Controlling a rigid body quadcopter using Control Theory and Reinforcement Learning

A Report Submitted in Partial Fulfilment of the Requirements for the

SN Bose Internship Program, 2024

Submitted by

Utkarsh Rajput

Under the guidance of

Dr. Jupitara Hazarika
Assistant Professor
Department of Electronics & Instrumentation Engineering
National Institute of Technology Silchar

Department of Electronics & Instrumentation Engineering
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR
Assam

June-July, 2024

DECLARATION

Controlling a rigid body quadcopter using Control Theory and Reinforcement Learning

We declare that the art on display is mostly comprised of our own ideas and work, expressed in our own words. Where other people's thoughts or words were used, we properly cited and noted them in the reference materials. We have followed all academic honesty and integrity principles.

Utkarsh Rajput Scholar ID: 2115066

Department of Electronics and Instrumentation Engineering
National Institute of Technology Silchar, Assam

ACKNOWLEDGEMENT

We would like to thank our supervisor, Dr. Jupitara Hazarika, EIE, NIT Silchar, for her invaluable direction, encouragement, and assistance during this project. Her helpful suggestions for this entire effort and cooperation are gratefully thanked, as they enabled us to conduct extensive investigation and learn about many new subjects.

Our sincere gratitude to Dr. Jupitara Hazarika for her unwavering support and patience, without which this project would not have been completed properly and on schedule. This project would not have been feasible without her, from providing the data sets to answering any doubts we had to helping us with ideas whenever we were stuck.

Utkarsh Rajput Scholar ID: 2115066

Department of Electronics and Instrumentation Engineering
National Institute of Technology Silchar, Assam

ABSTRACT

The SACopters project explores the development and comparison of control strategies for quadcopters within a simulated environment created using Pygame. The primary goal was to evaluate traditional control methods, such as PID controllers, against modern reinforcement learning algorithms like Soft Actor-Critic (SAC). The simulation tasked a quadcopter with collecting target points, represented as balloons, to assess the effectiveness of each control strategy. While PID controllers provided stability and predictability, they were outperformed by the SAC agent, which demonstrated superior adaptability and efficiency in dynamic scenarios. The project involved a collaborative effort where each team member contributed to different aspects, including the development of the PID, SAC, and Human control strategies, as well as the integration of models and assets. The findings suggest that while traditional methods are reliable, machine learning offers significant advantages in complex environments, though it requires extensive training and computational resources. This project lays the foundation for future work in UAV control, including the potential implementation of Deep Q-Networks (DQN) and the exploration of hybrid control strategies.

Department of Electronics and Instrumentation Engineering
National Institute of Technology Silchar, Assam

List of Tables

List of Figures

Contents

Chapter 1

Introduction

1.1 Problem Statement

Controlling trajectories of a UAV via Machine Learning Algorithms

1.2 Objective

To develop a program to control trajectories of a UAV simulated in Pygame via various traditional control strategies and comparing them with Machine Learning algorithms such as SAC, DQN to realize an optimized control strategy that can be used to maneuver a quadcopter.

SACopters is a comprehensive project focused on exploring and comparing different control strategies for Unmanned Aerial Vehicles (UAVs) through the simulation of quadcopter flight in a 2D environment using Pygame. The primary objective of the project is to evaluate various control strategies, including traditional methods like PID (Proportional-Integral-Derivative) controllers and modern Machine Learning algorithms such as Soft Actor-Critic (SAC), in order to determine the most efficient approach for controlling the trajectory of a quadcopter. The simulation is designed to test these strategies in a controlled environment, where the goal is to collect as many target points (balloons) as possible within a set time limit.

Chapter 2

Methodology

2.1 Simulation Environment

The SACopters project is built around a 2D simulation environment developed using Pygame. This environment simulates the physical dynamics of a quadcopter navigating towards randomly generated waypoints, represented as balloons. The simulation environment is designed to mimic real-world physics, including gravitational forces, thrust, and rotational dynamics. The environment provides a platform for testing and evaluating different control strategies under consistent conditions.

2.2 Control Strategies

The performance of each control strategy is evaluated based on the number of balloons collected within a fixed time frame [3]. Three distinct control strategies were implemented within the simulation environment:

- **Manual Control:** A human player directly controls the quadcopter using keyboard inputs. This control method serves as a baseline for evaluating the performance of the autonomous control strategies.
- **PID Control:** The PID controller is a traditional control strategy that uses feedback loops to maintain desired positions or speeds. The PID controller in this project adjusts the quadcopter's thrust based on the error between its current position and the target position, aiming to minimize this error over time.
- **SAC (Soft Actor-Critic):** SAC is a reinforcement learning algorithm that learns an optimal policy through interaction with the environment. The SAC agent in this project is trained to control the quadcopter by maximizing a reward function, which incentivizes collecting balloons and avoiding crashes.

Chapter 3

Theoretical Framework

3.1 Control Strategies Implemented

3.1.1 Manual Control

The manual control strategy relies on human input via keyboard commands. The player controls the quadcopter by adjusting the thrust of its rotors using the arrow keys. This strategy, while intuitive, is prone to human error and is used as a baseline to compare the performance of the autonomous control strategies.

3.1.2 PID Control

The PID controller is a widely used control strategy in various engineering applications. In this project, the PID controller adjusts the thrust of the quadcopter's rotors based on the positional error between the quadcopter and its target [1]. The controller uses three terms—Proportional, Integral, and Derivative—to calculate the appropriate adjustments needed to minimize this error over time. The PID controller provides stable and predictable control, but may struggle with complex or dynamic environments.

3.1.3 SAC (Soft Actor-Critic) Control

SAC is a state-of-the-art reinforcement learning algorithm that optimizes the control of the quadcopter by learning from interactions with the environment. The SAC agent is trained using a reward function that encourages the collection of balloons and penalizes crashes.[3] Over time, the agent learns the optimal policy for controlling the quadcopter, resulting in highly adaptive and efficient control.[5]

3.2 Key Code Snippets

3.2.1 Main Game Loop (balloon.py)

The main game loop is the core of the simulation, handling everything from initializing the game and loading assets to updating the quadcopter's position and checking for collisions with balloons. The loop continuously updates the game state based on the control inputs and the physics simulation.

Python

```
while True:
    pygame.event.get()
    for player_index, player in enumerate(players):
        if player.dead == False:
            player.x_acceleration = 0
            player.y_acceleration = gravity
            # Calculate position
            player.x_position += player.x_speed
            player.y_position += player.y_speed
            player.angle += player.angular_speed
        ...
```

3.2.2 PID Control (player.py)

The PID control algorithm is implemented to adjust the quadcopter's thrust based on the error between its current position and the target position. The algorithm uses the Proportional, Integral, and Derivative terms to compute the necessary adjustments.

Python

```
def act(self, obs):
    error_x, xd, error_y, yd, a, ad = obs
    ac = self.xPID.compute(-error_x, self.dt)
    error_a = ac - a
    action1 = self.aPID.compute(-error_a, self.dt)
    thruster_left += action0 * self.thruster_amplitude
    thruster_right += action0 * self.thruster_amplitude
    ...
```

3.2.3 SAC Agent Implementation (env_SAC.py)

The SAC agent is a reinforcement learning model that controls the quadcopter by learning from a reward function [4] [2]. The agent's policy is optimized through training, allowing it to make decisions that maximize the reward, such as collecting balloons and avoiding crashes.

Python

```
def step(self, action):
    ...
    self.reward = 0.0
    for _ in range(5):
        self.time += 1 / 60
        if self.mouse_target is True:
            self.xt, self.yt = pygame.mouse.get_pos()
        # Calculate accelerations
        self.xdd = 0
        self.ydd = gravity
        self.add = 0
        thruster_left = self.thruster_mean
        thruster_right = self.thruster_mean
        thruster_left += action0 * self.thruster_amplitude
        thruster_right += action0 * self.thruster_amplitude
        thruster_left += action1 * self.diff_amplitude
        thruster_right -= action1 * self.diff_amplitude
        ...
        if dist < 50:
            self.xt = randrange(200, 600)
            self.yt = randrange(200, 600)
            self.reward += 100
        elif dist > 1000:
            self.reward -= 1000
            done = True
            break
```

Chapter 4

Comparative Analysis

4.1 Performance Metrics

The performance of each control strategy is evaluated using several key metrics:

- **Number of Balloons Collected:** The primary metric, which indicates how effectively the quadcopter can reach and collect target points within the simulation time limit.[3]
- **Stability:** Assesses how consistently the control strategy maintains the quadcopter's trajectory without causing crashes or large deviations.
- **Efficiency:** Measures how quickly the quadcopter can reach and collect target points.

4.2 PID Control vs. SAC

The comparison between PID control and SAC highlights the strengths and weaknesses of traditional control strategies versus modern Machine Learning approaches.

1. **PID Control:**
    - **Strengths:** Simplicity, ease of implementation, and stable performance. PID controllers are effective for maintaining steady control in relatively simple or well-understood environments.
    - **Weaknesses:** Limited adaptability and performance in complex or dynamic environments. Manual tuning is required, and the controller may struggle with rapidly changing conditions.

2. **SAC (Soft Actor-Critic):**

   - **Strengths:** High adaptability and performance in complex environments. The SAC agent can learn optimal control strategies that outperform traditional methods in terms of efficiency and stability.
   - **Weaknesses:** Requires extensive training and computational resources. The complexity of the algorithm can make it challenging to implement and interpret.

4.3 Advantages and Disadvantages

The analysis reveals that while PID controllers offer reliable and predictable control, they are limited in their ability to handle complex, dynamic environments. They require manual tuning and are less flexible when it comes to adapting to new or unforeseen circumstances. This makes them well-suited for scenarios where the environment is relatively static and well-understood, but less effective in more challenging settings.

On the other hand, SAC, as a reinforcement learning algorithm, excels in environments that are complex and dynamic. It has the ability to learn and optimize control strategies through interaction with the environment, making it highly adaptive. SAC can outperform traditional methods in terms of efficiency and stability, particularly in situations where the environment is not fully known or changes over time. However, the trade-offs include a higher requirement for computational resources and the need for extensive training data. Additionally, SAC's complexity can make it harder to interpret and debug compared to traditional control methods.

Chapter 5

Results

5.1 Tabular Data

Table 5.1: Simulation Results

Test No.	SAC	PID	Manual
1.	57	51	24
2.	55	46	6
3.	55	53	6
4.	56	48	10
5.	63	55	15
6.	59	51	15
7.	55	53	6
8.	64	51	26
9.	61	48	18
10.	55	46	6

5.2 Inference

This data captures the score of each type of control strategy implemented on the rigid body quadcopters and thus compares it on the basis of target locations reached (points) within the given time frame (the simulation time) of 100 seconds.

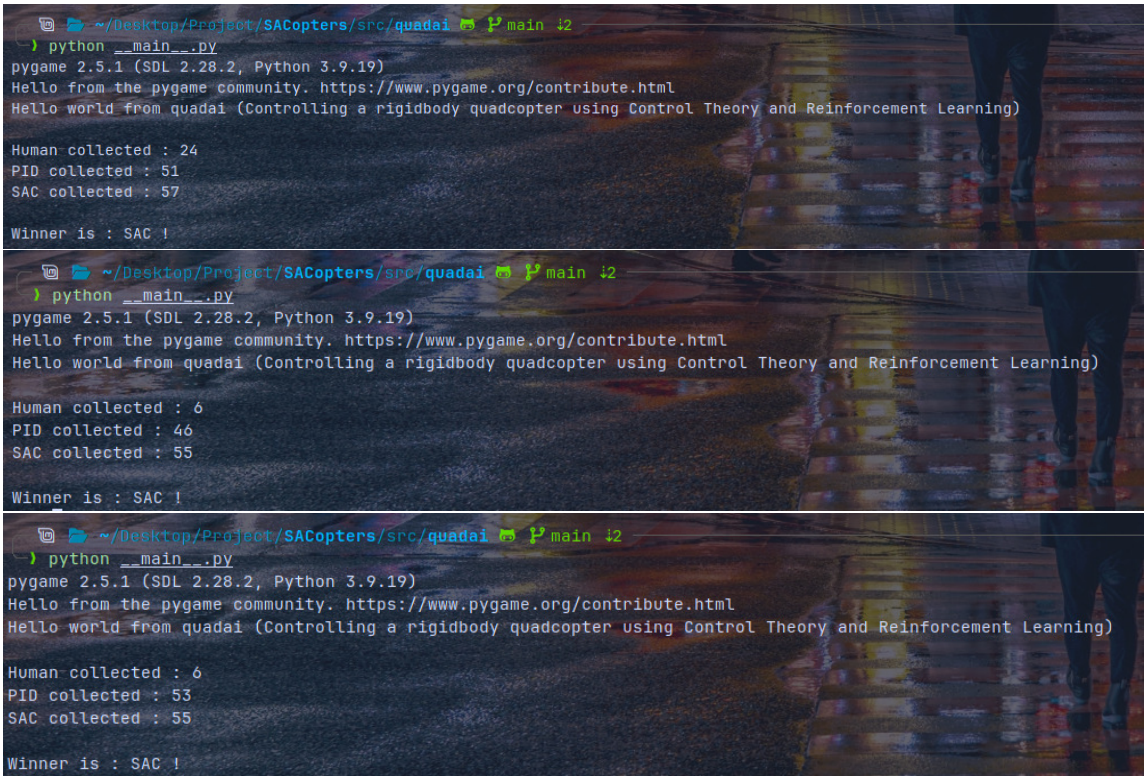

Figure 5.1: Terminal Snippets Of Simulation Test Case

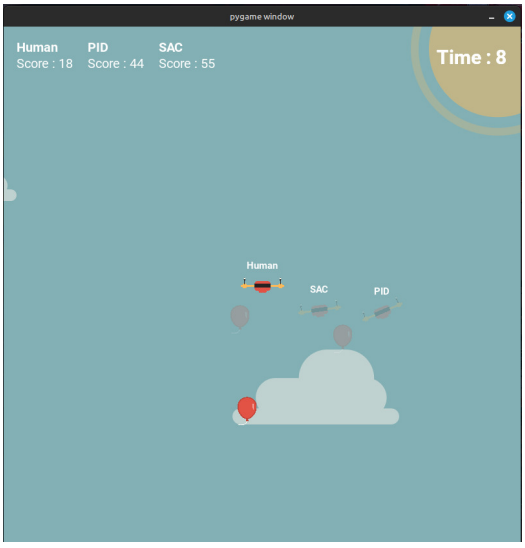

Figure 5.2: Game Window Snippet Of Simulation

Chapter 6

Future Work

6.1 Implementation of DQN

One of the future directions for this project is the implementation of a Deep Q-Network (DQN) model. DQN is another reinforcement learning algorithm that could provide a useful comparison with SAC, particularly in how it handles discrete action spaces versus SAC's continuous actions. Implementing DQN would allow for a more comprehensive analysis of reinforcement learning strategies in UAV control.[5]

6.2 Exploration of Hybrid Control Strategies

Another interesting direction for future work is the exploration of hybrid control strategies that combine the strengths of both traditional and machine learning methods. For instance, a PID controller could be used for basic stabilization, while a reinforcement learning agent could handle more complex decision-making processes. This approach could potentially offer the best of both worlds, combining the reliability of traditional methods with the adaptability of machine learning.

6.3 Impact of Environmental Disturbances

Finally, further research could explore the impact of environmental disturbances, such as wind or sensor noise, on the performance of the control strategies. This would provide valuable insights into the robustness of each approach and help identify ways to improve their performance in real-world scenarios.

Chapter 7

Conclusion

The SACopters project successfully demonstrated the potential of Machine Learning algorithms, particularly SAC, in controlling UAV trajectories. While traditional PID controllers offer a stable and straightforward approach, they are outperformed by the SAC algorithm in more complex and dynamic environments. It's ability to learn from the environment and optimize control strategies makes it a powerful tool for UAV control, although it comes with the challenges of higher computational requirements along with the increased complexity.

References

[1] R. Arnold, E. Mezzacappa, J. Jablonski, and B. Abruzzo. Multi-role uav swarm behaviors for wide area search using emergent intelligence. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, London, UK, July 2020. IEEE. DOI: 10.1109/WorldS450073.2020.9210392.

[2] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. *arXiv preprint arXiv:1604.06778*, 2016. 14 pages, ICML 2016. DOI: 10.48550/arXiv.1604.06778.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. DOI: 10.48550/arXiv.1312.5602.

[4] F. Salis. Lower dimensional S^1-invariant Kähler-Einstein metrics via integrable structures. *arXiv preprint arXiv:2206.07755*, 2022. DOI: 10.48550/arXiv.2206.07755.

[5] G. Sanderson. Neural networks by 3Blue1Brown, 2017. *YouTube Playlist*.