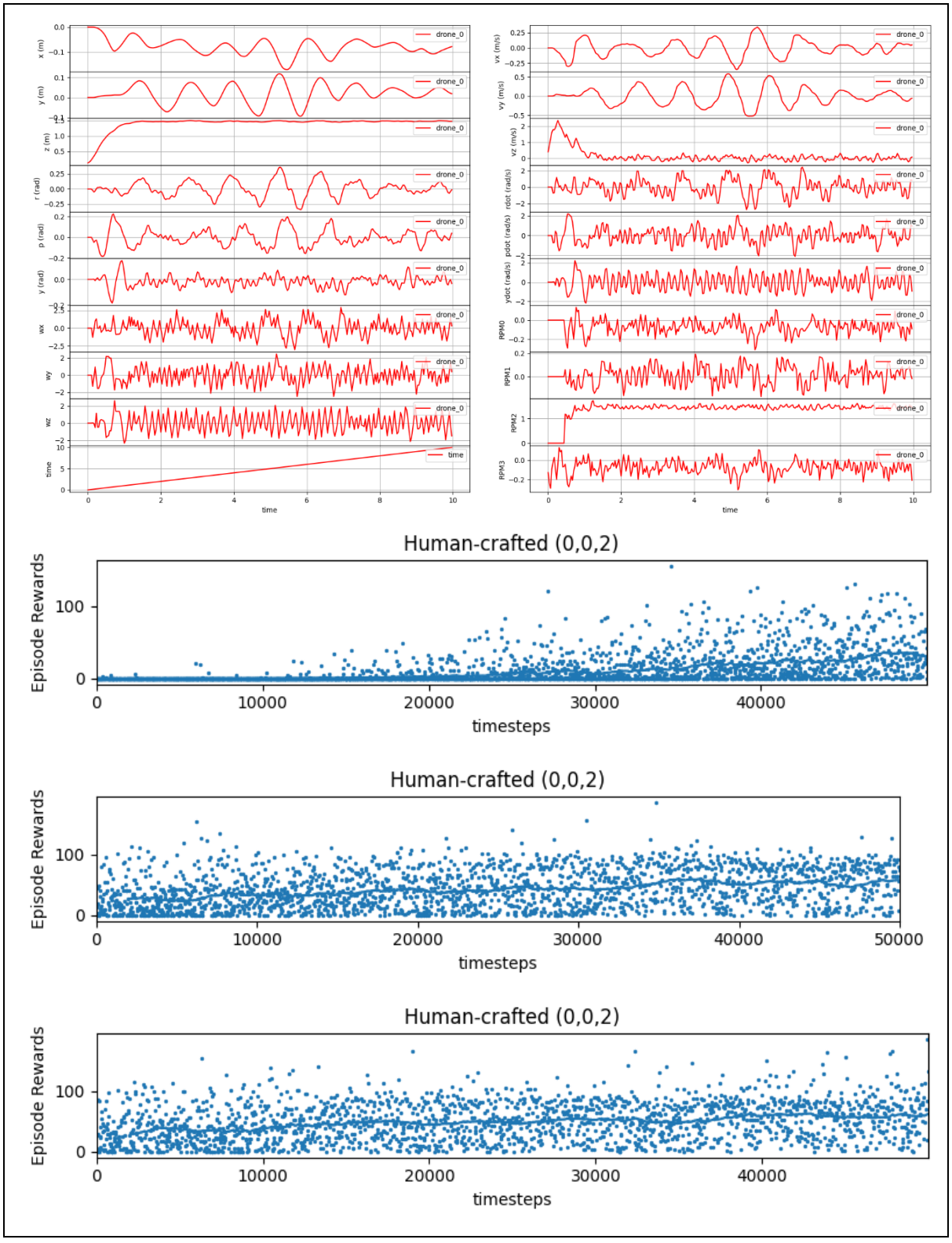

Human-crafted (0,0,1)
Episode Rewards
timesteps

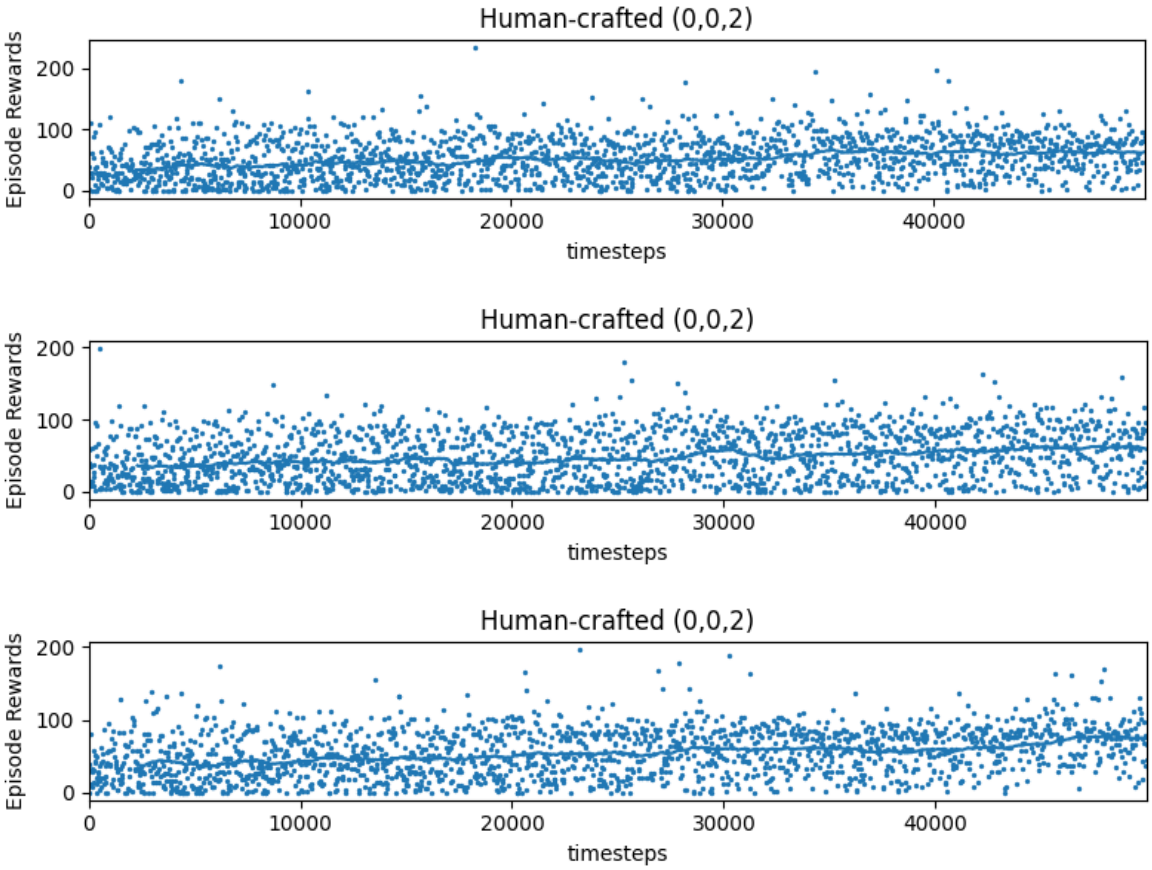

Human-crafted (0,0,1)
Episode Rewards
timesteps


Human-crafted (0,0,1)
Episode Rewards
timesteps

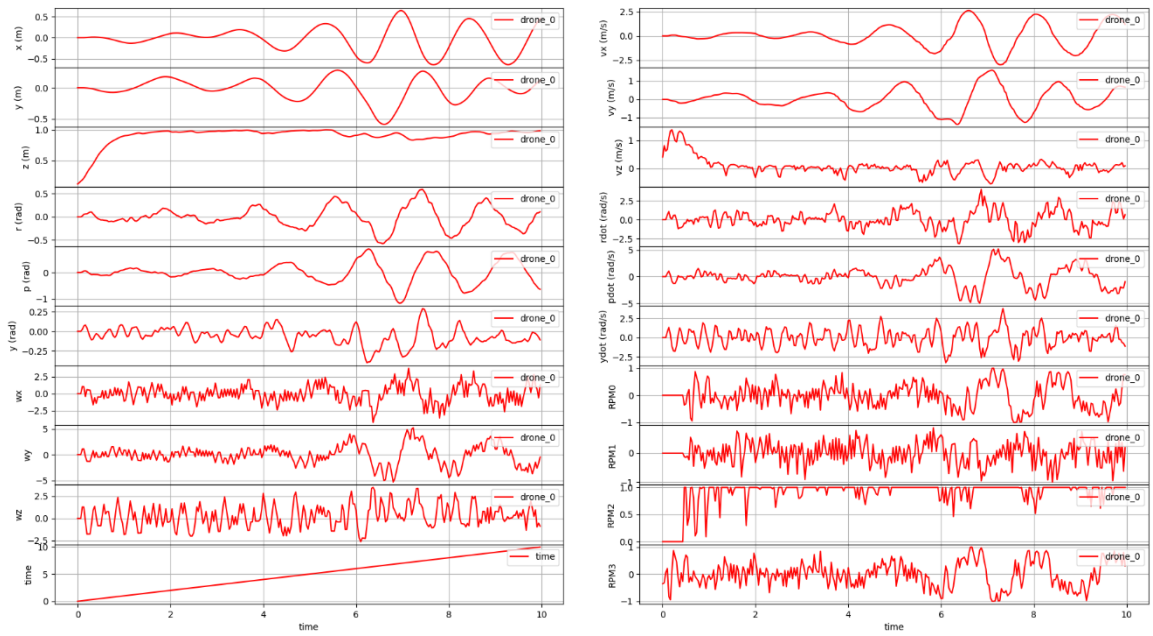

Human-crafted (0,0,1)
Episode Rewards
timesteps

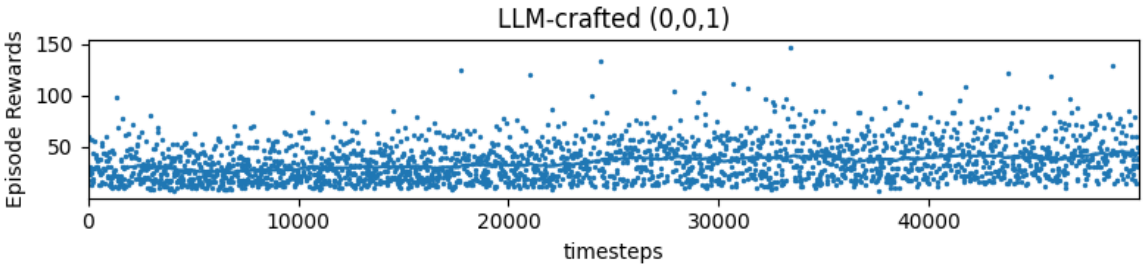
Annex A: Graphs of drone's dynamics during final evaluation and rewards during training using a human-crafted rewards function with target position (0,0,1)

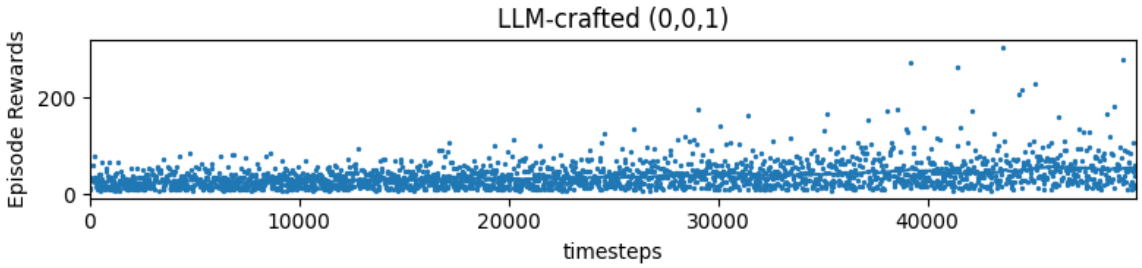




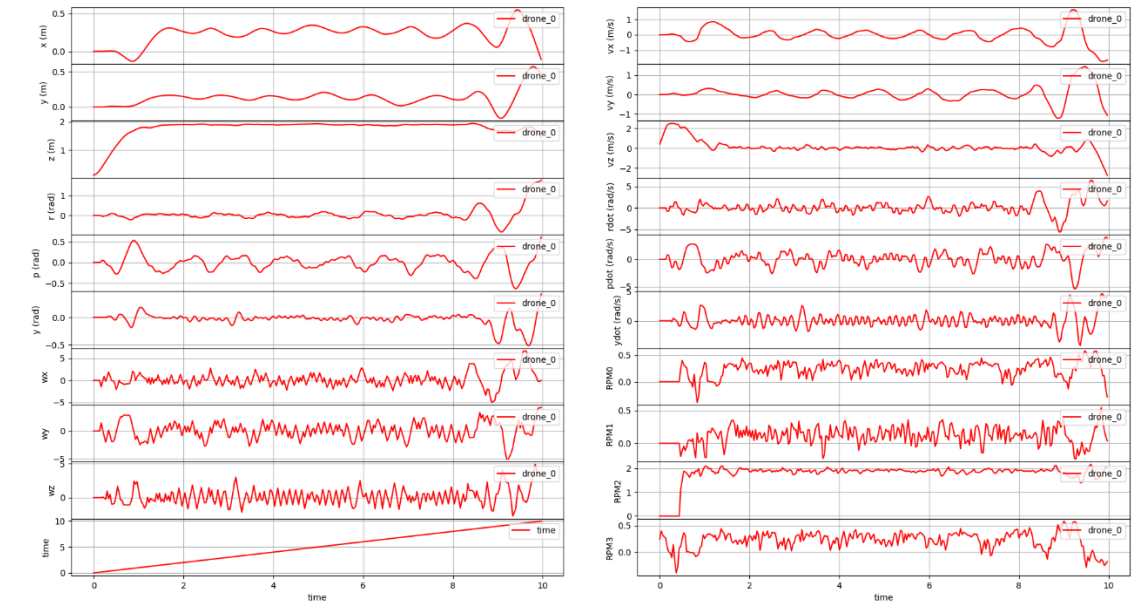
Annex B: Graphs of drone's dynamics during final evaluation and rewards during training using a human-crafted rewards function with target position (0,0,2)

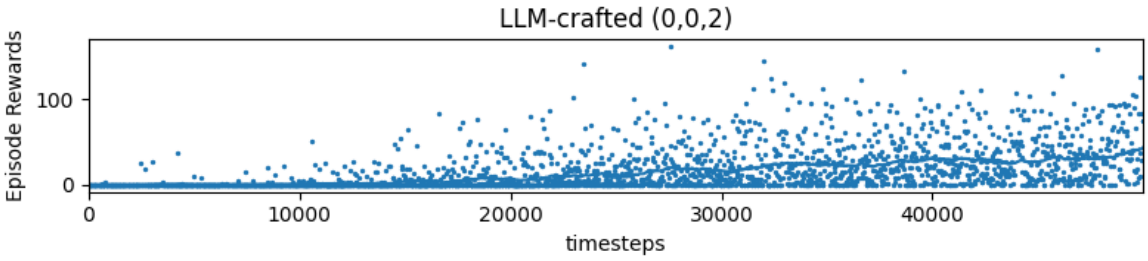


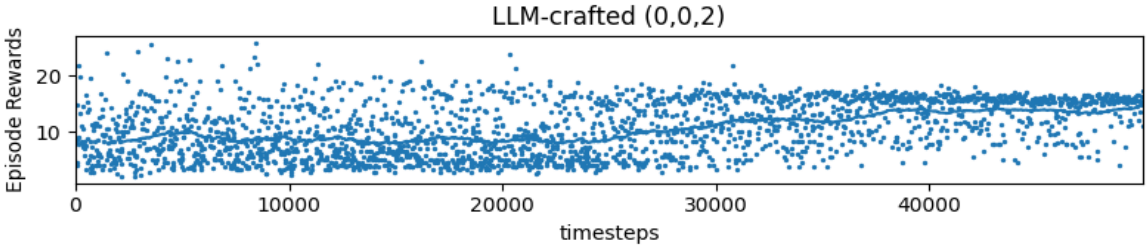


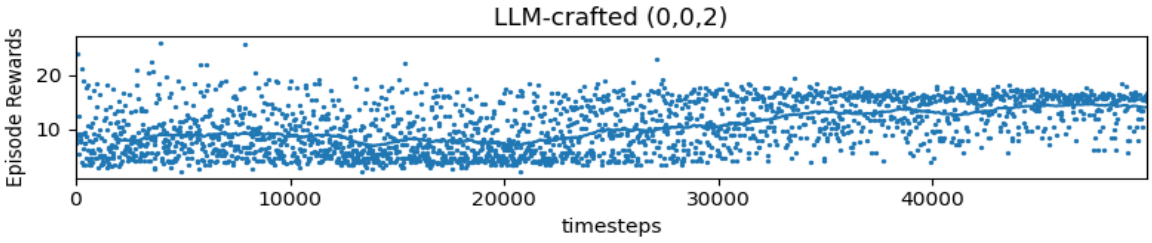


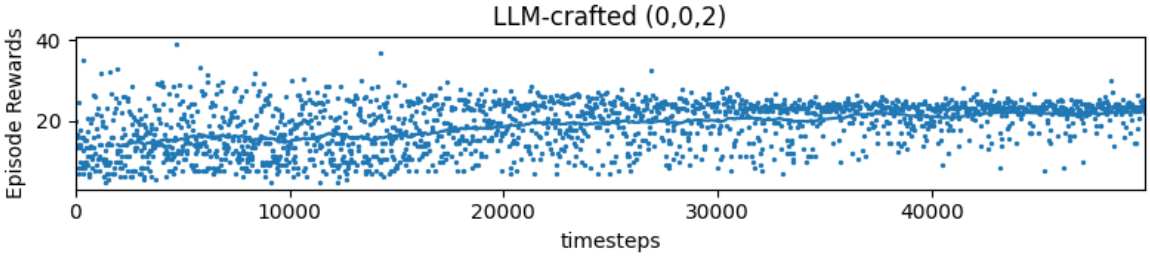
Annex C: Graphs of drone's dynamics during final evaluation and rewards during training using LLM-crafted rewards function with a target position of (0,0,1)











Annex D: Graphs of drone's dynamics during final evaluation and rewards during

training using LLM-crafter rewards function with target position (0,0,2)

Annex D: Graphs of drone's dynamics during final evaluation and rewards during training using an LLM-crafted rewards function with a target position of (0,0,2)