Simulation-Based Reinforcement Learning for Novel Hybrid Vehicle Autonomous Landing

Will Borges, Preston Burnside, John Lagarde, Scott Wolter, and Blake Hament

***Abstract*—This study explores the integration of simulated reinforcement learning (RL) and a novel hybrid vehicle design to achieve complex physical tasks. Specifically, the team aimed to perform an autonomous landing of a rocket-like device, inspired by the SpaceX Falcon 9 booster landings. The RL agent was first trained within a physics-based simulation environment using a subset of RL called Proximal Policy Optimization (PPO), allowing the agent to learn how to stabilize and execute a controlled descent by managing four motors and four aerodynamic fins. The trained agent was then deployed to a custom-built rocket prototype designed to emulate the dynamics of a rocket using brushless motors and onboard sensors. By bridging simulation-based learning with real-world implementation, this study evaluated the effectiveness of RL for autonomous control compared to traditional methods. In the current state of this project, the researchers observed a ~32% successful landing rate with another 35% in "soft impact" failed attempts over 350,000+ simulations. Future work will focus on improving these results by moving to a different simulation method that will allow us to have more processing power available to go towards larger training sets at a quicker rate. The team also plans to refine their physical design to improve stability and efficiency. These results contribute to the growing field of autonomy in aerospace and robotics, highlighting both the potential benefits and the challenges of applying reinforcement learning to complex dynamic systems.**

I. INTRODUCTION

Machine learning has emerged as a powerful tool for enabling autonomous decision-making in complex and dynamic systems. Among its various approaches, reinforcement learning (RL) has proven particularly effective for control tasks, as it allows an agent to learn optimal behaviors through iterative interaction with an environment and feedback in the form of rewards. Unlike traditional control methods that rely on explicit modeling and predefined rules, RL enables systems to adapt to uncertainty and nonlinearity through experience. This capability has led to its increasing application in domains such as robotics, aerospace, and autonomous vehicles, where real-time decision-making and adaptability are critical. In this study, reinforcement learning was applied to the problem of autonomous landing for a rocket-inspired hybrid vehicle system, with a focus on evaluating the effectiveness of simulation-based training for real-world deployment.

This study was based upon extensive research on machine learning and reinforcement training algorithms. For guidance within the realm of reinforcement learning, the team turned to Deepanshu Mehta's "State-of-the-Art Reinforcement Learning Algorithms," [1] which details the equations and processes of current training procedures. These algorithms boil down to a basic feedback loop, where a virtual agent takes an action within a simulated environment, receives weighted feedback according to the parameters through which it is being evaluated. Once it receives this feedback, it re-attempts the task to maximize positive feedback and minimize negative results.

Another valuable resource was Kolla Bhanu Prakash's "Computational Intelligent Techniques in Mechatronics" [2], a collaborative book recently published in 2024, that combines several ideas about mechatronics and implementing machine learning in one succinct document.

Another paper that probed this topic is one by a student at the University of Zilina, Slovakia. In Martin Bohušík's paper, "Mechatronic Device Control by Artificial Intelligence" [3], he displays how machine learning can be used to find an approximate function to describe the kinematics of an "agile eye" mechanism to calibrate a device's motors accordingly. He used a neural network that, once trained, could accept manual inputs to control the device. Neural networks like these have been proven useful in an extremely wide range of applications, such as autonomous vehicles.

There is a lot of overlap between how autonomous vehicles are trained, which was explained in depth in "Autonomous Driving Architectures: Insights of Machine Learning and Deep Learning Algorithms" by Mrinal R. Bachute and Javed M. Subhedar [4], and the virtual agent that the team developed. Both used some form of path finding and reinforcement learning, and both were intended to have the capacity for fully autonomous control (though there are varying levels of this in autonomous vehicles).

The biggest difference between the two was the set of parameters in which they operate; both encouraged the idea that such neural networks and reinforcement learning algorithms can be used for a multitude of complex tasks. All together, these resources provided a foundation

for the model that the team developed. However, we expanded these concepts by using reinforcement learning with the goal of the agent having complete control of the device with no human input.

With this project, the team wished to demonstrate that neural networks trained entirely in a virtual environment can be implemented seamlessly into a real-world physical device. Such work is necessary to investigate whether simulation-trained models can perform effectively in real-world conditions and the capability of neural networks to handle complex, dynamic tasks currently managed by human operators.

The team aimed to evaluate whether training a machine learning model in a virtual simulation is viable for achieving the desired performance in a complex physical task. Specifically, the model was trained to autonomously land a device inspired by SpaceX's Falcon-9 booster. It shows how reinforcement learning provides a robust framework for training intelligent agents through experience, allowing them to adapt and improve in dynamic environments without explicit programming for every situation. The team wanted to show that that author's research underscores the critical importance of simulated environments as safe and efficient proving grounds for developing and refining complex control systems before real-world deployment, saving time, resources, and potentially preventing costly or dangerous failures. Finally, the teams' work aimed to validate "Sim-to-Real" transfer showing that simulation-trained models are effective in physical systems.

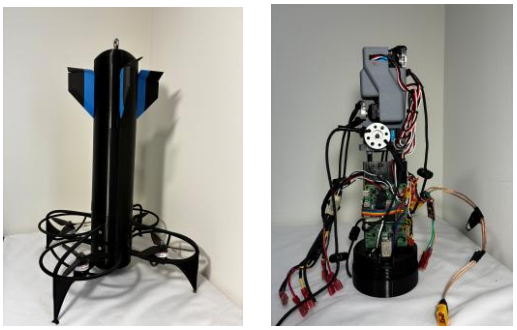
II. HYBRID VEHICLE DEVICE DESIGN

The device that was created was based on the SpaceX Falcon 9. The team modified the device to be propelled by brushless motors and stabilizing fins so that we could repeatedly test and conduct trials without having to constantly replace rocket propulsion systems. We trained a machine learning model to control the power of these motors based on inputs from several included sensors. This model utilizes brushless motors and servos to adjust the position and orientation of the hybrid device, such that it lands upright with minimal forces from maneuvers or touch down.

The physical model, while imitating a rocket, is closer in function to a quadcopter drone given the designs tried and true stability and versatility. It differs, however, in its movement patterns, maneuvers, and overall construction to more closely simulate points of thrust on a rocket. It includes servo-controlled fins for aerodynamic adjustment and stabilization. It also houses the internals such that the weight distribution and center of mass is closer to that of a rocket on re-entry.

A. Fabrication of the Surrogate Rocket Booster

The team decided to implement drone motors as a reusable method of flight. The consequence of this change results in a less-than-ideal simulation of a rocket flight. Drones stabilize themselves purely by modifying the power to certain propellers, resulting in the aircraft's roll, pitch, and yaw. However, given a different flight pattern closer to the flight pattern of a rocket, this design implemented greater control through servo-controlled fins as opposed to the traditional propeller-exclusive quadcopter control.



(a) (b)

Fig. 1: Structural design of the surrogate rocket booster showing (a) the outer frame with motor mounts and grid fins, and (b) the internal layout including the Raspberry Pi, battery system, and servo mounting configuration.

B. Fabrication of the Light Control Surfaces

The team needed to collect as much data as we could about the orientation and location of their rocket's flight to train the model as accurately as possible. To do this, we needed to include several sensors in the rocket, consisting of an accelerometer, altimeter, ultrasonic distance sensor, GPS sensor, and gyroscope. These are each standalone sensors, which in turn relay information to a microcontroller loaded with the completed software model that reacts to said information.

The team needed to determine specifics about the drag, thrust, aerodynamics, and mass properties of the rocket. This information was then used during the simulations for training the model and was obtained through CAD programs such as SolidWorks by running aerodynamic testing on the model rocket, or by designing testing physical rigs that extracted data from the electronics or parts. With some final mass calculations and assessments, the team determined the thrust and acceleration limits of the model and translated that to the software environment for training the virtual agent.

III. MACHINE LEARNING METHODOLOGY

This study used a subset of Machine Learning (ML) called "Reinforcement Learning" (RL) specifically Proximal Policy Optimization (PPO) that trained the model based on experienced trial and error. In Practice, the PPO algorithm is composed of three main parts: The Agent, Environment, and Reward Signal. The Agent: The neural network responsible for making decisions for what actions to take. The agent in this project uses inputs from sensors on the device and outputs decisions on how much thrust for the motors and the angles of the four fins on the device to land safely. Our agent is composed of 12 inputs, 512 nodes by 3 layers, and 8 outputs.

TABLE I
Reinforcement Learning Hyperparameters

Hyperparameter	Assigned Value
Learning Rate (η)	0.00001
Decay Rate (β)	0.001
Exploration Rate (ε)	0.2
Regularization Factor (λ)	0.95
Discount Factor (γ)	0.995
Number of Layers	3
Size of Layers	512

The Environment: Where the training takes place, the actions that the agent takes influence the environment and vice versa. The teams' environment approximates the physics and sensor information including natural noise modeled by the sensors present in real-world situations. We also needed to simulate kinematics including approximate aerodynamic forces, random wind variances, and center of mass. The Reward Signal: Evaluation of the actions that the agent takes. The team' reward function was tailored to encourage actions that result in a landing. We based the reward function on the position of the rocket, closer to the ground is a higher reward and closer to a given x,z coordinate for precision. We also involved negative rewards for higher velocity scaled by how close the agent is to the ground; this discouraged crashing while allowing for quicker approaches before landing. Finally, we gave rewards based on the angle of the rocket itself as well as its angular velocity, this encouraged the rocket to be facing opposite its velocity, encourage the rocket to become upright, as well as encouraging to avoid unnecessary alignment patterns. We went through many iterations of the reward function watching how the agent behaved. We learned what factors of the reward function are being overly corrected for vs ignored by the agent. We would then adjust the weights of the reward function accordingly until we started seeing improvements throughout the training.

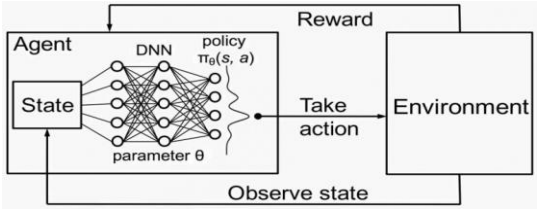


Fig. 2: Reinforcement learning framework illustrating the interaction between the agent, environment, and reward signal. Source: [5]

Reinforcement learning was utilized through computer-simulated scenarios to train the machine learning model to achieve a controlled landing. The benefit of training the model in a virtual environment instead of the physical world was that the team was able to perform a lot more training sessions in a relatively short amount of time. This allowed the agent (the simulated hybrid vehicle device) to see a lot more scenarios than would have been possible otherwise. An important aspect of the training was the reward function, as it was how the quality of the decisions made by the agent was judged and adjusted. An important aspect of this was the reward function's relation to time. The team needed to optimize the reward function through trial and error to encourage the agent to value decisions that improved the chance of landing in the long term, which included sacrifices in the short term. The team used the Unity Game Engine for the testing grounds for the simulation, as Elon University provides students with keys to download and utilize the software. Scripting in Unity is done using C#, so that is the code we used for all of the physics and logic of the simulation. We were able to use the Pytorch library which is known for handling machine learning functions.

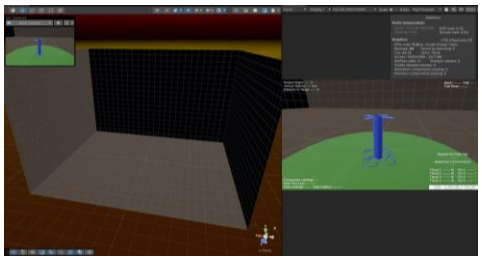


Fig. 3: Simulation environment in Unity used for training the reinforcement learning agent, showing (left) the terrain boundary conditions, (center) the tracking camera following the vehicle, and (right) the inspector panel for configuring simulation parameters.

To encourage learning, the simulation started simply with flat terrain, no wind, and a less intense trajectory. Gradually, the situations that the agent is put in became more complex as the reward function determined the agent's landing proficiency had increased. The simulation made greater variability in the bumpiness of the terrain, wind strength, initial velocity and orientation, and final target landing of the agent. This allowed the agent to learn and discover how to orient itself and land before concerning itself with the imperfect factors of a real-world environment.

TABLE II
Evaluation Results at 25,000-Trial Checkpoints

Trial Checkpoint	Trials in Following Session	Number of Successful Landings	Number of Soft Impacts
0	0	0	0
25,000	3872	0	4
50,000	4192	4	6
75,000	3984	44	56
100,000	4357	139	179
125,000	4013	130	173
150,000	4688	478	558
175,000	4231	419	448
200,000	3964	543	610
225,000	4203	1063	1210
250,000	4027	1120	1228
275,000	4756	1384	1536
300,000	2881	919	985
325,000	4162	1340	1473

Throughout our data, we were able to see a very positive trend in our data. Initially, we didn't see large improvements in our success rate until around 75,000 training trials. We measured this with both soft impacts and successful landings. The data shows a slight plateau after 250,000 trials, up until we concluded our data.

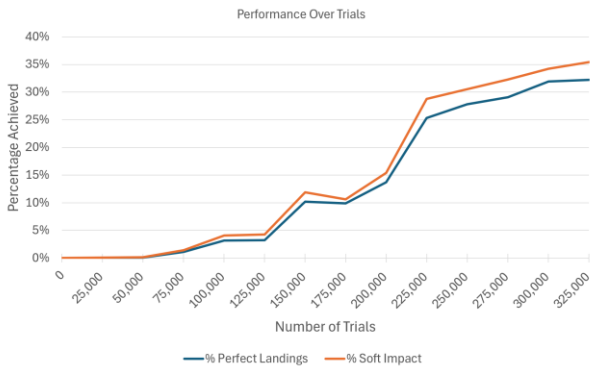


Fig. 4: Model rocket landing success rate versus number of simulated trials, showing performance improvement and distinguishing between perfect upright landings and survivable soft impact landings.

IV. BASELINE CONTROLS USING CLASICAL PID METHODS

To provide a baseline for comparison with the reinforcement learning (RL) approach, a traditional proportional–derivative (PID) controller was implemented and evaluated within the simulation environment. PID control is widely used in aerospace and mechatronic systems due to its simplicity, interpretability, and effectiveness when system dynamics are reasonably well understood. Unlike reinforcement learning, which develops control policies through experience, PID control relies on predefined gain parameters to regulate system behavior based on error feedback.

The rocket is dynamically modeled as a uniform slender rod, with rotational dynamics modeled in terms of angular acceleration, given by the equation,

$$\ddot{\theta} = \frac{r_m \cdot u}{I} (1)$$

where r_m is the radial distance from the motor to the center of mass and u is the thrust input. Because the center of mass does not coincide with the geometric center of the rocket, inertia is modeled as

$$\frac{1}{12} mL^2 + md^2 \quad (2)$$

where L is the length of the rocket, and d is the distance between the center of mass and the geometric center. With the dynamic model established, taking the Laplace transform yields us with the plant transfer function.

$$G(s) = \frac{r_m/I}{s^2} \quad (3)$$

Substituting our rocket parameters gives us the final transfer function.

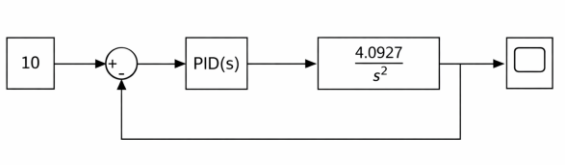


Fig. 5: Closed-Loop PID Control System with Second-Order Plant

The controller was implemented using a parallel PID structure in Simulink, as shown in Figure 5. The gains were tuned empirically to achieve stable and responsive behavior, resulting in proportional, integral, and derivative values of P= 1.888, I = 0, and D = 3.42. The integral term

was intentionally omitted to avoid windup effects and instability during rapid dynamic changes in descent, as well as to prioritize transient response over steady-state error correction. A derivative filter coefficient was also included to reduce sensitivity to high-frequency noise.

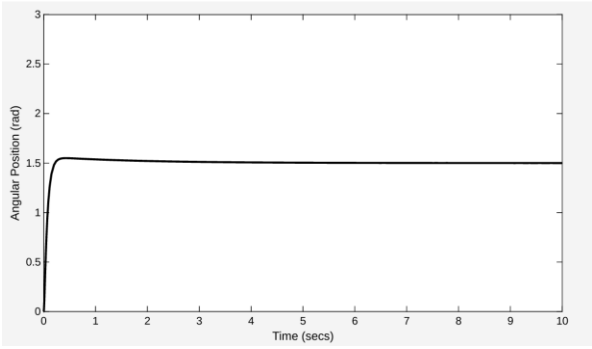


Fig. 6: Step response of the system under PID control, illustrating fast rise time, minimal overshoot, and stable steady-state behavior following a unit input.

The above figure illustrates the step response of the system under PID control. The response demonstrates a fast rise time with minimal overshoot and smooth convergence to a steady-state value, indicating that the controller is capable of producing stable behavior under nominal conditions. The derivative component contributes to damping, reducing oscillatory behavior, and improving overall system stability during transient phases of motion.

While the PID controller performed effectively in controlled simulation scenarios, its performance is inherently limited by its reliance on fixed gain values and an assumed system model. In the context of the surrogate rocket landing task, which involves nonlinear dynamics, external disturbances such as wind, and sensor noise, the PID controller lacks the adaptability required to maintain consistent performance across varying conditions. In contrast, the reinforcement learning approach seeks to learn these dynamics directly from interaction with the environment, offering the potential for more robust and adaptive control in complex, real-world scenarios.

V. SURROGATE BOOSTER LANDING EVALUATION

The evaluation of the surrogate booster system consisted of both simulation-based performance analysis and a single full-scale physical test flight. In simulation, the reinforcement learning agent demonstrated partial success, achieving approximately a 32% successful landing rate after 350,000+ simulated trials, with an additional 35% classified as “soft-impact” failures (meaning hitting the ground at a velocity below 1.5 m/s). While these results indicate that the agent is beginning to learn meaningful control strategies, they remain below the threshold required for reliable real-world deployment.

To contextualize the performance of the reinforcement learning controller, a baseline comparison was made using a classical PID control approach. The PID controller demonstrated stable and predictable behavior under nominal simulation conditions, with fast response and minimal oscillation. However, its performance degraded when subjected to disturbances, nonlinear dynamics, and variability in initial conditions, as it relied on fixed gain parameters and an assumed system model. In contrast, the reinforcement learning controller, while less consistent overall, exhibited a greater capacity to adapt to varying conditions encountered during simulation. These observations highlight a key tradeoff between the reliability and interpretability of traditional control methods and the adaptability of learning-based approaches.

A key limitation encountered during this project was the time-intensive nature of training the reinforcement learning model. Despite completing over 350,000 simulation iterations, the agent had not converged to a consistently successful policy. The computational demands of the training process limited the ability to further refine the model within the project timeline. As a result, the trained policy lacked the robustness necessary to handle variability and uncertainty in a physical environment.

To assess real-world applicability, a single test flight of the physical system was conducted. The system did not achieve a successful landing, with performance issues arising from a combination of factors. These included insufficient model accuracy, discrepancies between simulated and real-world dynamics (commonly referred to as the “sim-to-real gap”), and mechanical or control inconsistencies within the physical platform. Additionally, the limited number of physical tests restricted opportunities for iterative debugging and refinement.

The observed gap between simulation performance and real-world results highlights the challenges of transferring reinforcement learning models to physical systems. Environmental disturbances, sensor noise, and unmodeled dynamics all contributed to reduced system performance. These findings emphasize the need for improved simulation fidelity, more extensive training, and incremental real-world validation before full deployment.

Despite these limitations, the evaluation process provided valuable insights into both the capabilities and current shortcomings of simulation-based reinforcement learning for autonomous control. The results demonstrate that while the approach is promising, significant

development is still required to achieve reliable real-world operation.

VI. SUMMARY

This project explored the application of reinforcement learning to the autonomous landing of a rocket-inspired hybrid vehicle device. By training an agent in a simulated environment and attempting to deploy it on a physical platform, the team aimed to evaluate the feasibility of simulation-based learning for complex, real-world control tasks.

While the system design and simulation framework were successfully developed, the overall performance did not meet the criteria for consistent real-world success. The reinforcement learning model, though partially effective in simulation, required significantly more training and refinement to achieve reliable results. The single physical test flight further demonstrated the challenges associated with sim-to-real transfer, as the system was unable to perform as intended under real-world conditions.

One of the most important findings of this work is the substantial gap between simulated training environments and physical implementation. Factors such as environmental variability, hardware limitations, and sensor inaccuracies introduced complexities that were not fully captured in the simulation. Additionally, the computational cost and time required for effective reinforcement learning training proved to be a major constraint.

Future work will focus on improving simulation fidelity, optimizing the training process, and increasing the robustness of the learned control policies. Incorporating domain randomization, enhancing sensor modeling, and conducting incremental subsystem testing are potential approaches to reduce the sim-to-real gap. Further physical testing will also be essential to validate improvements and iteratively refine the system.

Although the project did not achieve a fully successful autonomous landing, it represents a meaningful step toward integrating reinforcement learning into aerospace and mechatronic applications. The lessons learned provide a strong foundation for future research and development, and highlight both the potential and the current limitations of applying machine learning to complex dynamic systems.

VII. REFERENCES

[1] D. Mehta, "State-of-the-art reinforcement learning algorithms," *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 12, Dec. 2019. [Online]. Available: https://www.ijert.org/research/state-of-the-art-reinforcement-learning-algorithms-IJERTV8IS120332.pdf.

[2] Kolla Bhanu Prakash, Satish Kumar Peddapelli, Ivan, Wai Lok Woo, and V. Jain, *Computational Intelligent Techniques in Mechatronics*. John Wiley & Sons, 2024.

[3] M. Bohušík *et al.*, "Mechatronic Device Control by Artificial Intelligence," *Sensors*, vol. 23, no. 13, pp. 5872–5872, Jun. 2023, doi: https://doi.org/10.3390/s23135872.

[4] M. R. Bachute and J. M. Subhedar, "Autonomous Driving Architectures: Insights of Machine Learning and Deep Learning Algorithms," *Machine Learning with Applications*, vol. 6, p. 100164, Sep. 2021, doi: https://doi.org/10.1016/j.mlwa.2021.100164.

[5] V. V. PV, "Deep Reinforcement Learning: Artificial Intelligence, Machine Learning and Deep Learning —…," *Medium*, Aug. 03, 2020. https://medium.com/@vishnuvijayanpv/deep-reinforcement-learning-artificial-intelligence-machine-learning-and-deep-learning-e52cb5974420

ACKNOWLEDGMENT

The authors acknowledge the Clark Prototyping Lab at Elon University for their assistance with the design and manufacture of the Halbach cylinder.