

Beyond PID: Benchmarking Reinforcement Learning Controllers for Quadcopter Stabilisation

Wietske van der Westhuizen*

vwswie001@myuct.ac.za

University of Cape Town

Cape Town, Western Cape, South Africa

Mathew Maartens*

mrtmat016@myuct.ac.za

University of Cape Town

Cape Town, Western Cape, South Africa

Abstract

Quadcopters (drones) have experienced significant growth in popularity over the past few decades. Advances in hardware miniaturisation and software sophistication have enabled quadcopters to achieve capabilities previously unattainable. The traditional Proportional Integral Derivative (PID) system is widely regarded as the industry standard for quadcopter control and performs reliably under stable conditions. However, PID control systems are limited in environments characterised by high variability in atmospheric conditions, such as strong wind turbulence. Recent technological advancements have made it increasingly feasible to incorporate advanced control methods, such as Reinforcement Learning (RL), into quadcopter applications. Nevertheless, RL also faces challenges; specifically, training in simulations does not always translate effectively to real-world conditions. Furthermore, cross-study analysis becomes inconsistent due to the varying environmental conditions and benchmarking metrics. This project addresses these challenges by benchmarking contemporary on-policy and off-policy RL algorithms against the PID implementation. The paper will produce a reproducible MuJoCo/Gymnasium benchmark for quadcopter hover stabilisation, comparing PID against six RL controllers under identical environment, reward, tuning, and disturbance specifications.

1 Introduction

With the rapid growth of Unmanned Aerial Vehicles (UAVs), particularly quadcopters, their applications have expanded across a wide range of domains, including agriculture, surveillance, and logistics [9]. The property that has made quadcopters widely used in these domains is their stabilisation controls, without which they could not stabilise themselves to perform useful tasks.

The current industry standard for quadcopter stabilisation is the Proportional-Integral-Derivative (PID) approach, valued for its computational efficiency and well-established tuning methodologies [16]. However, PID control is often limited in its ability to handle nonlinear dynamics and disturbances in real-world environments such as wind effects, rotor coupling, and varying payload conditions [6]. Reinforcement Learning (RL) offers a data-driven alternative instead of relying on a mathematical model. It achieves stabilisation through interaction with the environment. The recent advances in the field have demonstrated promising results in continuous control tasks of comparable complexity.

Despite this potential, the literature surrounding data-driven approaches to quadcopter stabilisation remains fragmented. Many

studies focus on a small subset of algorithms in isolation, making it difficult to reliably draw cross-study comparisons. This is compounded by variations in study conditions, such as different physics simulators, reward functions, termination conditions and tuning procedures. This results in one algorithm appearing more favourable than another, but it raises an important research question: are these differences due to an intrinsic algorithmic advantage, or a reflection of experimental conditions?

This study addresses this question by benchmarking on-policy and off-policy algorithms against a classical PID baseline, using MuJoCo as the unifying physics simulation framework. The RL algorithms benchmarked are Proximal Policy Optimisation (PPO), Advantage Actor-Critic (A2C), Constraint Regularisation PPO (SCR-PPO), Twin Delayed DDPG (TD3), Soft Actor-Critic (SAC), and Twin Average Delayed DDPG (TAD3). These algorithms are benchmarked against the classical implementation, the Proportional-Integral-Derivative (PID). These algorithms will be evaluated in an identical environment and under a standardised hyperparameter optimisation protocol, and reward functions to avoid reward shaping. By following this process, the study ensures that the observed performance differences reflect genuine algorithmic properties rather than experimental inconsistencies.

2 Background

The literature in this area has addressed many data-driven approaches to controller implementation, with varying degrees of depth and experimental conditions. This is presented in Table 1.

The contributions made by the papers focused on comparisons between RL techniques and the various properties of simulation frameworks. While new RL algorithms were developed to solve problems, many of them introduced further complexities due to overestimation and underestimation biases. While the use of MuJoCo as a physics simulation framework in one of the previously mentioned papers demonstrates its viability in research, that same paper was published in 2018. This is important, as MuJoCo was still proprietary at that point. Subsequent improvements to the simulation framework, especially after it was made open-source in 2022, were not addressed in the context of comparing RL techniques. The paper comparing simulation frameworks, however, was published in 2024 and described challenges regarding limited usability.

3 Problem Statement and Research Question

3.1 Problem Statement

The current literature on RL based quadcopter stabilisation lacks reproducible and standardised comparison against a classical PID baseline. Evaluations are done using a small subset of algorithms

*Both authors contributed equally to this research.

Study	Algorithms / Controllers	Environment	Classical baseline?	Task-specific flight/control metrics?	Key gap
Bernini et al. (2021) [7]	RL controllers for quadcopter attitude control	Quadcopter simulation	Partial; not central	Yes	Does not provide a broad benchmark across modern on-policy and off-policy algorithms under a shared PID comparison.
Hsiao et al. (2019) [11]	PPO, A2C, PID	OpenAI Gym-style quadcopter environment	Yes	Partly	Useful baseline comparison, but limited algorithm coverage and less emphasis on standardised benchmarking.
Fujimoto et al. (2018) [10]	DDPG, TD3, PPO, SAC	MuJoCo continuous-control tasks	No	No	Strong RL comparison, but not quadcopter-specific and no classical control baseline.
Liu (2024) [14]	DDPG, TD3, SAC, PPO	General continuous-control simulation tasks	No	No	Compares major deep RL algorithms for continuous control problems, but does not evaluate them in the quadcopter hover-stabilisation benchmark or against PID baseline.
Kaup et al. (2024)[12]	Physics engine comparison	Multiple simulators, including MuJoCo	No	No	Supports simulator choice, but does not evaluate controllers.
Proposed study	PID, PPO, A2C, SCR-PPO, SAC, TD3, TAD3	MuJoCo + Gymnasium quadcopter benchmark	Yes	Yes	Provides a standardised benchmark with shared environment, reward, HPO protocol, disturbance conditions, and flight metrics.

Table 1: Comparison of quadcopter-control, continuous-control RL, and simulation studies relevant to the proposed benchmark.

with varying simulation environments [7]. Furthermore, these studies lack evaluating the RL results directly with PID controllers, using domain specific flight metrics and not theoretical RL metrics [7]. Subsequently, it is unclear whether the observed performance differences between algorithms reflect genuine algorithmic advantages or are a result of differing experimental conditions.

3.2 Research Question

This study investigates whether RL controllers outperform a classical PID baseline for quadcopter hover stabilisation, when evaluated under identical environment specifications. The study is divided into two workstreams:

- **Wietske:** Off-policy RL methods – SAC, TD3, and TAD3
- **Mathew:** On-policy RL methods – PPO, A2C, and SCR-PPO

Each workstream compares its respective algorithms against the PID baseline across three performance dimensions: control performance, training reliability, and computational feasibility.

3.3 Research Objectives

Table 2 defines the metrics used to assess the RL methods.

4 Procedure and Methods

The project’s implementation will consist of 4 phases. Each phase focuses on achieving the milestones required for the subsequent phase. Phase 1 outlines the setup of the simulation environment and the PID baseline. Phase 2 implements the RL algorithms, with phase 3 dedicated to optimisation of hyperparameters. Finally, the simulation runs and result gathering will be conducted in phase 4.

4.1 Phase 1: Environment Setup and PID Baseline

This phase provides an environmental setup that enables fair comparisons of algorithms for the quadcopter. Due to the diversity of

Objective	Metric
Hover Stability	RMSE measured over the final 10 seconds of a hover episode, control smoothness, maximum overshoot, and attitude oscillations reported as peak-to-peak Euler angle amplitude
Disturbance Rejection	RMSE and attitude oscillations measured across five wind disturbance conditions relative to the no-disturbance baseline
Convergence Reliability	Success rate across seeds and the convergence speed in environment time steps
Computational Feasibility	Inference latency in microseconds per control step, validated against the 250Hz-1kHz real-time control requirement

Table 2: Shared Research Objectives and Evaluation Metrics

RL algorithms used, it is important to consider the architecture of the setup to ensure that one algorithm does not perform better than the others. Implementing the controllers systematically allows convenient swapping one for another. Implementing the PID as a baseline also serves as a test of the environment, ensuring that all components work cooperatively.

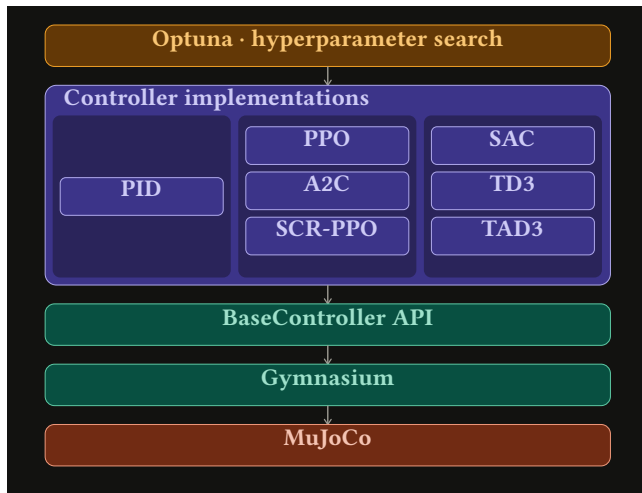


Figure 1: This diagram describes the structure of the experiment. Each controller implementation will interact with the underlying simulation framework through a BaseController API for standardisation.

4.1.1 MuJoCo Environment. MuJoCo (Multi-Joint dynamics with Contact) is selected as the simulation framework due to its high simulation fidelity, computational efficiency, and its convex optimisation-based solver, which ensures superior mathematical precision, making it the preferred environment for continuous control problems [12].

4.1.2 Gymnasium API. The Gymnasium API provides a standardised interface for reinforcement learning environments by separating agent logic from environment dynamics. By wrapping the simulation framework in this abstraction layer, each quadcopter controller can interact with an observation and action space in a consistent manner. This enables direct compatibility with standard RL libraries without exposing the underlying simulation complexity.

4.1.3 BaseController API. The BaseController further abstracts over the general Gymnasium API to conform to quadcopter experiments. To ensure that each controller implementation has access to the same number of parameters, an interface regularises the interactions between controllers and the environment.

4.1.4 Controller implementations. The controller implementations serve as the basis for comparison in this study. Of the six algorithms to be compared, four have already been implemented in the open-source library StableBaselines3. This allows greater focus on implementing controllers with novel algorithms, such as PID, TAD3, and SCR-PPO. Due to the algorithmic simplicity, the PID controller will be implemented in phase 1 to test the framework for further controller implementations.

4.1.5 PID Implementation. A cascade PID controller architecture will be used, with multiple PID controllers arranged in series, with the output of the outer control loop feeding into the inner control loop.

4.1.6 Hyperparameter search. The Optuna framework will be integrated into the tech stack to standardise hyperparameter optimisation (HPO). Optuna will ensure that performance differences can be attributed to an algorithm’s architecture as opposed to disparities in tuning. The specifics of the optimisation strategy and validation process is detailed in Phase 3.

4.1.7 Simulation Specifications. To ensure algorithm comparability, the following environment variables will be defined and frozen before any implementation takes place: the quadcopter model’s rigid-body dynamics, motor thrust non-linearities, and coupled attitude behaviour. The state space of the simulation includes the quadcopter’s position, velocity at each timestamp, and orientation. The continuous motor thrust outputs must be normalised to ensure consistent ranges across tests. The episode termination conditions, such as crash detection, maximum deviation, and maximum episode length, ensure consistent stopping conditions. Lastly, the reward function is shared across all algorithm implementations and the PID baseline to ensure fair comparison. A quadratic cost formulation penalising attitude errors and control effort is selected for the reward function. It produces dense, continuous reward signals that are well-suited for both on-policy and off-policy learning, as its mathematical properties do not inherently advantage either learning type [17].

4.1.8 Quadcopter model. A high-fidelity quadcopter model will be used to fully capture the quadcopter-specific flight dynamics [10]. To achieve this, the project will make use of the *Crazyfly 2.0* model from the GitHub repository, mujoco menagerie by Google DeepMind. This choice allows for precise modelling of rigid-body dynamics, non-linear motor thrust, and the coupled attitude behaviour resulting from the six degrees of freedom [7]. The Gymnasium interface will be used to maintain a consistent agent-environment interaction protocol across all algorithms.

4.2 Phase 2: Algorithm Implementation

StableBaselines3 provided Python implementations for four of the six algorithms being tested, with the novel algorithms (TAD3 and SCR-PPO) requiring manual implementation. All algorithms will be validated in an abstract MuJoCo test before their application in a quadcopter-specific environment.

4.2.1 Off-Policy Algorithms. Off-policy methods are well-suited to the large training volumes required for quadcopter control, as replay buffers improve sample efficiency [7]. SAC and TD3 are the primary off-policy baseline algorithms, having consistently outperformed other variants in higher-dimensional continuous tasks [14]. However, existing comparisons between SAC and TD3 lack domain-specific flight metrics and an inconsistent environment framework, which prevents meaningful cross-study comparisons [7].

TAD3 is included as a novel contribution that addresses the systematic underestimation bias prominent in TD3, which, in quadcopter control, could cause the agent to undervalue corrective manoeuvres and compound drift from the target state [18]. TAD3 demonstrated promise for generic continuous tasks but has not been evaluated in a quadcopter-specific environment, which represents the primary gap for this work stream.

4.2.2 On-Policy Algorithms. The study also considers 3 on-policy reinforcement learning methods: Proximal Policy Optimisation (PPO), Advantage Actor-Critic (A2C), and Stabilising Constraint Regularisation PPO (SCR-PPO). On-policy algorithms learn directly from data generated by the current policy, meaning that once trajectories are collected, they are generally used once and then discarded, rather than being stored and reused. While this approach may lead to lower sample efficiency as training progresses, this trade-off is acceptable in scenarios where stability is the priority, such as in dynamic environments of physics simulations.

The novel SCR-PPO algorithm builds on the PPO clipping objective. Incorporating additional stabilising constraints during the training phase encourages safer, more stable policy behaviour. This algorithm has been implemented in a quadcopter system to mitigate oscillations in rotor speeds, but further comparison is required to evaluate its performance relative to other RL algorithms.

4.3 Phase 3: Hyperparameter Optimisation (HPO)

To ensure the validity of the comparison, each algorithm must undergo equal hyperparameter optimisation to determine the optimal parameter configuration [15]. Bayesian optimisation with Asynchronous Successive Halving (ASHA) will be implemented via the Optuna framework. Bayesian optimisation is sample-efficient and produces static configuration outputs, simplifying reporting and test reproduction [19, 20]. The implementation of ASHA will increase computational efficiency by evaluating trials at intermediate checkpoints and terminating poor-performing trials early, prioritising the compute budget towards runs showing promise [20].

4.3.1 HPO Procedure. Each trial adheres to a fixed number of training timesteps, evaluation episodes, and random seeds, with the mean episodic reward across seeds serving as the objective metric. An initial locally executed exploration is conducted to identify the search space bounds for a comprehensive HPO search on a high-performance computing cluster (see section 7.2). Each algorithm will have an independent Optuna study for its distinct parameter set, with jobs running in parallel to increase efficiency.

Upon completion, the top 5 configurations by mean reward for each algorithm are re-evaluated over 5 random seeds and longer training horizons. The configuration that achieves the highest mean reward and lowest variance across these validation runs will be used as the final configuration for that algorithm. This dual-stage selection process ensures the statistically optimal configuration is selected.

As with RL algorithms, the PID controller will undergo a structured tuning procedure. PID gain values (K_p , K_i , K_d) are tuned using the Ziegler-Nichols method to establish initial values, followed by a grid search over a bounded region around the initialised values [4]. The PID tuning procedure will follow the previously mentioned dual-staged selection process, ensuring a well-tuned controller is created for fair comparison.

Phase 3 outputs a single validated configuration for all controller algorithms, which will remain unmodified for phase 4's final benchmark tests.

4.4 Phase 4: Final Benchmark Experiments and Metrics

All final benchmark runs will be executed on the UCT High Performance Computing Cluster, using the fixed configurations and environment outlined in Phases 1 and 3.

4.4.1 Experimental Protocol. The algorithms will be evaluated across a minimum of 10 random seeds per disturbance, confirmed by a pilot statistical power analysis that will be conducted during phase 1. This is recommended as a minimum for detecting medium to large effect sizes [8]. The simulations will take place in a MuJoCo environment as specified in phase 1. The quadcopter will be evaluated over the following five wind disturbances: no-disturbance baseline, a low-magnitude constant wind, a high-magnitude constant wind, a step gust, and a sinusoidal wind force. The specific magnitudes will be calibrated in phase 1 to ensure the low constant conditions fall in a stable operating range for the PID and the higher magnitude conditions exceeds it operating range, to establish meaningful stress test for the RL algorithms. These conditions are designed to represent real-world atmospheric turbulence. The configurations and results of each algorithm's training runs will be documented to ensure reproducibility across seeds.

A reward sensitivity analysis will be conducted by varying the weighting coefficients of the reward function. Confirming whether the conclusions drawn from the benchmark tests are robust with regard to reward formulations and not an artefact of a particular coefficient configuration [5].

4.4.2 Evaluation Metrics. The chosen domain-specific flight metrics are grouped into three categories:

Solution Quality captures the precision and stability of hover performance. Root Mean Square Error (RMSE) is the mean deviation of the quadcopter's position over the final 10 seconds of a hover episode, measured in meters. Attitude oscillations are measured as the peak-to-peak amplitude of Euler angle variation during steady-state hover, reported in degrees. Additionally, the control smoothness and maximum overshoot will be measured to assess rotor noise and tracking precision [7, 13].

Convergence Reliability measures the consistency of training outcomes across seeds. Success rate is the proportion of seeds that complete a full episode without triggering an early stopping condition, such as the maximum deviation threshold. Convergence speed is determined by the number of environment time steps required to achieve and maintain a reward above a defined threshold for consecutive windows. As noted in section 7.1 Risk 7, the structural sample inefficiency of on-policy methods calls for the separate evaluation of on-policy and off-policy methods to avoid bias [8].

Computational Overhead captures inference latency and memory footprint. Inference latency is measured as the time it takes to produce a single control output, reported in microseconds per control step. The latency of a controller has to fall below the maximum allowed bounds of 1000 to 4000 microseconds per control step [7]. Any algorithm exceeding the threshold is flagged as computationally infeasible for real-world applications, regardless of flight performance.

5 Ethical, Professional and Legal Issues

This project raises no significant ethical concerns as it is conducted entirely in a simulation environment, with no human participation, physical materials or external datasets. The source code will be made openly available via GitHub to promote reproducibility, however, no guarantees against dual-use of findings can be provided.

6 Anticipated Outcomes

This project will produce both a research contribution and a software framework to support further research and experimentation in reinforcement learning for quadcopter stabilisation controls.

6.1 Research

The project will focus on comparative insights into the performance of various RL algorithms for stabilising a quadcopter under simulated conditions. These insights allow for broad comparison of the properties and the trade-offs that occurred.

It is anticipated that a significant contribution to these insights will come from comparing on-policy and off-policy algorithms.

6.2 System

The outcome will also produce a modular reinforcement learning framework for quadcopter control. The broad study of multiple RL algorithms (PPO, A2C, TD3, etc.) necessitates a research flow that is both consistent and flexible for future RL algorithms. The architectural design of the framework will follow SOLID design principles, ensuring that researchers can easily design their own RL algorithms and integrate them into the framework.

6.3 Expected impact of the research

The research conducted in this study aims to achieve two objectives: a broad comparison of data-driven techniques for quadcopter stabilisation, and a standard of comparison between current and future data-driven algorithms. The study aims to break away from many prior studies and, instead of focusing deeply on a few data-driven algorithms for quadcopter stabilisation, provides a broad overview of each algorithm's performance in a standardised environment.

While comparing the algorithms will provide a holistic overview, the approach taken in this study can be generalised to other implementations of the controller for quadcopters. The comparison of the algorithms, therefore, not only serves to inform the performance of each implementation but also presents an opportunity to build a framework for consistently implementing each controller. This resource is useful in research scenarios, where experimentation with variations of algorithms can be applied to the same framework in a consistent manner. In addition, a standard framework enables more comprehensive cross-study analysis for studies that use it.

6.4 Key success factors

This study acts as a union between control theory and reinforcement learning. As such, the success factors used to evaluate it are multi-dimensional:

- (1) **Tracking Error:** how closely the quadcopter follows a desired position or attitude setpoint.
- (2) **Settling time:** how quickly the quadcopter recovers stability after a disturbance.
- (3) **Reward convergence (RL):** cumulative reward increasing and stabilising over episodes.
- (4) **Computational overhead:** how demanding of processing power the controller implementations are.

It is therefore important to select factors that, while they individually describe a property of the experiment, collectively describe the reality under which these systems are evaluated when considering them for applications.

7 Project Plan

The project plan outlines the potential risks, mitigation strategies, milestones, deliverables, work allocation and the proposed timeline for this project.

7.1 Risk Assessment and Mitigation

Listed below are the potential risks that the project is most susceptible to, accompanied by proactive mitigation strategies.

1. Shared Environment Specification Drift

Impact: Different simulation environment specifications leading to incomparable results across algorithms.

Mitigation: Create a shared environment specification document requiring agreement from all parties to edit.

2. Limited TAD3 and SCR-PPO Documentation

Impact: Unavailability of open-source quadcopter-specific TAD3 and SCR-PPO algorithms threatens the feasibility of implementation.

Mitigation: Failure to implement TAD3 or SCR-PPO will lead to omitting the algorithms from the evaluation, as the other comparisons will still constitute a meaningful research contribution.

3. Unavailability of a Team member

Impact: Delayed timeline and the loss of either on or off-policy experimentation.

Mitigation: Develop a shared PID implementation and simulation environment as early as possible and shift the scope of the comparison to only include one learning type.

4. Poor Reward Function Design

Impact: The exploitation of reward loopholes produces high scores but may negatively impact the flight behaviour, voiding the domain-specific evaluation metrics.

Mitigation: Design and test the reward functions against the PID baseline. If the PID can't achieve reasonable rewards, there is an inherent flaw.

5. Project Timeline Overrun

Impact: Rushed evaluations compromise the results and the statistical validity of the conclusion.

Mitigation: Focus efforts on pre-defined minimum viable project requirements, prioritising quality of evaluations as opposed to quantity of algorithm implementations.

6. Hyperparameter Sensitivity Producing Misleading Results

Impact: Performance differences may reflect tuning efforts rather than true algorithmic performance, undermining the benchmarking frameworks.

Mitigation: Use a standardised hyperparameter search protocol for all experiments, combined with multiple random seeds to ensure independence between tuning and performance.

7. On-Policy Sample Inefficiency Causing Asymmetric Training Costs

Impact: PPO and A2C require more environment interaction to converge than off-policy methods, making fair comparisons difficult given time and computational limitations.

Mitigation: Account for this asymmetry, presenting metrics that are not biased towards off-policy algorithms like convergence time.

8. HPC Queue Delays

Impact: Extended HPC queue wait time delays training runs in phases 3 and 4, threatening the start of Phase 5.

Mitigation: The availability of local machinery will serve as a fallback for smaller seed count runs.

7.2 Resources Required

The following computational and software resources are required to successfully implement a comprehensive evaluation.

7.2.1 Computational Resources. The development and testing of the simulation environment and algorithmic setup require local machines to have the following recommended specifications. GPU with 8GB VRAM to handle large batch sizes, 16GB of RAM to store replay buffers and enable parallelisation of training runs. The MuJoCo physics solver runs solely on the CPU; 6+ cores are required to sustain suitable data throughput. The final multi-seed benchmarking test is larger in scale and requires 40GB VRAM per GPU to provide headroom for multiple seeds running on each node. Due to parallel execution on a node sharing memory, 64GB of RAM is required for efficient parallelisation, along with 8-16 cores per CPU. We have identified the UCT High Performance Computing Cluster (HPC), specifically the A100 or L40S GPU partitions [3], to be sufficient for the final simulation runs. Access has been requested for UCT HPC use and is pending approval.

7.2.2 Software Resources. The proposed research relies on a set of open-source software tools. GitHub version control will be used [1], alongside cloud-hosted storage for experiment data to manage, track, and store data. MuJoCo will be used for the physics simulation [2], paired with Stable Baselines 3 and Optuna for RL algorithm implementation and parameter optimisation, respectively. The Docker container framework will be used to make the source code executable via SLURM on the University of Cape Town HPC [3], ensuring that the local setup environment matches the HPC environment and ensuring reproducibility. The data visualisation will require statistical analysis and visualisation libraries.

7.3 Timeline

The project timeline (Gantt Chart) is included in Appendix A, and is split into phases as described in section 4.

7.4 Deliverables

The deliverables that are due throughout the lifetime of the project are listed in Table 3 accompanied by the due dates.

Deliverable	Due Date (2026)
Project Proposal Draft	24 April
Project Proposal Presentation	28–30 April
Project Proposal Final Submission	5 May
Project Progress Demonstration	27 July
Project Paper Scaffold	10 August
Full Draft of Final Paper	24 August
Project Paper Final Submission	4 September
Code Final Submission	14 September
Final Project Demonstration	16–23 September
Poster	28 September
Website	5 October

Table 3: Project Deliverables and Due Dates

7.5 Milestones

The internal milestones in Table 4 mark key artefacts produced in each phase.

Phase	Milestone	Description
1	PID Stable Hover Confirmed	PID implemented and validated as a reliable performance baseline
1	Environment and Reward Frozen	MuJoCo environment and reward function confirmed to produce valid simulations
2	All Algorithms Stable	SAC, TD3, TAD3, PPO, A2C and SCR-PPO validated on benchmark tasks and quadcopter environment
3	All Configurations Identified	Optimal hyperparameter configurations identified for all algorithms via Bayesian search
3	Progress Demonstration	All algorithms implemented and tuned, ready for final benchmark runs
4	All Results Collected	Multi-seed HPC benchmark runs completed and documented for all algorithms
4	Statistical Comparisons Completed	Results analysed and conclusions drawn across all algorithms and disturbance conditions

Table 4: Project Milestones

7.6 Work Allocation

The work allocation in Table 5 is structured to remove co-dependence between group members and their work.

8 AI usage statement

Claude (Anthropic) was used as a generative AI tool during the preparation of this proposal. Its use was strictly limited to the

Partner	Responsibilities
Both	MuJoCo quadcopter simulation setup and specifications, baseline PID implementation, reward function and evaluation framework design, HPC configuration for test runs, statistical analysis of the PID results
Wietske	Implementation and benchmarking of off-policy algorithms SAC, TD3, and TAD3, implementation of a custom TAD3 algorithm
Mathew	Implementation and benchmarking of on-policy algorithms PPO and A2C, SCR-PPO, implementation of the SCR-PPO algorithm

Table 5: Work Allocation

purposes permitted under the assignment guidelines. Specifically, Claude was used to provide critical feedback on argument structure, identify weaknesses in sections, and suggest improvements to clarity and organisation across all sections of the proposal. Claude was also used to assist with planning the methodology, risk assessment, and project plan sections, and to provide feedback on the appropriate level of technical detail for the project proposal.

All writing in this proposal was produced by the authors. AI-generated suggestions were critically evaluated, substantially rewritten, and adapted before inclusion. No AI-generated text was copied directly into the document. The final content, arguments, and decisions reflect the authors' own understanding and judgment.

References

- [1] 2026. GitHub · Change is constant. GitHub keeps you ahead. <https://github.com/>
- [2] 2026. MuJoCo — Advanced Physics Simulation. <https://mujoco.org/>
- [3] 2026. UCT HPC University of Cape Town High Performance Computing. <https://ucthpc.uct.ac.za/>
- [4] Anagha Philip Antony and Elizabeth Varghese. 2016. Comparison of performance indices of PID controller with different tuning methods. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. IEEE, Nagercoil, India, 1–6. doi:10.1109/ICCPCT.2016.7530307
- [5] Muhammad Azam and Hossein Salehfar. 2025. Reward Function Sensitivity Analysis in RL Based PMSM Control. In *2025 5th International Conference on Electrical, Computer and Energy Technologies (ICECET)*. 1–5. doi:10.1109/ICECET63943.2025.11472470
- [6] Ahmad Taher Azar, Anis Koubaa, Nada Ali Mohamed, Habiba A Ibrahim, Zahra Fathy Ibrahim, Muhammad Kazim, Bilel Benjdira, Adel Ammar, Ibrahim A Hameed, and Gabriella Casalino. [n. d.]. Drone Deep Reinforcement Learning: A Review. ([n. d.]).
- [7] Nicola Bernini, Mikhail Bessa, Rémi Delmas, Arthur Gold, Eric Goubault, Romain Pennec, Sylvie Putot, and François Sillion. 2021. A few lessons learned in reinforcement learning for quadcopter attitude control. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. ACM, Nashville Tennessee, 1–11. doi:10.1145/3447928.3456707
- [8] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. 2018. How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments. doi:10.48550/arXiv.1806.08295 arXiv:1806.08295 version: 2.
- [9] Nourhan Elmeseiry, Nancy Alshaer, and Tawfik Ismail. 2021. A Detailed Survey and Future Directions of Unmanned Aerial Vehicles (UAVs) with Potential Applications. *Aerospace* 8, 12 (Nov. 2021), 363. doi:10.3390/aerospace8120363
- [10] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. doi:10.48550/ARXIV.1802.09477
- [11] Fang-I Hsiao, Cheng-Min Chiang, and Alvin Hou. 2019. Reinforcement Learning Based Quadcopter Controller. (2019).
- [12] Michael Kaup, Cornelius Wolff, Hyerim Hwang, Julius Mayer, and Elia Bruni. 2024. A Review of Nine Physics Engines for Reinforcement Learning Research. doi:10.48550/arXiv.2407.08590 arXiv:2407.08590 version: 2.
- [13] Parsa Yeganeh Khorasani. 2025. Using a proportional integral derivative (PID) system to control the quadrotor drone to ensure its reliability and robustness. doi:10.21203/rs.3.rs-6448814/v1
- [14] Shijie Liu. 2024. An Evaluation of DDPG, TD3, SAC, and PPO: Deep Reinforcement Learning Algorithms for Controlling Continuous System. In *Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)*, Badrul Hisham Ahmad (Ed.). Vol. 180. Atlantis Press International BV, Dordrecht, 15–24. doi:10.2991/978-94-6463-370-2_3
- [15] Nicolai A. Lynnerup, Laura Nolling, Rasmus Hasle, and John Hallam. 2020. A Survey on Reproducibility by Evaluating Deep Reinforcement Learning Algorithms on Real-World Robots. In *Proceedings of the Conference on Robot Learning*. PMLR, 466–489. <https://proceedings.mlr.press/v100/lynnerup20a.html>
- [16] Abhishek Manalathody, Kishoor Santhana Krishnan, Jisnu Adindra Subramanian, Sakthivel Thangavel, Jinraj V. Pushpangathan, and Hariprasanth Paranjothy. 2025. Nonlinear and Linear PID Controllers-Based Hybrid Flight Control Strategy for a Quadcopter With Slung Load. *IEEE Access* 13 (2025), 48520–48542. doi:10.1109/ACCESS.2025.3551622
- [17] Syed Ali Asad Rizvi and Zongli Lin. 2020. Reinforcement Learning-Based Linear Quadratic Regulation of Continuous-Time Systems Using Dynamic Output Feedback. *IEEE Transactions on Cybernetics* 50, 11 (2020), 4670–4679. doi:10.1109/TCYB.2018.2886735
- [18] Teckchai Tiong, Ismail Saad, Kenneth Tze Kin Teo, and Herwansyah Bin Lago. 2020. Deep Reinforcement Learning with Robust Deep Deterministic Policy Gradient. In *2020 2nd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*. IEEE, Kuala Lumpur, Malaysia, 1–5. doi:10.1109/ICECIE50279.2020.9309539
- [19] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. 2021. Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track (Proceedings of Machine Learning Research, Vol. 133)*, Hugo Jair Escalante and Katja Hofmann (Eds.). PMLR, 3–26. <https://proceedings.mlr.press/v133/turner21a.html>
- [20] Tong Yu and Hong Zhu. 2020. Hyper-Parameter Optimization: A Review of Algorithms and Applications. doi:10.48550/ARXIV.2003.05689

Appendix A: Project Timeline

