

Reinforcement Learning Approach for Co-Management of Computational and Mechanical Power Consumption in Mobile Robots

UNIVERSITY OF TURKU
Department of Computing
Master of Science (Tech) Thesis
Robotics and Autonomous Systems
May 2025
Afrooz Naseri

Supervisors:
Adjunct Prof. Hashem Haghbayan
Prof. Juha Plosila

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

UNIVERSITY OF TURKU
Department of Computing

Afrooz Naseri: Reinforcement Learning Approach for Co-Management of Computational
and Mechanical Power Consumption in Mobile Robots

Master's Thesis, 54 p.
Robotics and Autonomous Systems
May 2025

Optimizing energy consumption remains a critical challenge in autonomous mobile robotics and is essential for extending robot battery life. Two significant sources of energy consumption are the mechanical and computational components, both of which contribute to battery usage. Recent studies indicate that dynamically coordinating mechanical and computational power usage, such as tuning processor frequency based on the robot's mechanical speed, can greatly enhance energy efficiency across diverse mobile robot platforms. This improvement is primarily due to the relationship between decision-making processes based on mechanical speed and computational workload. In this thesis, we begin by presenting a simulation model that estimates the instantaneous power consumption of a mobile robot by taking into account both its mechanical and computational components while also analyzing how its computational aspects impact path planning. The simulation model is adaptable to be tuned based on the level of accuracy needed for estimating the power consumption for the robot and the simulation time penalty. This makes a multi-fidelity power estimation tool for the robot with the capability to run-time changing the fidelity according to environmental conditions and internal computational capabilities. Subsequently, we propose an agile reinforcement learning algorithm for dynamic co-management. Using a rover capable of manipulating the speed of its brushless motor, adjusting the frequency of the Jetson TX2 processing unit, and utilizing an event-based camera as the perceptual sensor, we demonstrate that our method effectively addresses the scalability and accuracy issues found in previously proposed techniques. As a result, we achieve significantly greater efficiency in overall energy reduction and mission duration. Our experimental results show energy efficiency improvements ranging from a minimum of 16.98% to a maximum of 60.86% compared to the most efficient approaches documented in the literature.

Keywords: Mobile Robot, Computational Power, Mechanical Power, Multi-Fidelity Power Estimation, Path Planning, Reinforcement Learning

Contents

1	Introduction	1
1.1	Objectives and Research Questions	4
1.2	Thesis Structure	6
2	Literature Review	7
2.1	Mechanical Energy	7
2.2	Computational Energy	10
2.3	Co-management of Mechanical and Computational Energy	14
3	Preliminary	17
3.1	Simulation Framework	17
3.2	Estimation of Mechanical Power	19
3.3	Estimation of Computing Power	20
3.4	Estimation of Energy	21
3.5	Multi-Fidelity Energy Estimation	21
3.6	Path Planning	22
3.6.1	Markov Decision Process	23
4	Reinforcement Learning Approach	25
4.1	Working Scenario	25
4.2	Proposed Model	26
5	Results	31
5.1	Experimental setup	31
5.2	Simulation Results	32

5.3	Path Planning Results	37
5.4	Reinforcement Learning Results	39
6	Discussion	49
6.1	Contribution of this study	49
6.2	Limitation	52
6.3	Future Works	53
	References	55

List of Figures

2.2.1 The resource management system with real-time reliability awareness . .	11
3.1.1 Schematic of simulation system	18
3.6.1 Markov decision process	23
4.1.1 Overview of the robotic system	25
4.2.1 Overview of the RL framework.	27
5.1.1 Demonstrator setup	31
5.1.2 Number of events per batches vs. speed of robot in different complexities	32
5.2.1 Energy consumption for different speed granularity configurations in low complexity	33
5.2.2 Energy consumption for different speed granularity configurations in high complexity	33
5.2.3 Error in energy estimation and computational time vs. speed granularity interval	34
5.2.4 Mechanical energy for different speed configurations	35
5.2.5 Energy consumption vs. application throughput for high complexity . . .	36
5.2.6 PLPT compared to frequency level for different speed settings	36
5.2.7 Average batch per second vs. frequency level for different speeds	37
5.3.1 Paths in complex environment	38
5.3.2 Comparison of two paths in a more complex environment	39
5.4.1 Optimal configurations for different complexities	40
5.4.2 Energy vs. speed in low complex environment for Proximal Policy Optimization (PPO) model	43

5.4.3 Energy vs. speed in high complex environment for PPO model	44
5.4.4 Energy vs. time for different methods	45
5.4.5 Energy consumption in different environment complexities	45
5.4.6 Mission time in different environment complexities	46
5.4.7 Execution time vs. different speed interval	47

List of Tables

5.2.1 Key Variables in Mechanical Power Modeling	32
5.3.1 Energy Consumption for Different Paths	38
5.4.1 PPO’s Model Parameters	43

List of Acronyms

- **RL** - Reinforcement Learning
- **PPO** - Proximal Policy Optimization
- **QoS** - Quality of Service
- **PLPT** - Per Loop Processing Time
- **ANN** - Artificial Neural Network
- **HL** - Hill Climbing
- **SO** - Separate Optimization
- **PO** - Proactive Energy Optimization
- **MPC** - Model Predictive Control
- **MDP** - Markov Decision Process
- **DVFS** - Dynamic Voltage and Frequency Scaling

1 Introduction

Energy efficiency plays a crucial and increasingly important role in the control and operation of battery-powered mobile robots. As these robots are typically designed to perform various tasks autonomously, their energy consumption directly impacts their operational time, performance, and longevity. These robots are composed of multiple components, including sensors, computing units, and mechanical systems. Each of these components employs its own energy-efficient policy, which aims to reduce power consumption and optimize performance in specific tasks. However, the interdependence among these units, where the behavior of one component may affect the others, complicates the energy management process. Consequently, this necessitates a holistic co-management approach to ensure that the overall system's energy efficiency is optimized, rather than optimizing individual components separately. Recent research has increasingly addressed this challenge, proposing innovative strategies for integrated energy management across various robotic subsystems [1].

One of the key aspects of this interdependence is the mutual reliance of mechanical and computational energy consumption. For instance, previous studies have demonstrated how a robot's mechanical and computational energy requirements are intertwined, particularly when the robot's perception accuracy needs to be maintained at a certain level [2]. In these cases, to meet the required accuracy in tasks such as object detection or environmental mapping, adjustments to the robot's mechanical behavior, such as its speed or movement pattern, must be made in conjunction with the computational power used by its processors. More specifically, the robot's speed is often adjusted dynamically based on the voltage and frequency of its processors to ensure that the perception system operates within an acceptable margin of accuracy. These interdependencies between mechanical and computational energy consumption present a significant challenge in achieving opti-

mal overall energy efficiency.

A primary challenge in effectively jointly handling the mechanical and computing aspects of a mobile robot is the design of a suitable controller that can co-optimize both domains in real time. The challenge arises because, although energy consumption serves as the common cost function for optimization, the perception and actions involved in the mechanical and computational components may not align directly in terms of dimensionality or control structure. The mechanical system may be governed by physical dynamics that involve continuous and time-dependent control variables, while the computational system involves discrete decisions made by processors based on complex algorithms. These discrepancies make it difficult for traditional control methods, which often rely on predefined models and fixed strategies, to manage both components effectively. To address this, the information processed by the co-manager can be categorized into two types: objective mechanical information and subjective computational information. Objective information refers to data derived from the robot's mechanical components, such as sensors, actuators, and its interaction with the physical environment. This information helps describe the robot's configuration and its behavior in space and time. On the other hand, subjective information is generated by the robot's computing units, which process and represent the robot's internal model of the world. This internal representation is crucial for decision-making, but it may not always correspond directly to the real-world measurements captured by the sensors. Consequently, despite the significant interdependence between these components, conventional model-based controllers are often ill-suited for designing a fast and accurate co-management system that can adapt in real time to both mechanical and computational demands.

Current state-of-the-art methods for inter-unit co-management primarily rely on naive, exhaustive search approaches for quantizing speed ranges and organizing them in a multi-dimensional lookup table. While these methods are able to provide solutions for specific configurations, they tend to suffer from scalability and response time issues. This becomes particularly problematic when trying to predict future behavior of the robot or when attempting to integrate new features into the system, as the lookup table-based approaches often require recalibration and fine-tuning. These limitations hinder the ability to scale the system to larger, more complex robots or more diverse operational environ-

ments. Recent studies have highlighted these limitations and stressed the need for more integrated and adaptive control mechanisms to handle such complexity [3]. Therefore, there is a growing need for a more flexible, scalable approach that can better handle the dynamic nature of energy consumption and control in mobile robots.

In this research, we introduce a novel and structured simulation framework designed to estimate a mobile robot's power consumption during its operation. This framework explicitly considers the interdependence between the mechanical and computational units, providing a more accurate and integrated model for energy consumption. By incorporating both systems into a single framework, we can more precisely track how changes in one domain, such as an adjustment in the robot's speed, impact energy consumption across other components, such as the processors or sensors. Furthermore, this framework allows for dynamic adjustments in power estimation accuracy, which can be fine-tuned based on computational cost constraints. As a result, the framework is adaptable for both real-time applications, such as Reinforcement Learning (RL)-based decision-making and Model Predictive Control (MPC), as well as offline optimization processes. Unlike traditional methods that rely on fixed-configuration search techniques, our framework can adapt its fidelity during operation. The power estimation accuracy can be adjusted in real-time based on environmental conditions and the available internal computational resources, thereby enabling proactive energy optimization that is responsive to both external and internal factors.

In addition to the simulation framework, we propose using data-driven methods, specifically Reinforcement Learning (RL), to directly develop the best control strategies from real-world data. In contrast to traditional methods that rely on manually defined models and optimization procedures, RL enables the system to learn from experience and adapt its control strategies based on real-time feedback from the environment. In this context, we utilize a feed-forward neural network with carefully selected features as a flexible and adaptive solution for co-managing both the computational and mechanical components of the robot. To optimize the control policy, we adopt Proximal Policy Optimization (PPO), an RL algorithm known for its stability and reliability in policy updates. PPO uses a neural network architecture to optimize policy-based learning models, ensuring that updates to the policy are made in a stable manner, preventing drastic changes that could

destabilize the system [4]. This is achieved through the use of a proximal update mechanism, which constrains the magnitude of policy changes during each training step. By incorporating PPO, our approach is able to continuously refine its control policy in a stable manner without significantly deviating from the existing policy, ensuring consistent improvements over time.

Furthermore, the network training process is driven by a reward function based on the overall system energy, allowing the controller to learn energy-efficient policies that minimize consumption while still achieving the desired performance. This approach leads to a fast, scalable, and multi-objective controller capable of integrating new features from both the robot's internal and external environments. Unlike traditional methods that search for optimal configurations in a fixed configuration space, our RL-based approach scales well with configuration granularity, meaning that it can efficiently handle both fine-grained configurations and the addition of new components to the system. We will demonstrate through experimental results that this approach significantly enhances the robot's response time and optimizes long-term energy consumption, providing a promising solution to the challenges of energy management in mobile robots.

1.1 Objectives and Research Questions

The objective of this thesis is to propose a comprehensive framework for enhancing energy efficiency and computational performance in mobile robots through advanced energy estimation and reinforcement learning techniques. More specifically, this thesis focuses on the development of a simulation framework that incorporates real-time power estimation, a dynamic tuning mechanism for balancing energy estimation accuracy and computational efficiency, and reinforcement learning for computational and mechanical co-management. The contributions include:

- Proposing a simulation framework for real-time power estimation of a mobile robot, incorporating both mechanical and computational energy consumption.
- Implementing a dynamic tuning mechanism to balance energy estimation accuracy and computational efficiency based on environmental conditions and computational con-

straints.

- Analyzing how varying environmental complexities affect energy consumption and, in turn, influence navigation efficiency within a Markov Decision Process-based path planning framework.
- Extracting suitable features and defining a reward function for the reinforcement learning of computational and mechanical co-management processes in robots.
- Proposing a neural network model trained at design time to predict near-optimal mechanical speed and the computational frequency of the processor.
- Conducting system identification and experimental validation on a wheeled ground robot equipped with an event camera for perception, a brushless DC motor for the mechanical system, and a microprocessor for computational tasks.

The research questions are designed to help create a checklist of key requirements for building an effective energy management system in mobile robots. In this thesis, the questions are labeled as RQ_i and are organized as follows:

- **RQ1:** How can real-time power estimation be effectively integrated into mobile robotic systems?
- **RQ2:** What dynamic tuning mechanisms can enhance the balance between energy estimation accuracy and computational efficiency?
- **RQ3:** How can reinforcement learning be utilized for the computational and mechanical co-management of mobile robots?
- **RQ4:** What features and reward functions are essential for effective reinforcement learning in robotic systems?
- **RQ5:** How can the proposed neural network model be validated in predicting mechanical speed and computational frequency in practice?

1.2 Thesis Structure

The remainder of this thesis is organized as follows: In Chapter 2, we provide a comprehensive overview of the existing research related to power consumption in robotics. This chapter reviews key studies and highlights significant advancements in this field, establishing a foundational understanding of how power efficiency is approached in robotic systems. Chapter 3 presents a detailed description of the preliminary work and the methodology employed to simulate the power estimation system. Following this, Chapter 4 introduces our proposed reinforcement learning approach, which focuses on the dynamic co-management of power usage in mobile robots. We elaborate on the architecture, training process, and anticipated benefits of this innovative strategy. In Chapter 5, we dedicate our attention to the results and discussion, analyzing the performance of our proposed methods and comparing them with existing techniques. This chapter also addresses the implications of our findings and identifies areas for future research. Finally, Chapter 6 concludes the thesis with a summary of our contributions, reflecting on the overall impact of this research and offering concluding remarks that encapsulate the key insights gained throughout the study.

2 Literature Review

The related works in this research can be examined from three points of view: mechanical, computational, and co-management of mechanical and computational energy usage.

2.1 Mechanical Energy

There has been extensive research on estimating mechanical energy in various types of robots. Most proposed techniques rely on modeling the robots' mechanical components using different formal and data driven techniques. For instance [5] presents a simplified modeling approach to estimate the energy usage of robotic systems. Researchers propose mechanical and motor electrical power models, which are then extended to calculate the total energy along specified trajectories. By adjusting motion interpolator parameters such as acceleration and velocity, the model aids in optimizing energy usage. Their approach utilizes differentiable inertial and kinematic models from standard open-source tools, integrating with standard ROS planning methods. An inverse dynamics-based energy model is optionally extended with a single-parameter electrical model, simplifying the model identification process. They compare the inertial and electrical models on a collaborative robot, showing that simplified models provide competitive accuracy and are easier to deploy in practice.

Similarly, [6] addresses the significant energy consumption of industrial robots in manufacturing by focusing on mechanical energy consumption and developing a detailed consumption model. Recognizing that direct torque measurements are often inaccessible, they propose an alternative method to identify inertial and friction parameters using software-simulated power data. This approach involves analyzing the energy flow within the robot, focusing on the power consumed by the control system and the motors.

The researchers validate their method through extensive simulations on an ABB IRB 1200 robot, demonstrating that their model effectively estimates energy consumption without relying on direct torque data. Additionally, they analyze the relationship between the robot speed and energy consumption, enabling more energy-efficient operation through parameter optimization insights. The authors in [7] developed a mathematical model of a brushed DC motor to predict mechanical power based on angular velocity. They focused on the relationship between the motor's electrical input and its mechanical output, considering factors such as armature resistance and back electromotive force. By analyzing the voltage induced in the rotor, which is proportional to the angular velocity, they derived equations to estimate the motor's performance under various operating conditions. This approach allows for accurate predictions of mechanical power output without the need for direct torque measurements, facilitating more efficient design and control of robotic systems. In [8] researchers developed a comprehensive physical model to analyze the energy consumption of industrial articulated robots. Their approach utilizes graphically-oriented computer-aided design (CAD) tools, specifically SolidWorks, in conjunction with MATLAB/Simulink and its SimMechanics and SimPowerSystems libraries. This integration facilitates the creation of a dynamic simulation model that accurately represents both the mechanical structure and the drive systems of the robot. The researchers conducted mathematical analyses to derive equations of motion for the robot's mechanical components and integrated these with dynamic models of the robot's drives. This comprehensive modeling framework enables precise calculations of energy consumption during various robot motions, providing valuable insights for optimizing operational efficiency. In a more recent study [9], authors introduced a hybrid energy consumption model for industrial robot arms that combines parametric dynamic modeling with data-driven techniques. Recognizing the challenges in accurately estimating dynamic parameters—such as inertia, center of mass, and friction—they proposed integrating an approximate parametric model with an artificial neural network (ANN). This ANN is trained to learn the discrepancies between simulated outputs and actual observed data. To efficiently gather training data, the researchers employed Halton-sequence sampling, focusing on joint angles as inputs and corresponding energy consumption as outputs. The proposed approach utilizes a single hidden-layer ANN with 475 neurons. The optimal number of

neurons was determined through hyperparameter optimization using the Tree-structured Parzen Estimator (TPE) method [10], which provides an efficient and automated means of searching for the most suitable hyperparameters for the model. Experimental validations showed that this hybrid approach significantly reduces mean squared error and improves R-squared values compared to individual parametric or data-driven models, all while relying solely on approximate dynamic parameters.

Further contributions have leveraged machine learning and hybrid modeling approaches to improve the estimation of mechanical energy consumption. In [11], Zhang et al. introduced a hybrid modeling framework that integrates deep reinforcement learning (DRL) with traditional dynamic modeling to predict energy consumption in industrial robots. Their approach addresses the challenge of inaccessible dynamic and electrical parameters by employing a DRL agent to identify these parameters from trajectory data. Subsequently, a deep neural network, incorporating long short-term memory (LSTM) and one-dimensional convolutional neural network (1D-CNN) layers, predicts energy consumption based on joint torques and velocities. Experimental validation on a KUKA KR60-3 robot demonstrated that this method achieves a mean absolute percentage error of less than 2% under fixed loads and under 3% for varying loads, highlighting its robustness and accuracy in diverse operational conditions.

In a related effort, [12] developed a digital twin model of a six-axis industrial robotic arm to estimate energy consumption. Utilizing Unity for the digital environment and deriving dynamic equations through the Euler-Lagrange method, their model allows for intuitive path planning by directly manipulating the end effector in the virtual space. The energy consumption for each planned path is computed based on the derived dynamics. Validation experiments showed that the physical robot closely followed the digital twin's movements, with the estimated energy consumption aligning well with actual measurements, demonstrating the model's effectiveness in energy estimation and path planning. Another notable contribution is [13], which introduced a KAN-LSTM (Kernel Attention Network–Long Short-Term Memory) model to predict and optimize the mechanical energy consumption of industrial robots operating under unknown load conditions. The focus of this work is specifically on the energy used by the robot's actuators and motion-related components. In their analysis, the authors categorize power consumption into sev-

eral components: P_E , P_B , P_R , P_{M_i} , and P_{MD_i} , representing the power consumed by all electronics, braking systems, chopping resistors, each motor, and motor drivers, respectively. Tested on the AUBO-E5 collaborative robot, the model achieved a mean relative error of 1.18% in predicting energy consumption, outperforming traditional models like BP and standard LSTM networks. Additionally, the model facilitated energy optimization, achieving potential savings between 53.1% and 64.7%. This approach underscores the efficacy of advanced neural network architectures in improving mechanical energy efficiency in dynamic industrial environments.

Although the proposed techniques provide a good approximation of the robot's mechanical components, they only account for a fraction of the various power-consuming elements within the system. Furthermore, the functionalities of separate components—such as sensors, computing units, and mechanical parts—interact with one another, resulting in a dependency on power consumption among different parts. Another crucial aspect that these models often overlook is the impact of environmental characteristics on the robot's power consumption. Different environmental conditions can significantly influence the performance and power consumption of the robot's various components.

2.2 Computational Energy

Parallel to advancements in mechanical energy modeling, computational energy management has also seen progress. For example, in the following works, researchers have explored various methods to model and optimize computational energy consumption. In [14], authors model computational energy by integrating run-time thermal and power models to optimize energy allocation in many-core architectures. They propose a two-layered framework that combines dynamic power management with runtime resource management, which balances energy efficiency and reliability. This approach addresses the challenges of dark silicon by ensuring that not all cores are active simultaneously, thus preventing overheating and mitigating aging effects. The framework includes application and task mapping strategies that allocate idle cores while monitoring the aging status of each core to inform decisions on resource allocation. Experimental results demonstrate that their method effectively optimizes performance while extending the chip's lifespan

under various power management strategies. Similarly, [15] explores computational energy modeling in Chip Multiprocessors (CMPs) by integrating energy-aware workload allocation with run-time management techniques. They introduce a comprehensive framework that not only estimates energy consumption but also assesses performance metrics and reliability of the chip in real-time. This framework employs advanced algorithms to dynamically allocate resources based on workload characteristics and thermal conditions, allowing for effective energy management. By leveraging predictive models, the system can optimize task scheduling and core activation, ensuring that energy efficiency is maximized while maintaining the overall performance and reliability of the CMP. The overall structure of the proposed lifetime-reliability-conscious run-time resource manager is shown in Figure 2.2.1.

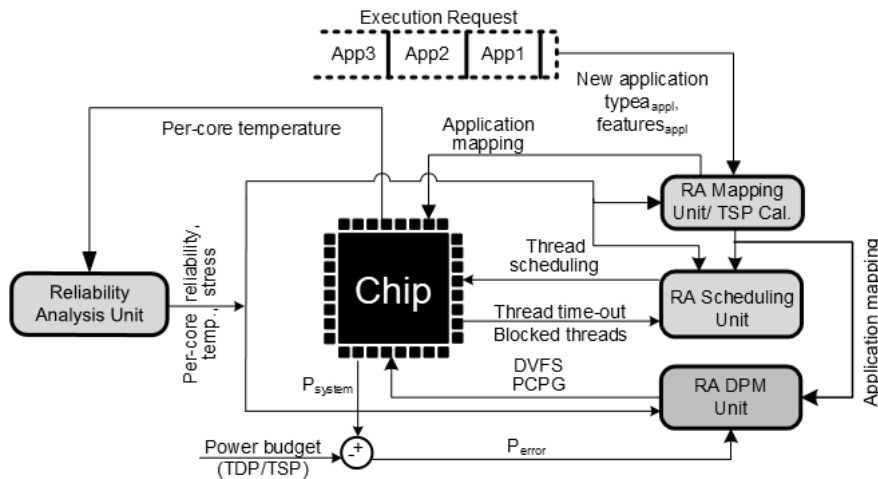


Figure redrawn from [15]

Figure 2.2.1: The resource management system with real-time reliability awareness

Their experimental results highlight significant reductions in energy usage without compromising computational effectiveness, addressing the critical need for efficient resource management in modern multi-core systems. Moreover, in [16], McPAT-Calib, a framework designed to enhance the accuracy of power modeling in contemporary CPU architectures, has been introduced. By calibrating existing models with empirical data, McPAT-Calib provides more precise estimations of power consumption across various microarchitectural components. This calibration process leverages machine learning techniques to refine power estimates, allowing the framework to quickly adapt to new CPU

configurations and workloads. Additionally, McPAT-Calib supports the evaluation of energy efficiency in heterogeneous multiprocessor systems by providing insights into how different architectural components contribute to overall power consumption. The framework is instrumental in identifying potential optimization areas, facilitating better design decisions that enhance both performance and energy efficiency in modern CPU designs. The authors in [17] introduce a power optimization strategy for asymmetric multiprocessors that dynamically adjusts power usage across processors based on workload characteristics, effectively balancing energy consumption and performance. This strategy leverages the heterogeneous nature of asymmetric multiprocessor systems, where processors with varying performance and power consumption profiles are utilized to execute tasks. By analyzing the specific demands of running applications, the system allocates workloads to processors that offer the best trade-off between performance and energy efficiency. This dynamic adjustment not only enhances energy efficiency but also ensures that performance requirements are met, making it particularly beneficial for systems with power constraints. The approach is implemented within the Linux operating system, utilizing the ARM big.LITTLE architecture, which integrates high-performance cores with energy-efficient cores. Experimental results demonstrate that this method achieves significant energy savings while maintaining the desired performance levels, highlighting its effectiveness in real-world applications. Similarly [18] proposes an energy-efficient runtime mapping and thread partitioning technique for concurrent OpenCL applications on CPU-GPU Multi-Processor Systems-on-Chips (MPSoCs). This approach aims to reduce energy consumption by effectively distributing workloads across processing units. The technique dynamically allocates tasks to CPU and GPU cores based on performance requirements and energy constraints, ensuring that each application meets its performance goals while minimizing energy usage. The authors validate their method experimentally on the Odroid-XU3 hardware platform, utilizing various applications from the Polybench benchmark suite. The results demonstrate an average energy saving of 32% compared to existing approaches, highlighting the effectiveness of their strategy in real-world scenarios. Researchers in [19] present an integrated CPU-GPU power management approach specifically targeting 3D mobile applications. This strategy jointly manages power across CPU and GPU cores to achieve energy savings while maintaining a high-quality gam-

ing experience. The authors propose a unified Dynamic Voltage and Frequency Scaling (DVFS) mechanism that adjusts both CPU and GPU frequencies in tandem, optimizing power consumption without compromising performance. By analyzing the workload characteristics of 3D games, the system dynamically allocates resources, ensuring that power usage aligns with the computational demands of the application. Experimental results demonstrate that this integrated approach effectively reduces power consumption compared to traditional methods, highlighting its potential for enhancing energy efficiency in mobile gaming devices.

To further enrich the landscape of computational energy modeling, several recent studies have introduced novel methodologies targeting both software-level optimizations and intelligent scheduling frameworks. For instance,[20] introduced EnergyAnalyzer, a static analysis tool designed specifically for estimating the energy consumption of embedded software applications. Unlike dynamic profiling tools, EnergyAnalyzer performs static analysis, meaning it does not require code execution to predict energy usage. The tool leverages techniques similar to Worst-Case Execution Time (WCET) analysis to estimate upper bounds on energy consumption. It maps code-level features to energy models tailored for embedded processors such as the ARM Cortex-M0. This static approach is particularly valuable in safety-critical or resource-constrained systems, where runtime instrumentation may not be feasible. The tool was validated using benchmark suites and demonstrated high correlation with measured energy values, making it an effective early-stage development utility for identifying and optimizing energy-critical code regions.

In a different but related area, [21] focused on embedded control systems and proposed the Enhanced Energy-Aware Feedback Scheduling (EEAFS) framework. Traditional embedded control systems often use static sampling rates, which may lead to unnecessary energy consumption when full performance is not needed. EEAFS addresses this by dynamically adjusting the sampling periods of control tasks in response to current system performance, such as control error or task execution time. This adaptation is achieved using a feedback control mechanism integrated with Dynamic Voltage Scaling (DVS), allowing the processor to operate at reduced voltages and frequencies without compromising control quality. Simulation results showed substantial energy savings while maintaining acceptable control performance, demonstrating the practical feasibility

of adaptive energy-aware scheduling in real-time control systems. Lastly, in the context of heterogeneous computing environments, [22] proposed an energy-aware task scheduling framework for DVFS-enabled heterogeneous clusters. Their approach aims to minimize total energy consumption while meeting task deadlines by leveraging DVFS. The authors developed an analytical model that captures the nonlinear relationship between task execution time and processor speed, particularly for GPU-accelerated applications. Based on this model, they designed heuristic scheduling algorithms that assign tasks to appropriate processors and determine optimal voltage/frequency settings. Experimental evaluations, driven by real-world power measurement traces, demonstrated that their scheduling algorithm achieved energy savings close to the theoretical upper bound, with recorded savings of 33–35% in scenarios where up to 36% was analytically possible. This work highlights the effectiveness of DVFS-based scheduling in reducing energy consumption in modern heterogeneous clusters.

To achieve a comprehensive energy management strategy in robotic systems, it is essential to consider both computational and mechanical components and their interdependencies. As highlighted in the previous sections, advancements in modeling and optimizing mechanical energy have improved our understanding of how robotic systems consume power. Similarly, the progress made in computational energy management has revealed significant insights into optimizing processing units. However, energy consumption in robots cannot be effectively addressed by focusing on either domain in isolation. The interplay between mechanical movements and computational tasks influences overall system performance and energy efficiency. Therefore, a co-management approach that integrates both mechanical and computational energy considerations is crucial for enhancing the operational efficiency of robotic systems.

2.3 Co-management of Mechanical and Computational Energy

Muralidharan and Mostofi [23] presented a comprehensive review of communication-aware robotics, highlighting approaches that exploit robot motion to enhance communi-

cation performance. Their work highlights strategies that exploit the physical movement of robots to enhance communication performance, particularly in challenging environments. By considering realistic communication constraints, the authors argue for a joint optimization framework that balances both navigation and communication needs. This perspective is vital for developing autonomous systems that not only navigate effectively but also maintain robust communication links, thereby improving overall operational efficiency in complex scenarios.

In the robotics literature, the only works that have specifically considered the cost of computation are [24] and [3]. These studies tackled the challenge of optimizing both mechanical and computational energy, but both faced limitations in execution time and efficiency. In [24], a Hill Climbing (HC)-based approach was employed to co-optimize mechanical and computational energy in mobile robotics. The method dynamically adjusted both CPU voltage/frequency and motor voltage to minimize overall energy consumption while maintaining performance. By iteratively searching for better configurations, the HC algorithm aimed to find an optimal balance between energy efficiency and task execution. However, due to the inherent trial-and-error nature of HC, the approach suffered from prolonged execution times, making real-time implementation challenging. Additionally, the lack of global exploration in HC increased the risk of converging to sub-optimal solutions, limiting its effectiveness in complex robotic tasks that require adaptive energy management. A more advanced method was proposed in [3] to enhance energy optimization in mobile robots. This approach utilized dynamically trained internal models to predict power consumption, thereby reducing unnecessary trials and conserving energy. By proactively adjusting CPU frequency and mechanical speed based on these predictions, the system aimed to achieve optimal energy efficiency. However, the reliance on exhaustive search methods posed significant challenges. The time-consuming nature of exhaustive search not only limited real-time application but also raised concerns about scalability. Indeed, this research have used the knowledge matrix—a lookup table used by the controller to store performance and energy metrics across different configurations—which highlights the scalability challenge, as the size of this structure grows rapidly with the number of speed and frequency levels considered.

As the complexity of robotic systems increases with the addition of more mechani-

cal or computational components, the exhaustive search approach becomes less practical. This limitation underscores the need for more efficient optimization techniques that can handle the increasing complexity of modern robots. Our approach in the present work improves the controller execution time by using a single inference for finding all the actions, significantly reducing the decision-making time compared to the previous methods. Moreover, our proposed approach does not discretize the search space and is able to find continuous configurations, further improving the system energy consumption.

3 Preliminary

3.1 Simulation Framework

In this research, we start by introducing a simulation model that estimates the real-time power consumption of a mobile robot, considering both its mechanical and computational aspects, while also analyzing how these computational factors influence the robot's path planning approach to optimize overall efficiency. Our approach integrates mechanical power consumption into the overall power budget of the robot, considering its dependence on environmental characteristics and the computing unit's ability to process perceptual data. For example, consider a rover that can adjust its speed based on its computational capabilities, which are influenced by the required accuracy of sensory data processing. Enhanced computational capabilities allow for faster data processing, enabling the rover to operate at higher speeds. Conversely, lower computational capabilities result in slower data processing and, consequently, a reduction in speed. This relationship is analogous to human driving behavior; for example, a more agitated or impaired mindset often prefers slower speeds, while driving at higher speeds requires greater focus and consideration. The dependencies between computational and mechanical components are not adequately captured in current state-of-the-art power modeling. This chapter presents an adaptive simulation framework based on [3] that allows for the tuning of computational features and robot speed to achieve the necessary accuracy across various optimization models. Specifically, it emphasizes the importance of balancing high-speed computation with accuracy, as the performance can vary among different approaches.

The overall structure framework of the simulation system proposed in this research is shown in Figure 3.1.1. The model takes three key input parameters, i.e., CPU frequency, robot speed, and fidelity level, which influence the power demands of computing and

mechanical components. The external environmental model accounts for the complexity of the surroundings, impacting both computing and mechanical power needs.

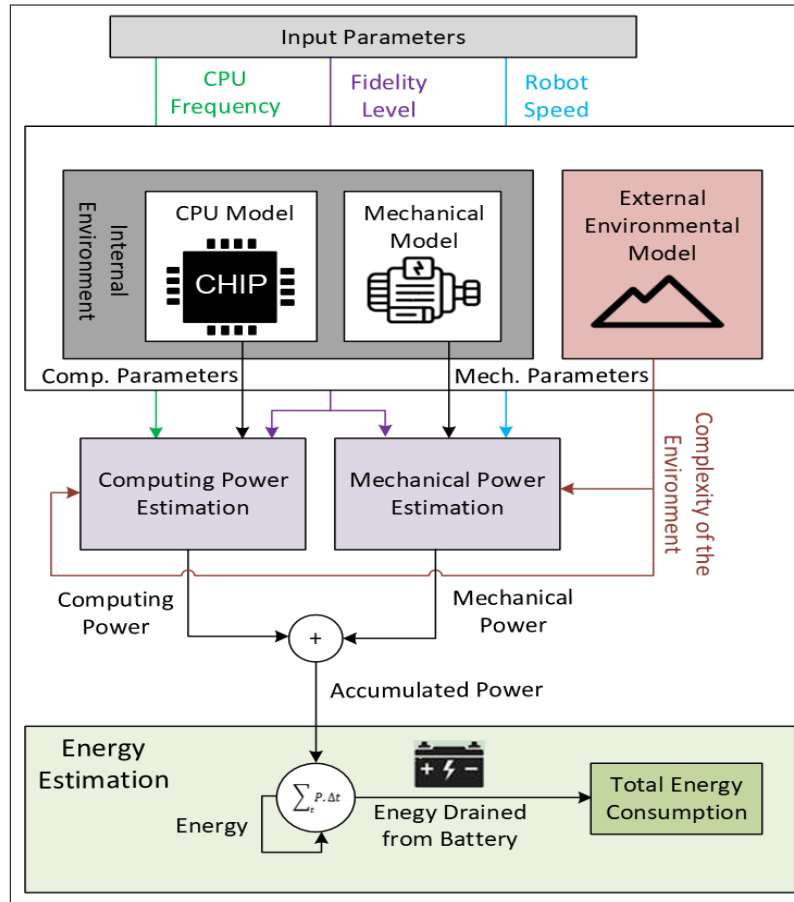


Figure 3.1.1: Schematic of simulation system

The internal environment consists of the mechanical and computational models of the robot, which includes the electric DC motor and steering system, and a CPU model, which consists of the onboard embedded computing chip. The running applications on the computing chip consist of vision algorithms to perform environment perception and enable autonomous operation. These applications receive event batches as input by a fixed rate R and need to process them in real-time to ensure safe operation and avoid information loss.

The Quality of Service (QoS) (or equivalently throughput) of these applications, defined as the number of processed batches per second, must then be equal to R for all applications. The QoS depends on the precision demanded by path planning algorithms and obstacle detection, which must improve as the robot's speed increases. The computational

workload is shown to be dependent on both the robot's motion speed and the visual data complexity shaped by its environment. Moreover, QoS depends on how much computing power is available and how capable each processor is. Consequently, when computational resources are constrained, the robot must adjust its speed to maintain QoS requirements at a manageable level. This interdependence between the mechanical and computational components highlights the necessity of managing both aspects cohesively rather than optimizing each independently without considering their mutual impact. Furthermore, the system evaluates application performance by recording how long each application takes to process one event batch, using this data to derive Per Loop Processing Time (PLPT), an application-level indicator for real-time monitoring. It provides a clear indication of how efficiently each application handles its workload under varying operational conditions. By integrating PLPT into the system, it becomes possible to dynamically assess the computational performance and adjust operational parameters to maintain the desired QoS while balancing resource constraints and environmental complexities.

Building on this interdependence, the energy consumption of the robotic system is estimated by considering both computational and mechanical power demands. The CPU model estimates the computing power required by utilizing CPU frequency, fidelity level, and environmental complexity, while the mechanical model calculates the mechanical power demand based on motor speed, fidelity level, and environmental complexity. These estimations of power consumption are then summed over time to compute the overall energy consumption of the system, providing a precise evaluation of the robot's energy requirements based on its operational parameters and environmental conditions.

We utilized a pair of mathematical models to evaluate the power demands of the mechanical and computing components under a specific configuration. Here are the models for estimation:

3.2 Estimation of Mechanical Power

In a similar manner to previous studies [25] and [26], we have developed a model that estimates mechanical power usage as a function of the robot's speed and the forces acting on it, using Newton's second law. The required power for the robot's motion over a flat

terrain is evaluated using the following formula:

$$P_m = Fv \quad (3.1)$$

The velocity of the robot is represented by v , and the propulsive force provided by the motor is denoted as F . The value of F is determined by the following calculation:

$$F = F_a + F_f + F_d \quad (3.2)$$

$$F_a = m \frac{dv}{dt} \quad (3.3)$$

$$F_f = (C_r + C_v v) \cdot mg \quad (3.4)$$

$$F_d = \frac{1}{2} \rho C_d A_f v^2 \quad (3.5)$$

Here F_a , F_f , and F_d are the acceleration, rolling friction, and aerodynamic drag forces, respectively. The robot's mass is denoted by m , and g is gravitational acceleration. The coefficient C_r is a constant, while C_v is the rolling resistance coefficient. ρ is the air density, C_d is the aerodynamic drag coefficient, and A_f refers to the frontal area of the robot. The parameters of the power model (C_r , C_v , and C_d) can be obtained by fitting equations (3.1–3.5) to measured data.. It should be mentioned that parameter fitting is done using the regularized least squares method.

Therefore, the mechanical power of the system can be estimated using the model presented in equation 3.1.

3.3 Estimation of Computing Power

In order to evaluate the computing power, we use a model that has been commonly referenced in previous studies on multi-core processing boards [27] and [19]. Following this, we determine the relevant parameters for the specific computing board under analysis. Specifically, the following formula is used to calculate the overall power consumption of the computing board:

$$P_c = P_{\text{board}}^{\text{idle}} + \sum_i [P_i^{\text{idle}}(f_i) + P_i^{\text{work}}(f_i) \cdot \%U_i] \quad (3.6)$$

The idle power consumption of the board, which includes the camera power, is represented by $P_{\text{board}}^{\text{idle}}$. The Dynamic power usage results from CPU activity, which depends on the frequency level f_i and the utilization U_i of each core i . In this context, P_i^{idle} and P_i^{work} are the measured power consumption values for core i at frequency f_i , where P_i^{idle} corresponds to the power when the core is inactive ($U_i = 0\%$) and P_i^{work} represents the power when the core is fully utilized ($U_i = 100\%$). Using the operating system's reported usage of each core, power consumption can be determined by applying a linear scaling function.

3.4 Estimation of Energy

To determine the robot's energy consumption over time, we integrate its power usage across the relevant period. Since our method operates in real time, we use energy per distance as the optimization objective. For a given setup with motor speed s and CPU frequency f , the energy per distance $E_d(s, f)$ is calculated by dividing the total power—mechanical plus computational—by the motor speed:

$$E^d(s, f) = \frac{P_m(s) + P_c(f)}{s} \quad (3.7)$$

3.5 Multi-Fidelity Energy Estimation

To enable the run-time fidelity adjustment, we provide an input for the simulation environment, i.e., the fidelity level, to adjust the level of granularity, of the simulation parameters including speed and CPU voltage/frequency level. This process allows energy consumption to be estimated at different levels of energy estimation fatalities. For example, by selecting a finer speed granularity, i.e., resulting in increasing the fidelity of the analysis, the model enhances the precision of energy consumption estimation. The benefit, however, is offset by increased processing requirements. Conversely, coarser granularity reduces computational time but may introduce higher estimation errors. This adaptive

mechanism allows the robot to estimate energy consumption in real time. The accuracy of the energy consumption model is traded off against computational overhead, adjusting dynamically based on available resources.

3.6 Path Planning

In the fields like autonomous ground vehicles, aerial drones, and robotic manipulators, path planning is a fundamental process. While constraints differ depending on the agent, the primary goal remains guiding it from a given starting point to its destination. Many path planning methods have been developed over the years, and one of the most well-known is the A* algorithm [28]. It works by choosing paths based on both the cost so far and an estimated cost to the goal. If this estimate (called a heuristic) is accurate and doesn't overestimate, A* will always find the best path [29]. In the absence of a heuristic, A* behaves similarly to Dijkstra's algorithm. For continuous spaces, the Euclidean distance is frequently adopted as the heuristic to guide the search toward optimal paths. However, A* requires map discretization, often using a grid, which limits flexibility and path optimality. Designers must balance grid resolution, cost, and performance, especially in mixed obstacle areas. To address these limitations, various enhancements to A* have been developed [30, 31]. Some methods dynamically adjust the heuristic function for real-time obstacle avoidance, while others use post-processing techniques, such as Bezier curves, to smooth paths. Because A* can be computationally expensive in high-dimensional environments, Rapidly-exploring Random Trees (RRT) are often used as a more efficient alternative [32]. RRT is more efficient in large, complex spaces but lacks optimality, which is improved in its variant, RRT* [33]. Despite these advancements, traditional A* and RRT struggle in dynamic environments as they lack a mechanism for local path modifications. A potential solution involves modeling path planning as a step-by-step decision-making problem, enabling the robot to refine its route in real time according to environmental utility. The Markov Decision Process (MDP) [34] provides a structured framework for this approach, modeling motion planning as a decision-making process where states, actions, policies, and rewards define the optimal navigation strategy. While A* and RRT use static strategies guided by predefined heuristics, MDP offers

a more dynamic approach, enabling adaptive decisions in dynamic environments. We now describe the mathematical formulation of MDP, explaining its components and how it can be effectively applied to path planning.

3.6.1 Markov Decision Process

The Markov Decision Process (MDP) used in path planning is composed of several elements: the set of states S , the set of actions A , the transition probabilities $P(s'|s, a)$, and the reward function $R(s, a, s')$. The state space S captures all possible situations the agent might encounter in its environment, while the action space A defines the available decisions or moves the agent can make. Within the MDP framework, each action influences not only the reward received immediately but also alters the following state, which impacts future outcomes. The way the agent and its environment interact through these components is illustrated in Figure 3.6.1.

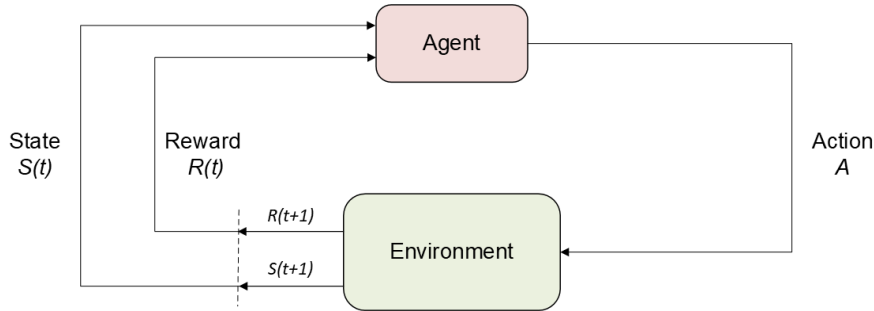


Figure 3.6.1: Markov decision process

The transition model describes the likelihood of reaching a new state s' from the current state s when action a is executed. To guide the agent toward favorable outcomes, the reward function provides a numerical value associated with each transition. Additionally, the state-value function $V^\pi(s)$ evaluates how beneficial it is to be in a given state under policy π , while the action-value function $Q^\pi(s, a)$ assesses the expected return of performing action a in state s while following policy π . These functions are mathematically defined as [35]:

$$V^\pi(s) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s \right] \quad (3.8)$$

$$Q^\pi(s, a) = \mathbb{E} \left[R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^\pi(s') \right] \quad (3.9)$$

The Bellman equations are given by [36, 37]:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma V^\pi(s')] \quad (3.10)$$

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^\pi(s') \quad (3.11)$$

The optimal policy π^* can be determined using the following set of equations [38, 39]:

$$V^*(s) = \max_a Q^*(s, a) \quad (3.12)$$

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^*(s') \quad (3.13)$$

The MDP can be addressed using methods such as Value Iteration and Policy Iteration algorithms, allowing the robot to make optimal decisions in dynamic environments based on its state and the uncertainties present.

4 Reinforcement Learning Approach

4.1 Working Scenario

The reference robotic system studied in this research is a wheeled ground rover consisting of mechanical part, computation part and running applications that enable autonomous functionality. Figure 4.1.1 provides a general illustration of the system architecture. As mentioned in preliminary chapter, the mechanical part includes the electric DC motor and steering system, whereas the computation part consists of the onboard embedded computing chip connected to an event-based camera and a controller.

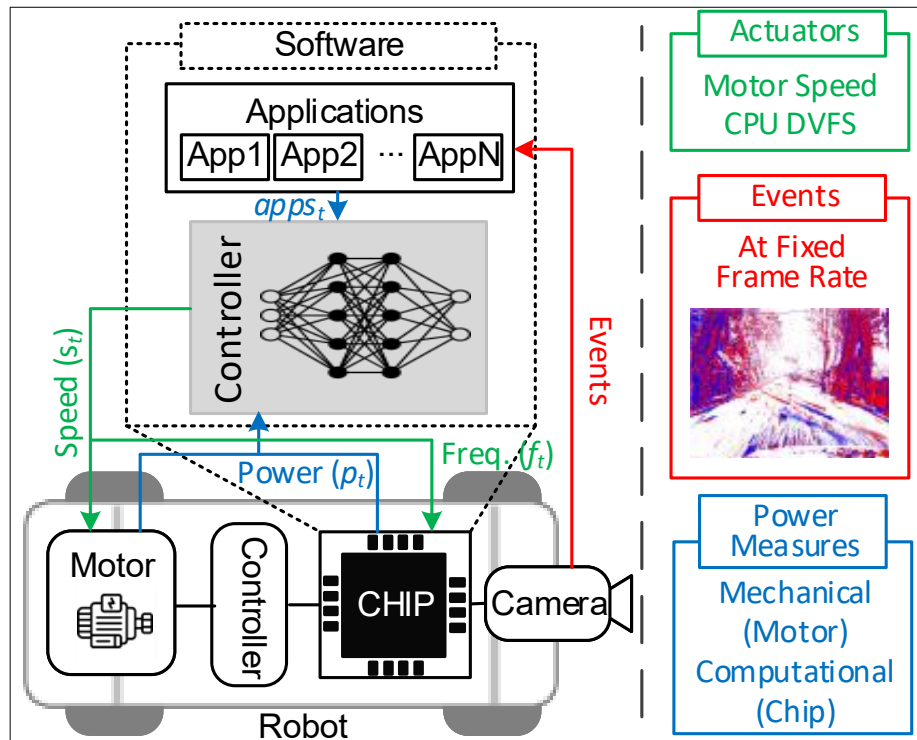


Figure 4.1.1: Overview of the robotic system

In this setup, our proposed controller adjusts both motor speed and CPU frequency at run-time to coordinate the robot’s mechanical and computational components, addressing their interdependence in terms of energy consumption and QoS. We observe the power measurements from the two parts and the actual QoS values for all running applications and based on current system state, we find the energy-optimal configuration that satisfies the QoS requirement.

4.2 Proposed Model

We propose a Reinforcement Learning-based control policy for co-management of the computational and mechanical parts of the robot to optimize its energy consumption at run-time while guaranteeing QoS for the running applications. We adapt the PPO algorithm [4], a model-free policy learning Reinforcement Learning (RL) scheme, to train a neural network-based control policy. The trained controller is then used to infer the energy-optimal configuration of the system that achieves acceptable QoS by observing current system state. Learning this direct mapping from system state to the optimal configuration via RL requires the controller to be able to assess (potentially many) control strategies, i.e., sequence of actuated configurations through time, to update its internal parameters accordingly. It is impractical to perform such trials on the physical robot. Therefore, for the model training purpose, we employ a simulated system consisting of all components in the physical robot, as well as a simulated model for considering various complexities in the external environment. Figure 4.2.1 demonstrates the overall view of the RL training procedure for the controller utilizing the simulated system. In particular, the system state includes current state in robot’s computation and mechanical parts as well as the information of running applications. The actions contain the next system configuration, i.e., motor speed and CPU frequency. The controller model (policy function) is parameterized by a neural network and can be represented as $\pi_{\theta}(a_t|s_t)$, characterizing the distribution of actions given the current state. Specifically, we consider the state of the system as a list of current robot speed, CPU frequency, power measurements for the computing and mechanical parts, and the QoS of the most critical application. The outputs of the policy function consist of mean and standard deviation parameters for two Gaussian

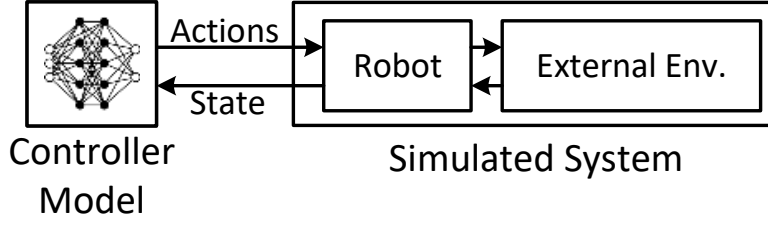


Figure 4.2.1: Overview of the RL framework.

distributions from which the optimal speed and frequency for the next action are sampled. Both outputs are considered to be continuous due to improved performance gain, as shown in [40]. However, since the frequency space is discrete, we employ factorized distribution by linearly spacing the output value into the frequency levels and rounding up the sampled frequency to the next higher level.

The model parameters θ are optimized with PPO algorithm to maximize the clipped objective function, as:

$$\hat{\theta} = \arg \max_{\theta} L^{\text{CLIP}}(\theta) \quad (4.1)$$

where

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(h_t(\theta) \hat{A}_t, \text{clip}(h_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \quad (4.2)$$

in which epsilon is the clipping hyperparameter, $h_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and \hat{A}_t is the estimated advantage of the current state and action computed by the discounted cumulative reward, as defined generally by:

$$\hat{A}_t = \sum_{l=0}^{\infty} \gamma^l f(s_{t+l}, r_{t+l}) \quad (4.3)$$

where γ is the discount factor and function f combines the instantaneous reward for time step t with its approximated state value. The \hat{A}_t is computed by using Generalized Advantage Estimation (GAE) [41], which affects reducing its variance by accounting for both cumulative reward and value function estimates (details are omitted due to brevity,

Algorithm 1: Policy Training via PPO

Input:

- Simulated environment env ;
- Number of train iterations E ;
- Number of Episodes K ;
- Episode Length T

Output:

- Learned parameters $\hat{\theta}$ for the control policy π

```

1 Initialize  $\hat{\theta}$ ;
2 for  $iter = 1$  to  $E$  do
3    $env \leftarrow \text{randomizedModify}(env)$ 
4    $\mathcal{D} \leftarrow \emptyset$ 
5   for  $k = 1$  to  $K$  do
6      $trajectory \leftarrow \text{Run policy } \pi_{\hat{\theta}} \text{ in environment } env \text{ for } T \text{ timesteps};$ 
7     Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ ;
8     Add  $trajectory$  to  $\mathcal{D}$ ;
9   Optimize  $L^{\text{CLIP}}(\theta)$  w.r.t.  $\theta$  on data  $\mathcal{D}$ ;
10   $\hat{\theta} \leftarrow \theta$ ;

```

for full description of PPO algorithm we refer the reader to [4]).

Our designed dense reward function combines the energy consumption criteria and QoS objective in robot operation, and is presented by:

$$r_t = -\frac{P_t^{mech} + P_t^{comp}}{v_t} - \exp(\eta) \cdot (R - Q_t^{critical}) \quad (4.4)$$

where P_t^{mech} and P_t^{comp} represent the measured power usage of the robot's mechanical and computational components, respectively, v is the robot's speed, R is the input frame rate of the event-camera and $Q_t^{critical}$ is the QoS of the most critical application at time step t . Hyperparameter η is the normalization factor to balance the energy consumption and QoS criteria. This parameter exponentially penalizes the QoS violation (if any) by considering the actual throughput of the most critical application with the required throughput. The most critical application is defined similar to [3] as the one with the minimum idle time (or equivalently maximum execution time).

The objective function of (4.2) is stochastically maximized by gradient ascent to find model parameters $\hat{\theta}$ that yield maximum cumulative reward. The update step for θ can be described as:

$$\theta \leftarrow \theta_{old} + \alpha \nabla_{\theta} L_{\text{CLIP}}(\theta_{old}) \quad (4.5)$$

where the learning rate α controls the update size.

Algorithm 2: Controller cycle at time t **Input:**

- Power measurements (P_t^{mech}, P_t^{comp});
- Applications' information $apps_t$;

Output:

- Predicted optimal speed s_{t+1} ;
- Predicted optimal CPU frequency f_{t+1} ;

- 1 **Constant:** Learned parameters $\hat{\theta}$ for the control policy;
- 2 $Q_t^{critical} \leftarrow$ Find QoS of the most critical application in $apps_t$;
- 3 $state_t \leftarrow [P_t^{mech}, P_t^{comp}, Q_t^{critical}]$;
- 4 $s_{t+1}, f_{t+1} \sim \pi_{\hat{\theta}}(\cdot | state_t)$;

Algorithm 1 shows the training process of the policy model via PPO algorithm. The simulated environment env represents the system dynamics and is given to the algorithm as input. This environment instance provides a class function `step` that applies a given action list on the system and returns next state along with the computed reward in (4.4). Inputs also include the number of training iteration E , number of episodes in each train iteration K and number of timesteps in each episode T . The output of this training procedure is the learned parameters for the control policy, ready to be deployed on the robot control loop. At each training iteration, we modify the simulated environment randomly (line 3) to obtain a new complexity level for the external environment and applications' setup. This is essential to achieve a generalized and robust control policy that is able to respond to unseen states of internal running applications and external conditions. Then, a dataset containing the sampled episodes are collected (lines 5 to 8) by running the environment `step` function repeatedly. Then, the dataset is used in the stochastic optimization (4.5) of the clipped objective function of (4.2) (line 9). Finally, the optimized parameters are set to $\hat{\theta}$ to be used in the next iteration (line 10).

After training the model, we evaluate its performance in robot operation. Algorithm 2 summarizes the control loop that performs periodic updates to the system configurations, i.e., speed and CPU frequency. The inputs to the algorithm are the power, p_t , and application information, $apps_t$, as shown in Figure 4.1.1, where the current configuration is read from the system. The output of the algorithm is the new speed and CPU frequency configuration based on the inference process, i.e., s_{t+1} and f_{t+1} respectively. The learned parameters of the model are considered constant parameters, denoted as $\hat{\theta}$. At every control cycle, the power measurement sensory data is read from the sensors and a list of all

running applications is retrieved from the operating system. After that, the QoS of the most critical application is identified based on Line 2 of the control algorithm. The system state is recorded in *state* (line 3), which includes power measurements and the QoS of the most critical application, $Q_t^{critical}$. Subsequently, the new speed and CPU frequency are inferred from the learned policy by sampling from the output Gaussian distributions for the speed and frequency configurations (line 4).

5 Results

5.1 Experimental setup

For the purpose of experimentation, we employed the robotic system depicted in Figure 5.1.1. A 3000Kv brushless DC motor is used to provide the robot's propulsion, with the motor's speed controlled by an Electronic Speed Controller (ESC) connected to a Pixhawk (Pix32) controller. The robot is equipped with an NVIDIA Jetson TX2, a power-efficient embedded computing platform that supports the processing of applications required for autonomous and intelligent functions. Additionally, the system features a DAVIS-346 event camera, which captures both intensity images and a high-frequency stream of asynchronous events.

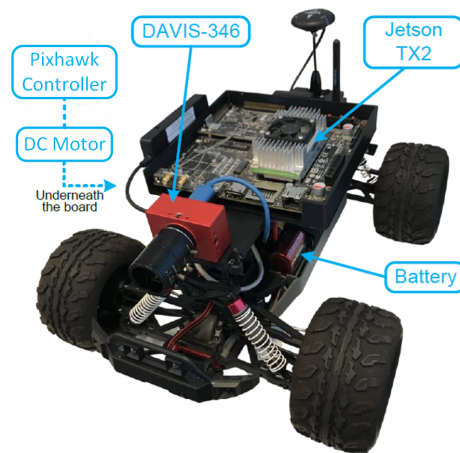


Figure 5.1.1: Demonstrator setup

We conducted experiments with the proposed model across three distinct environments, classified as i) low, ii) moderate, and iii) high complexity levels. As the com-

plexity of the environment increases, the number of generated events also rises. Each environment has an inherent complexity level that directly influences the size of the event batches in the incoming stream. More complex environments naturally lead to larger event batches. Moreover, for any given complexity level, variations in the robot's speed also affect the batch sizes, with higher speeds resulting in larger event batches. The association between the number of events per batch, the complexity of the environment, and the speed of the robot is visually depicted in Figure 5.1.2.

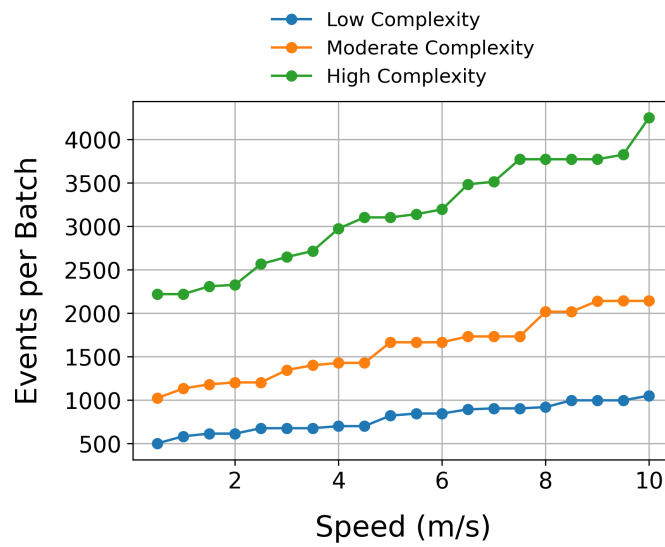


Figure 5.1.2: Number of events per batches vs. speed of robot in different complexities

5.2 Simulation Results

The parameters listed in Table 5.2.1 are crucial for understanding the mechanical power modeling. The values represent typical settings used in the analysis and are essential for accurate modeling and simulation. Each parameter affects different aspects of the model, and adjusting these values can lead to changes in the results.

Table 5.2.1: Key Variables in Mechanical Power Modeling

Parameter	m	Cr	Cv	Cd	Af	ρ
Value	3.7 (kg)	0.031 (N)	0.023 ($N \cdot s/m$)	0.68	0.1 (m^2)	1.2 (kg/m^3)

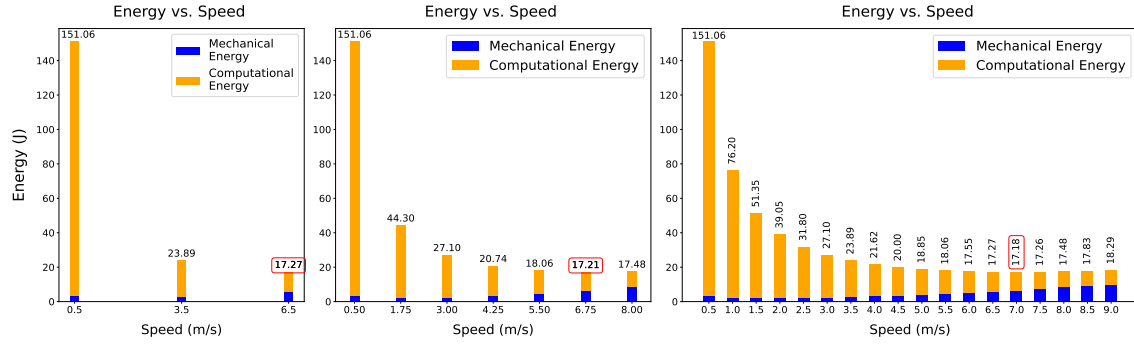


Figure 5.2.1: Energy consumption for different speed granularity configurations in low complexity

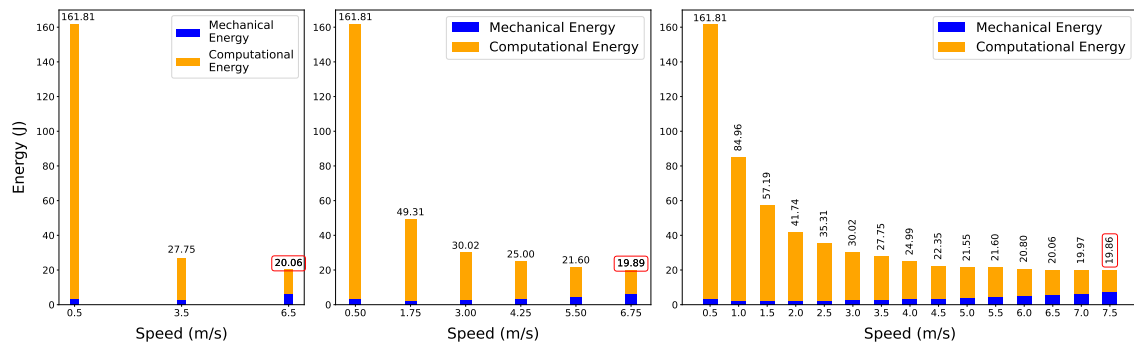


Figure 5.2.2: Energy consumption for different speed granularity configurations in high complexity

Figures 5.2.1 and 5.2.2 illustrate the relation between computational and mechanical energy consumption and speed across various environmental complexities. Notably, energy consumption peaks at the lowest speed, then declines as speed increases, reaching optimal points. However, after surpassing these optimal points, energy consumption starts to rise again with further increases in speed. In the high-complexity environment shown in Figure 5.2.2, increasing the speed beyond 7.5 (m/s) prevents the system from meeting the QoS standards. Conversely, in the low-complexity environment depicted in 5.2.1, exceeding the optimal speed increases energy consumption; however, the system still meets the QoS requirements. Additionally, as the speed intervals decrease—resulting in a finer-grained speed spectrum in the right-side graphs—the accuracy of the results improves, allowing for a more precise identification of the optimal energy point. The increased granularity in speed intervals allows for a finer resolution in capturing energy dynamics, ultimately leading to more informed decisions regarding energy efficiency. The enhanced accuracy in the plots highlights the significance of carefully selecting speed ranges when

analyzing energy performance, as it facilitates a more detailed examination of how varying speeds influence energy consumption in the given system.

Building on this energy analysis, Figure 5.2.3 examines the trade-off between energy estimation error and computational time across different speed granularity intervals. As the speed granularity interval decreases (increasing the granularity of the analysis), the error in energy consumption reduces. Additionally, the computational time increases significantly with smaller granularity, reflecting the trade-off between accuracy and computational cost. The observed trends emphasize the balance required to achieve accurate energy estimations while managing computational resources effectively.

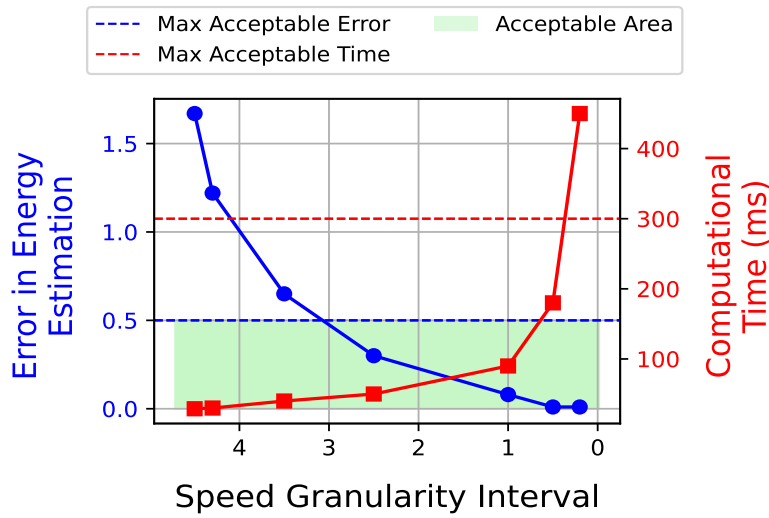


Figure 5.2.3: Error in energy estimation and computational time vs. speed granularity interval

Since the mechanical energy consumption in Figures 5.2.1 and 5.2.2 is relatively small compared to computational energy, we present Figure 5.2.4 to show the trend of mechanical energy variation across different speed settings. It is observed that When the robot operates at its lowest speed, energy consumption tends to be higher. This is due to the fact that the power integration occurs over a longer duration, so even though the power requirements are minimal, the extended period significantly impacts the total energy usage. On the other hand, at higher speeds, the motor requires much more electrical power, leading to increased energy consumption as a result of the larger power values during the integration process. As shown in Figure 5.2.4, when considering only mechanical energy, the optimal speed for minimizing mechanical energy consumption is 2 (m/s). However,

this differs from the optimal speed when both mechanical and computational power consumption are taken into account.

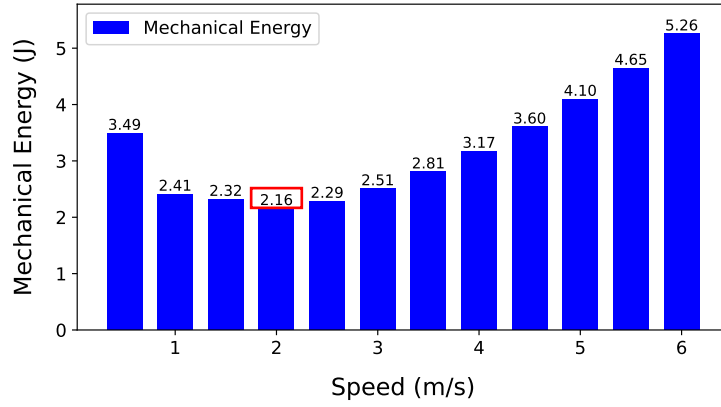


Figure 5.2.4: Mechanical energy for different speed configurations

To illustrate the impact of changes in robot speed and CPU voltage/frequency on application QoS, we conducted experiments at various speeds, gradually increasing the CPU voltage/frequency at each speed and measuring throughput in batches per second. This analysis is crucial because, as discussed in the methodology chapter, maintaining an acceptable QoS is a strict constraint when adjusting mechanical and computational parameters and must be carefully monitored at runtime. Figure 5.2.5 illustrates the relationship between energy consumption and application throughput (batches per second) in a high-complexity environment. It demonstrates how variations in speed and frequency influence both energy consumption and throughput. The frequency is varied from 0.4992 GHz to 2.0352 GHz, while the speed remains constant between 6 and 9 m/s. In the red curve, which spans from P1 to P3, the speed is incrementally increased while maintaining the frequency. The point labeled P* represents the energy-optimal configuration, where the energy consumption is minimized at a speed of 7.5 m/s and a frequency of 2.03 GHz, while still satisfying the throughput requirements.

Figures 5.2.6 and 5.2.7 present a comprehensive comparison of the average values of batch per second and PLPT across various frequency and speed levels. The data illustrated in these figures reveal insightful trends and relationships that are critical for understanding system performance. As anticipated, the PLPT demonstrates a noticeable decrease as frequency increases, indicating that higher frequencies facilitate quicker processing times and enhance overall efficiency. This trend is particularly important in applications that

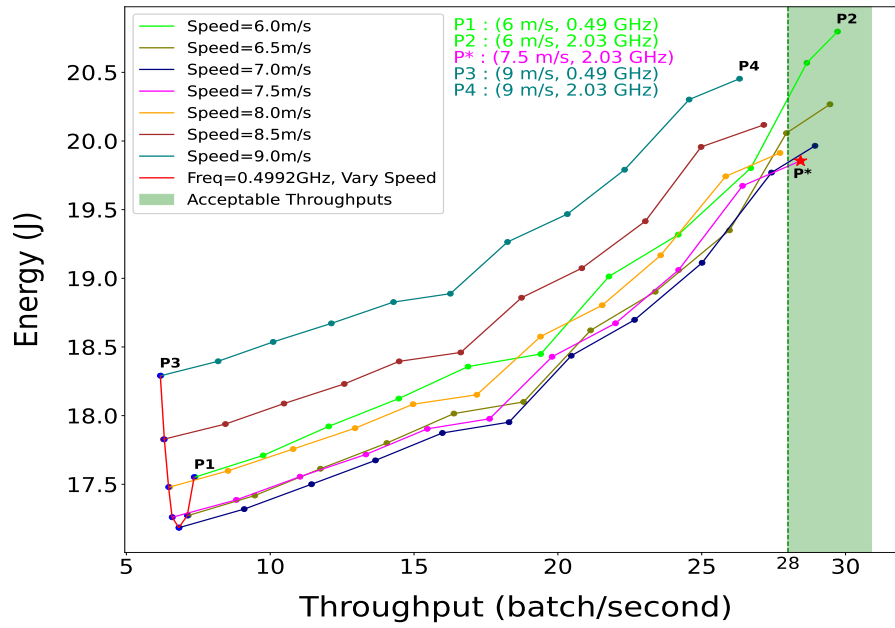


Figure 5.2.5: Energy consumption vs. application throughput for high complexity

require rapid data handling and processing, where reduced PLPT can lead to improved responsiveness and performance.

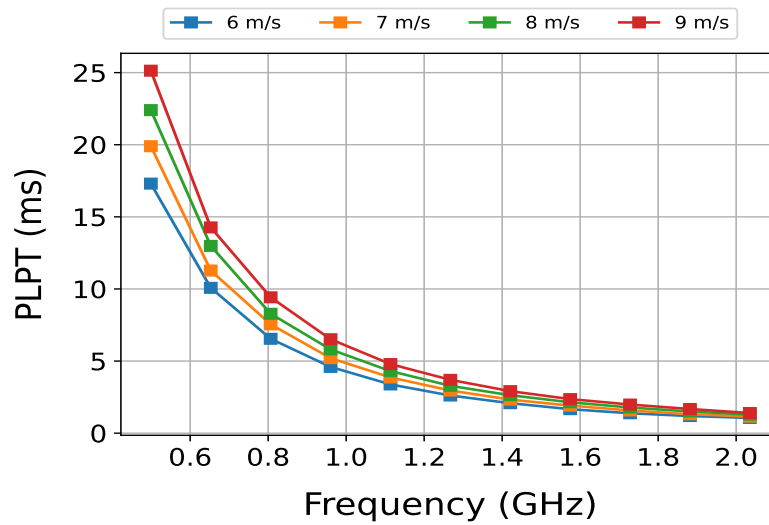


Figure 5.2.6: PLPT compared to frequency level for different speed settings

On the other hand, the batch per second metric demonstrates an upward trend as the frequency rises, ultimately reaching a saturation point of $R = 30$. As frequency increases, the system demonstrates an enhanced ability to handle larger volumes of data more efficiently, leading to a higher number of batches processed per second. This upward trend highlights the positive correlation between frequency and processing efficiency, suggest-

ing that optimizing frequency settings can significantly improve system performance.

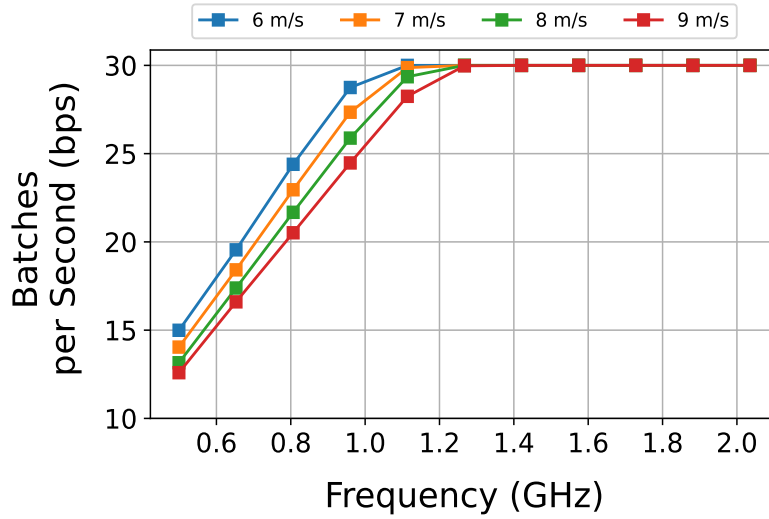


Figure 5.2.7: Average batch per second vs. frequency level for different speeds

5.3 Path Planning Results

In our implementation of the path planning approach using the Markov Decision Process, we analyze the impact of varying environmental complexities on the efficiency of the robot's navigation.

Figure 5.3.1 illustrates the paths taken by a mobile robot navigating through an environment characterized by varying complexities, including low, moderate, high, and extreme regions. As observed, the influence of different computational energy consumption resulting from these complexities significantly affects the efficiency of each path. If we only consider the mechanical aspects of movement, the optimal solution might appear to be the straight path. However, when we account for computational energy consumption, the optimal path changes, revealing that what may seem like the most direct route is not necessarily the most efficient. In this case, Optimal Path, despite not being the most direct route, demonstrates a lower total energy consumption compared to other paths that traverse more complex areas. This highlights the critical importance of integrating both mechanical and computational factors in path planning, as they play a crucial role in determining the overall efficiency and effectiveness of the route taken by the robot.

By implementing the MDP method, the optimal path for the environment illustrated

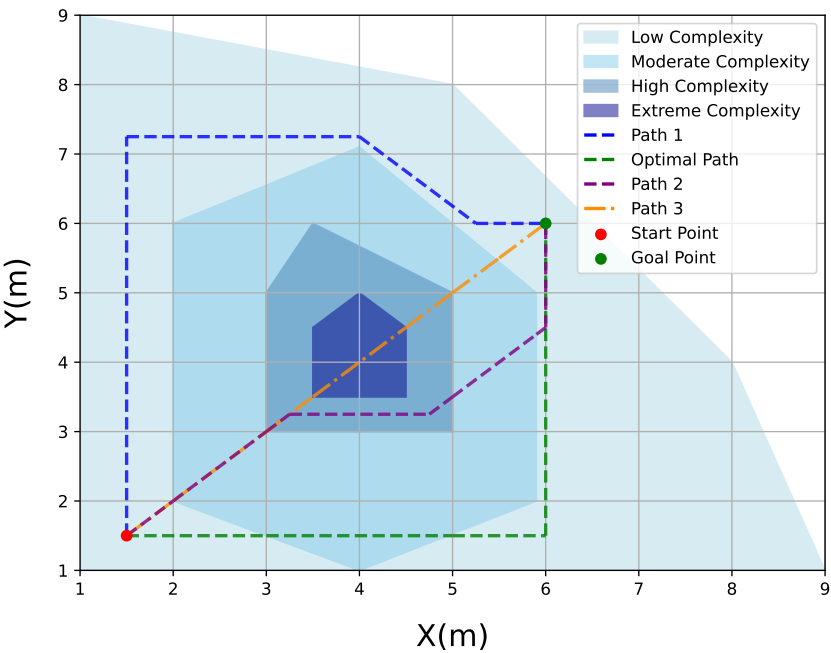


Figure 5.3.1: Paths in complex environment

in Figure 5.3.1 has been successfully determined. To assess the effectiveness of this approach, we examined three alternative paths and compared their total energy consumption against the optimal path identified by the MDP. Table 5.3.1 presents a summary of the energy consumption associated with each path, highlighting the efficiency gains achieved through the MDP-based decision-making process.

Table 5.3.1: Energy Consumption for Different Paths

Path	Total Energy Consumption
Path 1	296.33 (J)
Path 2	253.40 (J)
Path 3	274.63 (J)
Optimal Path	248.56 (J)

Additionally, Figure 5.3.2 illustrates the results of path planning in a more extended and complex environment. In this scenario, two distinct paths were evaluated: one generated by the MDP method and another alternative route. The MDP approach enables the robot to make informed decisions based on its current state and surrounding conditions. As the figure shows, the path resulting from the MDP method achieves lower energy consumption to reach the goal point. This demonstrates that the optimal path is not necessarily the shortest or most direct route; rather, it accounts for obstacles and environmental

complexities, resulting in a more energy-efficient trajectory.

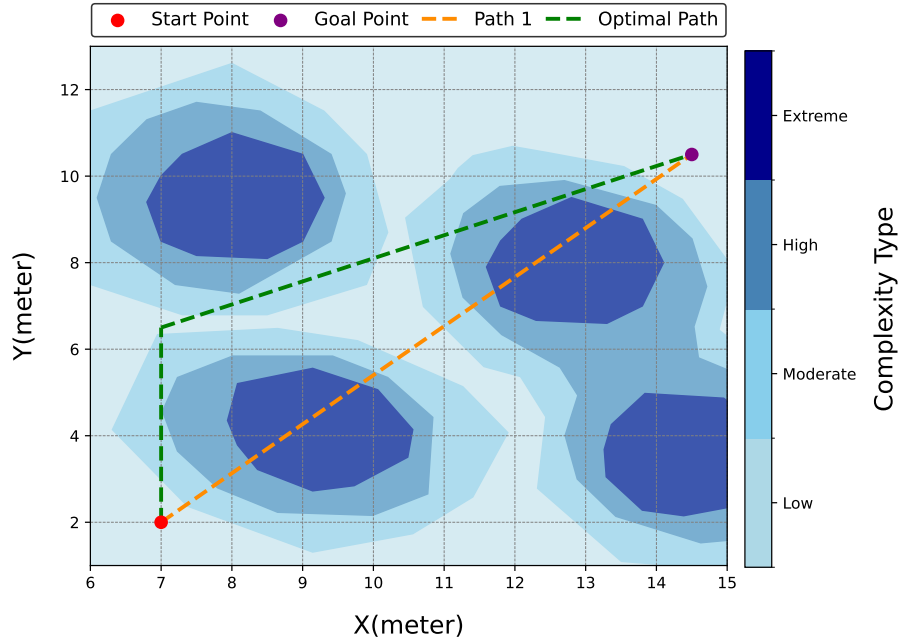


Figure 5.3.2: Comparison of two paths in a more complex environment

5.4 Reinforcement Learning Results

As mentioned in the Simulation section, Figure 5.4.1 illustrates that the energy-optimal configuration varies significantly and non-monotonically across different levels of environmental complexity. This observation highlights the dynamic nature of energy management in robotic systems operating under diverse conditions. As the external environment transitions from low to extreme complexity, the computational and mechanical demands of the system increase. In particular, the number of events processed within each batch rises, which in turn necessitates adjustments in system configuration to sustain optimal performance. Consequently, different configurations emerge as energy-efficient under varying conditions, each capable of minimizing power consumption while ensuring that the QoS constraint is satisfied.

In this dynamic context, our proposed controller operates as an adaptive decision-making entity that continuously co-manages the configuration of both the mechanical and computational subsystems of the robot—specifically, the motor speed and the CPU operating frequency. This joint management is crucial due to the interdependent nature of

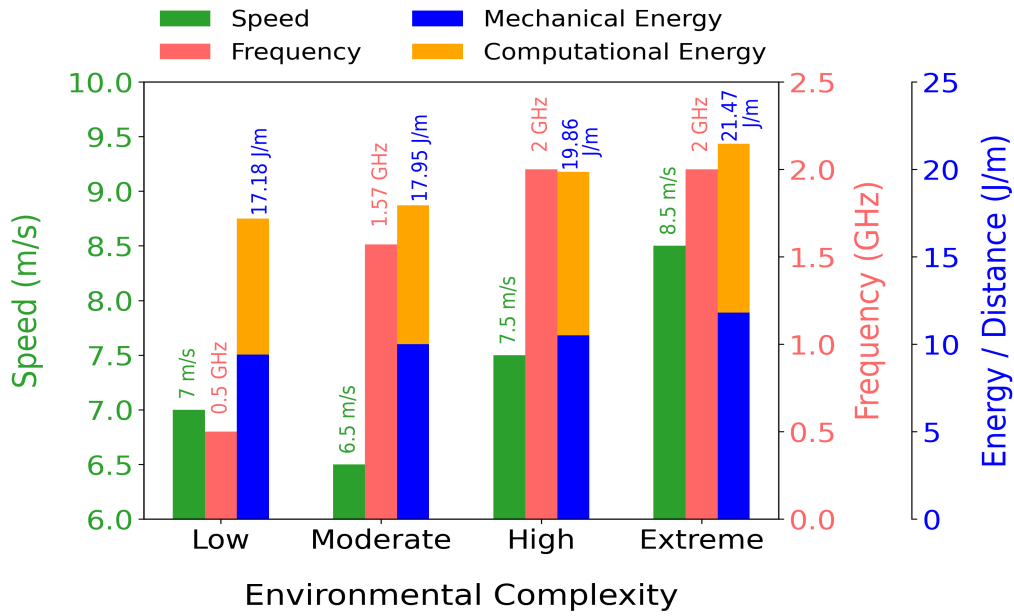


Figure 5.4.1: Optimal configurations for different complexities

energy consumption in these components; changes in one can influence the energy behavior of the other. The controller functions at run-time, enabling it to respond to real-time variations in workload and environmental stimuli. To achieve this, we continuously monitor real-time power consumption metrics from the motor and CPU, as well as actual QoS outputs from the running robotic applications. Note that we examined three distinct applications commonly used in robotics: Harris Corner Detection, Optical Flow and YOLO, each representing different computational and perception demands in robotic systems. To begin with, the Harris Corner Detection algorithm is a classical method for identifying interest points in visual data. Introduced by Harris and Stephens [42], it detects corners by analyzing changes in image intensity across multiple directions within a local window. These corners serve as stable features for tasks such as tracking and mapping. Due to its low computational overhead, Harris Corner Detection is well-suited for real-time robotic applications, particularly in event-based vision, where speed and efficiency are critical. In contrast, Optical Flow focuses on estimating the motion of objects or features across consecutive frames. Based on the assumption of brightness constancy and spatial coherence, it computes per-pixel motion vectors that reflect the dynamics of the scene. As detailed by Szeliski [43], optical flow plays a fundamental role in understanding movement and scene structure, which is especially valuable in navigation and obstacle avoidance. However, its

higher computational complexity makes it a more demanding application for embedded robotic systems. Finally, YOLO (You Only Look Once) is a modern, deep learning-based object detection algorithm that significantly differs from traditional computer vision approaches. As described by Redmon et al. [44], YOLO processes an entire image in a single forward pass to simultaneously predict object classes and bounding boxes. This makes it highly efficient and suitable for real-time robotic scenarios requiring fast and accurate object detection. However, YOLO also imposes substantial computational loads, thus presenting a more challenging test case for evaluating energy-efficient resource management strategies.

In our evaluation of these three applications, Harris Corner Detection stood out for its efficiency, requiring significantly fewer processing cycles per event compared to both Optical Flow and YOLO. While Optical Flow proved more efficient than YOLO, it still consumed more cycles per event than Harris. YOLO, although highly capable in terms of detailed object detection, incurred the highest number of processing cycles due to its deep learning-based architecture and complex computation, thereby leading to greater energy consumption. In addition to its computational efficiency, Harris Corner Detection also demonstrated superior performance in terms of Quality of Service (QoS), consistently achieving the desired target of processing 30 batches per second. This makes it particularly well-suited for resource-constrained environments where fast and reliable feature detection is essential. In contrast, the computational demands of YOLO, while justified in use cases requiring rich semantic understanding, limit its applicability in energy-constrained robotic platforms. Therefore, from both efficiency and QoS perspectives, Harris Corner Detection emerges as the most favorable option among the three, particularly in scenarios where real-time responsiveness and energy optimization are critical.

Now that we have established an understanding of the applications used for event-based analysis and processing, we turn our attention to the details of the PPO model employed in our proposed method. Table I represents the PPO model's parameters. We utilized 64 neurons per layer across two layers. We adjusted the training-related parameters, such as the episode count and the number of training iterations, denoted as K and E , respectively, to achieve acceptable accuracy in the training model. The discount factor in 4.3 is set to $\gamma = 0.99$. The learning rate, denoted as α in (4.5) is adjusted to 10^{-3} .

Reward scaling ($\eta = 2.5$) enhances sensitivity to QoS variations, and a gradient cap of 2 ensures smooth convergence, preventing instability. The timing epoch for executing Algorithm 2 is set to 50 ms to ensure an appropriate response time for adjusting the speed and frequency in real time. We assessed our proposed approach by comparing it with three distinct control strategies to demonstrate its effectiveness:

- **Separate Optimization (SO):** This method involves independently optimizing the mechanical and computational components of the robot. Specifically, the robot's speed is maintained at a constant value that minimizes mechanical energy consumption (e.g., 2 m/s), while the CPU frequency is controlled independently by an adaptive runtime controller, such as the one proposed by Angioletti et al. [27]. This decoupled approach simplifies the optimization process but may overlook the interdependencies between mechanical and computational energy consumption, potentially leading to suboptimal overall energy efficiency.
- **Hill Climbing (HC):** Hill Climbing is a local search algorithm commonly used in optimization problems. It aims to find a solution that either maximizes or minimizes a given objective function. Starting from an initial point, the algorithm iteratively explores neighboring solutions and moves in the direction that most improves the objective function. In the context of mobile robots, HC can be applied to adjust parameters such as speed and CPU frequency to minimize energy consumption. However, due to its local search nature, HC may converge to local optima and miss the global optimum, especially in complex energy landscapes [24].
- **Proactive Optimization (PO):** The proactive energy optimization method proposed by Shahsavari et al. [3] introduces a co-management strategy that simultaneously considers both mechanical and computational components. This approach leverages internally trained models that adapt dynamically to forecast the future energy consumption of the robot's components. Using these forecasts, the optimal values for the robot's mechanical speed and CPU frequency are calculated in real-time. By adjusting parameters in advance of anticipated energy requirements, this method has shown substantial gains in energy efficiency, achieving up to a 36.34% reduction in energy usage when compared to conventional techniques.

Table 5.4.1: PPO’s Model Parameters

Parameter	Values	Parameter	Values
#Neurons	64	#Episode (K)	10^4
#Layers	2	#Train Iteration (E)	10^6
Discount Factor (γ)	0.99	Learning Rate (α)	10^{-3}
Reward Scaling Factor (η)	2.5	Total Timesteps	77×10^4

Figure 5.4.3 and Figure 5.4.2 illustrate the relationship between robot speed and energy consumed per meter traveled, in environments of high and low complexity, respectively. These plots represent the outcome of the training process in which an internal model learns to predict energy consumption across varying speed configurations. Each blue dot corresponds to a sampled configuration evaluated during training, while the red-highlighted point marks the configuration with minimum energy consumption per distance. In both scenarios, the model demonstrates rapid convergence, reaching a near-optimal solution in fewer than 10^6 training iterations. This indicates that the learning algorithm effectively generalizes energy-performance patterns even under varying environmental complexities. Notably, the optimal speed for minimal energy expenditure shifts slightly depending on the complexity level—7.08 m/s for the low-complexity case and 7.58 m/s for the high-complexity one—highlighting the model’s sensitivity to environmental conditions. These results validate the efficiency of the training approach and the predictive power of the learned model in optimizing energy usage in dynamic robotic systems.

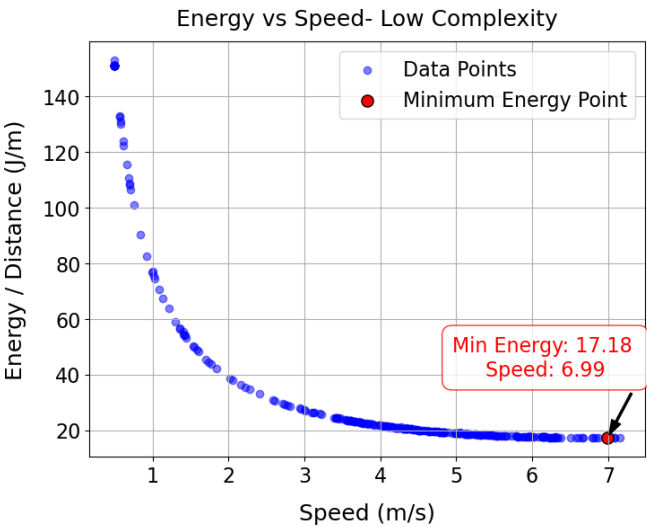


Figure 5.4.2: Energy vs. speed in low complex environment for PPO model

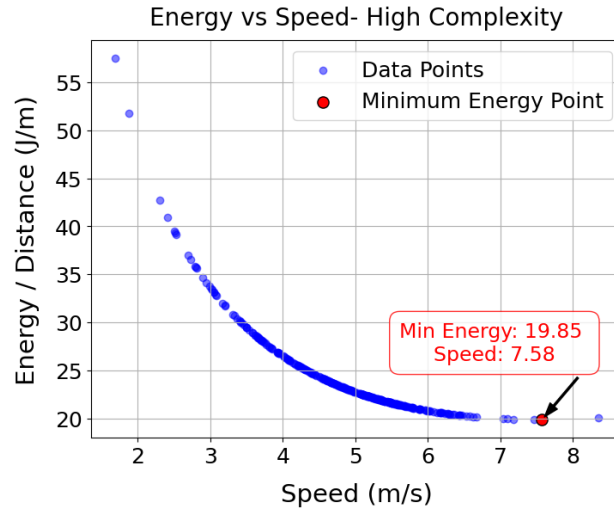


Figure 5.4.3: Energy vs. speed in high complex environment for PPO model

On the other hand, Figure 5.4.4 presents a comparison of energy consumption per distance over time for four optimization methods: SO, HC, PO and our approach in the highly complex environment. The PO and HC method use 0.5 speed interval in the optimization process. The settling time of near optimal energy/distance vales is also depicted in the figure. As it can be seen our controller very fast reaches a near-optimal speed/frequency configuration for enery per distance compared with the other methods by one inference on the trained model. SO method, which focuses on separately optimizing mechanical and computational energy without considering their interdependence, results in different configurations of motor speed and CPU frequency, leading to higher overall energy consumption. This highlights the importance of jointly considering the dependency between mechanical and computational energy consumption in the optimization process. The HC method has the longest response time with iteratively changing the systems configuration over time resulting in oscillation of energy per distance factor. This is due to inherent trial and error process in HC algorithm and possibilities to be trappen in local minima. The PO gives relatively better settlement time compared with HC, however due to long computation time for exhaustive search of the optimal configuration, the settlement time is higher than our proposed approach.

Figure 5.4.5 and 5.4.6 illustrates the total energy usage and overall mission time for the robot traveling along a straight path with three different lengths, namely 100, 250, and 500 meters, based on our proposed method compared to the other three methods. As

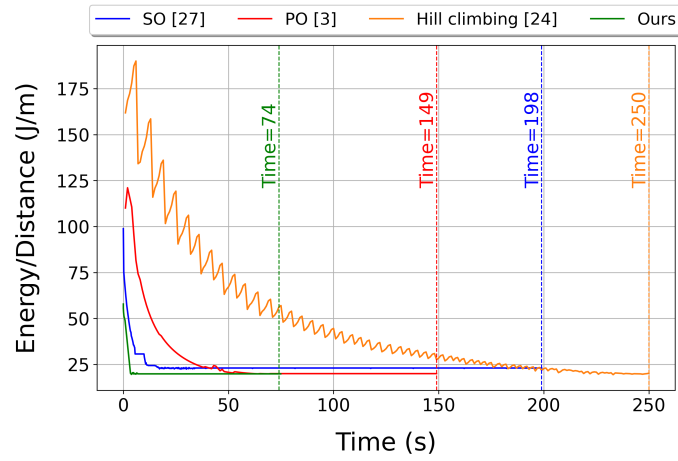


Figure 5.4.4: Energy vs. time for different methods

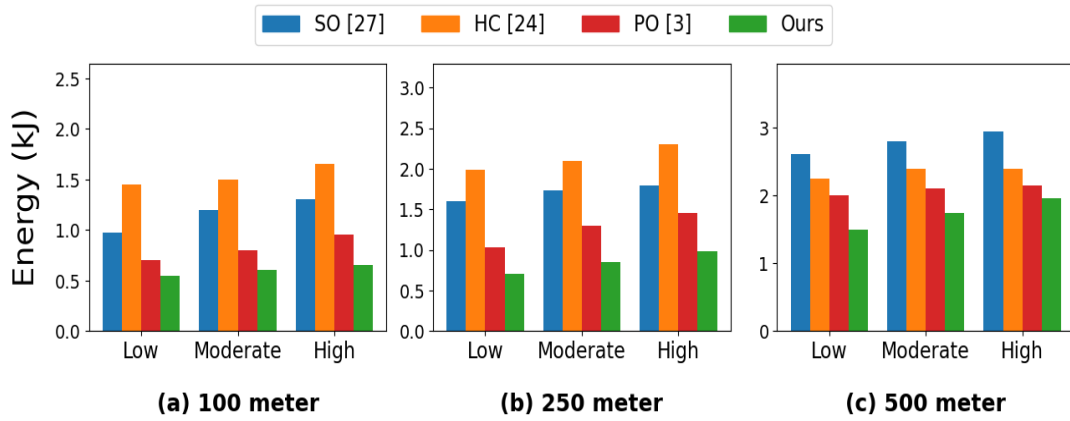


Figure 5.4.5: Energy consumption in different environment complexities

shown, our method outperforms the others in terms of both overall energy consumption and mission time. The HC method exhibits the highest energy consumption due to the redundant energy expended during the trial-and-error process. In the SO method, since the speed parameter is constant, overall energy consumption depends on the travel length and the separately optimized computing component of the robot, which is related to the complexity of the environment and the processing of the generated data. The PO method shows relatively better energy consumption compared to HC and SO; however, it is evident that its overall energy consumption is still higher than that of our method. The improvement of our method over PO can be attributed to two factors. First, the settlement time of PO causes a delay in achieving near-optimal configurations for frequency and robot speed. Second, the computational load of PO optimization imposes an additional energy burden on the system. Overall, our proposed method enhances energy efficiency

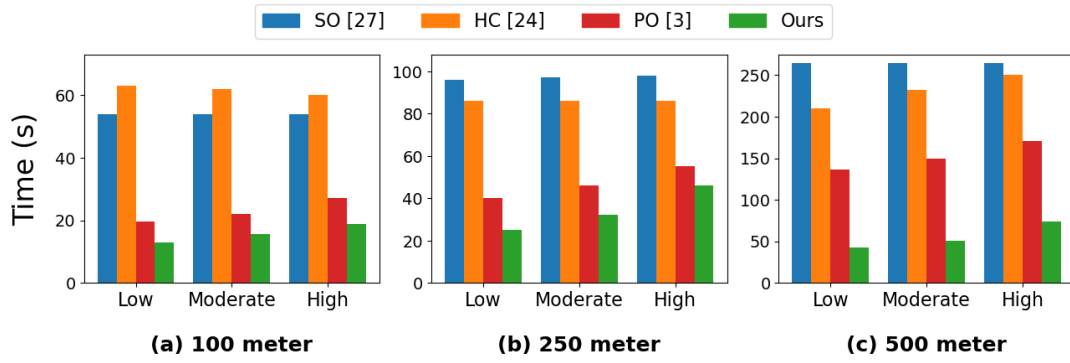


Figure 5.4.6: Mission time in different environment complexities

by 47.76%, 60.86%, and 26.33% for a 100 m length in comparison with SO, HC, and PO, respectively. Moreover, the improvements are 50.84%, 60.30%, and 33% for 250 m, and 37.98%, 26.39%, and 16.98% for 500 m, compared to SO, HC, and PO, respectively.

It can be observed that the mission time obtained for the proposed method is superior to that of the other approaches. This indicates that, despite the optimization function not being focused on process timing, the proposed method significantly reduces mission time by adjusting the speed. The rationale for this in the SO method is evident, as it employs a constant optimal speed that only considers mechanical energy consumption. The improvement in mission time compared to the HC method arises from the long response time required to achieve minimal energy per distance, which leads to several unnecessary low-speed cases until reaching the near-optimal speed. Additionally, due to the continuous searching nature of HC, some minor, unnecessary slow-down processes affect the overall mission time. When comparing the proposed approach to the PO method, the proposed method also demonstrates better overall mission time. This can be attributed to the fact that, while PO is proactive, the variation in computational workload indirectly influences the optimizer to select lower speeds to achieve better energy per distance, thus impacting mission time. Overall, our proposed method enhances mission time by 70.83%, 74.36%, and 31.13% for a length of 100 m in comparison with SO, HC, and PO, respectively. Furthermore, the improvements are 64.68%, 60.08%, and 28.10% for 250 m, and 78.86%, 75.98%, and 63.70% for 500 m, compared to SO, HC, and PO, respectively.

Figure 5.4.7 presents a comparative analysis of the execution time between the proposed learning-based control method and the exhaustive search strategy employed in the Proactive Energy Optimization (PO) approach. The comparison is carried out across a

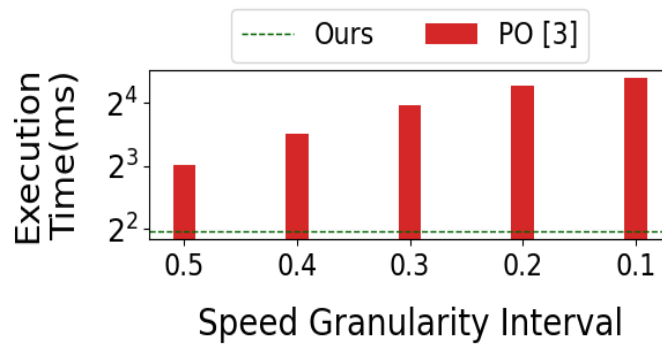


Figure 5.4.7: Execution time vs. different speed interval

range of arbitrarily selected speed discretization intervals, which are essential in the PO method to define the granularity of the search space used for optimizing actuator commands. In this context, finer discretization corresponds to smaller speed intervals, which theoretically leads to more precise optimization results but at the cost of increased computational effort. As shown in the figure, the execution time for the PO method increases rapidly with finer speed resolutions. Notably, the Y-axis of the plot is logarithmic, highlighting the exponential growth in execution time as the interval becomes smaller. This trend underscores a critical scalability issue inherent to the matrix-based prediction mechanisms employed by the PO method, which involve exhaustive evaluation over a combinatorially expanding set of possible configurations.

Such exponential growth in execution time presents a significant limitation for real-time robotic applications, where rapid decision-making is essential for responsiveness and safety. In contrast, the proposed method, which relies on a trained neural network for direct inference, exhibits a strikingly different behavior. Its execution time remains virtually constant across all discretization intervals, as indicated by the flat horizontal line in the figure. This constancy arises because the inference process in the neural network does not depend on the size or granularity of the configuration space. Once trained, the model can generalize across continuous input values without the need to explicitly enumerate and evaluate each possible configuration. As a result, the controller maintains a consistently low execution time regardless of the resolution of the decision space.

This performance characteristic makes the proposed method particularly well-suited for deployment in real-time and resource-constrained robotic systems, where maintaining low latency is crucial. It not only eliminates the need for costly online searches but also

ensures scalability when new features or control dimensions are introduced. Ultimately, the findings illustrated in Figure 5.4.7 highlight a key advantage of learning-based control strategies over traditional exhaustive optimization methods, particularly in terms of computational efficiency and practical deployability.

6 Discussion

This chapter provides a comprehensive discussion of the research findings. Section 6.1 highlights the key contributions, including the proposed simulation framework and reinforcement learning-based co-management strategy for improving energy efficiency. Section 6.2 addresses the limitations, such as the need for extending the approach to different robotic setups, considering the energy consumption of additional systems, and exploring the impact of disproportionate energy demands in various tasks. Finally, Section 6.3 explores future research directions, including testing the method in real-world environments, enhancing the system’s robustness, and expanding the framework to multi-robot systems to improve collaborative energy management in large-scale applications.

6.1 Contribution of this study

In this research, we presented a simulation framework for estimating the instantaneous power consumption of a mobile robot by incorporating both mechanical and computational energy usage, along with their interdependence. Unlike traditional energy estimation models that often focus on one aspect of power consumption, our approach considers both components simultaneously, providing a more holistic view of the robot’s energy profile. A key feature of the proposed framework is its multi-fidelity power estimation approach, which allows dynamic adjustments in accuracy and computational cost. By tuning the fidelity level in real time based on environmental conditions and available computational resources, the framework ensures a balance between precise energy estimation and computational efficiency. This adaptability makes it particularly well-suited for real-time predictive decision-making in various applications, such as model-based reinforcement learning (RL) and model predictive control (MPC), where energy-aware

decision-making is crucial. Experimental validation demonstrated the effectiveness of the proposed framework in accurately estimating energy consumption across different fidelity levels, reinforcing its potential for practical deployment in real-world robotic systems. Furthermore, the results highlighted a clear trade-off between estimation accuracy and computational cost, emphasizing the need for careful selection of fidelity levels to achieve optimal energy-efficient operation.

In addition to energy estimation and management, this study investigates the impact of environmental complexity on robotic navigation by incorporating path planning within a Markov Decision Process (MDP) framework. By modeling path planning as a sequential decision-making problem, we account for how varying levels of environmental complexity influence energy consumption and navigation efficiency. This approach allows the robot to adapt its decisions in real-time, balancing path optimality with energy constraints based on its perception of the surrounding environment. The integration of energy-aware decision-making into the planning process enables more efficient navigation, particularly in dynamically changing or resource-constrained scenarios.

Extending the capabilities of the proposed simulation framework, we developed a reinforcement learning-based approach for simultaneously handling the mechanical and computational energy of battery-powered mobile robots. Our method leverages a feed-forward neural network trained on relevant feature data, allowing the system to adaptively and efficiently regulate energy consumption in response to environmental changes. Unlike conventional optimization methods, which often suffer from high computational overhead and limited scalability, our RL-based approach enables fast, scalable, and responsive energy management. Experimental results demonstrated significant improvements in execution time, with the proposed model markedly outperforming baseline methods. More importantly, the approach achieved up to a 60.02% reduction in overall energy consumption compared to other optimization techniques, making it a highly effective solution for real-time robotic applications. This level of efficiency is particularly valuable in scenarios where robots must rapidly adjust their internal configurations to maintain both performance and energy efficiency under changing environmental conditions.

The findings of this research contribute meaningfully to the broader goal of energy-efficient robotic operation by offering a flexible and adaptive tool for both energy esti-

mation and management. By addressing the interdependence between mechanical and computational components and introducing a multi-fidelity simulation framework, this work provides a practical approach for real-time, energy-aware decision-making. The reinforcement learning-based controller further enhances this capability by enabling fast and adaptive energy regulation in response to dynamic environmental changes. Together, these contributions lay a strong foundation for building more intelligent and efficient robotic systems, capable of operating effectively in a wide range of conditions while maintaining a careful balance between energy consumption and performance.

It is worth mentioning that this research has also led to the publication of two conference papers that communicate the findings to the scientific community. The first paper, titled "Co-Management of Computational and Mechanical Energy in Mobile Robots Using Reinforcement Learning," has been accepted for presentation at the 23rd European Control Conference (ECC). This paper focuses on the development and application of the reinforcement learning-based energy management strategy, highlighting its advantages in terms of adaptability, scalability, and real-time performance. The second paper, "Simulation of Mechanical and Computational Power Consumption in Mobile Robots," has been accepted at the 39th International Conference on Modelling and Simulation (ECMS). It presents the proposed simulation framework in detail, emphasizing its multi-fidelity power estimation capabilities and the importance of considering the interdependence between mechanical and computational energy consumption. Together, these two publications showcase the scope of the research contributions made in this thesis and underline their relevance to the fields of control systems, robotics, and energy-aware computing. Moreover, the acceptance of these papers at reputable international conferences demonstrates the impact and scientific merit of the presented work, contributing to the ongoing advancement of energy-efficient robotic technologies.

6.2 Limitation

While this research makes significant progress toward comprehensive energy estimation and management in mobile robots, certain limitations remain that offer avenues for future work. The proposed simulation framework and reinforcement learning-based controller were developed and validated within a specific robotic setup. Extending the approach to different robot platforms with varying mechanical designs, computational architectures, and operational environments would be essential to fully establish its generalizability.

Moreover, the current framework primarily focuses on mechanical actuators and on-board computational resources, without directly considering the energy consumption of additional systems like sensors, communication devices, and other parts of the robot. Including these components would enable a more holistic and precise estimation of a robot's total energy profile, particularly for complex and sensor-rich systems. Furthermore, while battery constraints were considered indirectly through energy management, integrating battery-aware path planning strategies could enhance the system's ability to make navigation and control decisions that maximize operational longevity. Finally, although reinforcement learning was employed to great effect, exploring alternative machine learning paradigms could further improve adaptability and robustness, especially in highly dynamic or partially observable environments.

6.3 Future Works

There are several promising directions for future research that could significantly expand on the findings of this thesis. One interesting area is to explore how well the proposed energy management method performs in tasks where the demands on either mechanical or computational energy are disproportionately larger. For example, in certain applications, a robot might be required to move substantial distances or perform physically demanding tasks with minimal computational processing, while in other scenarios, the robot may be stationary but needs to execute complex algorithms. Studying these scenarios would provide valuable insights into the flexibility and adaptability of the proposed method across a wide range of use cases, ensuring it can be effectively deployed in both energy-efficient mobility and computationally intensive environments.

Another key direction for future work involves investigating the realism and robustness of the training process. While the current method has been validated in controlled settings, real-world environments are often more unpredictable and complex. The impact of environmental factors such as unexpected terrain variations, sensor noise, and dynamic obstacles could significantly affect energy consumption patterns. Therefore, it would be crucial to assess how much detail and realism are required during the training phase to ensure that the energy management system can handle such challenges. A deeper exploration into the system's ability to adapt to unforeseen environmental changes and disturbances would improve its robustness, enabling more reliable performance in real-world applications.

Lastly, an exciting and high-impact avenue for future research is to extend the energy management approach to multi-robot or swarm systems. In collaborative settings, where multiple robots work together to achieve a common goal, energy efficiency becomes even more critical due to the increased number of moving parts and the complexity of coordinating actions. Exploring how the proposed framework can be adapted for swarm robotics could lead to significant advancements in energy management across a group of robots, enhancing both individual and collective performance. This could have widespread applications in large-scale missions, such as search-and-rescue operations, environmental monitoring, and warehouse automation, where multiple robots are required to work together

efficiently while maintaining optimal energy usage. Developing strategies for effective energy co-management in such settings could not only extend the operational lifetime of robotic teams but also enable new forms of collaborative autonomy, opening up a wealth of possibilities in the field of multi-robot systems.

References

- [1] X. Liu, G. Chang, J. Tian, Z. Wei, X. Zhang, and P. Wang, “Flexible path planning-based reconfiguration strategy for maximum capacity utilization of battery pack,” *Journal of Energy Chemistry*, vol. 86, pp. 362–372, 2023.
- [2] H. Tang, Q. Zhu, B. Qin, R. Song, and Z. Li, “Uav path planning based on third-party risk modeling,” *Scientific Reports*, vol. 13, no. 1, p. 22259, 2023.
- [3] S. Shahsavari, H. Haghbayan, A. Miele, E. Immonen, and J. Plosila, “A coordinated approach to control mechanical and computing resources in mobile robots,” *IEEE Transactions on Robotics*, pp. 1–18, 2024.
- [4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [5] V. Petrichenko, L. Lokstein, G. Thiele, and K. Haninger, “Energy consumption in robotics: A simplified modeling approach,” *arXiv preprint arXiv:2411.03194*, 2024.
- [6] A. Liu, H. Liu, B. Yao, W. Xu, and M. Yang, “Energy consumption modeling of industrial robot based on simulated power data and parameter identification,” *Advances in Mechanical Engineering*, vol. 10, no. 5, p. 1687814018773852, 2018.
- [7] P. Tokekar, N. Karnad, and V. Isler, “Energy-optimal trajectory planning for car-like robots,” *Autonomous Robots*, vol. 37, pp. 279–300, 2014.
- [8] A. Othman, K. Belda, and P. Burget, “Physical modelling of energy consumption of industrial articulated robots,” in *2015 15th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2015, pp. 784–789.
- [9] K. Shivam and J.-C. Hsiao, “A hybrid energy consumption model for industrial robot arms using robot dynamics and data-driven approach,” *International Journal of iRobotics*, vol. 6, no. 2, pp. 9–15, 2023.
- [10] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [11] P. Jiang, J. Zheng, Z. Wang, Y. Qin, and X. Li, “Industrial robot energy consumption model identification: A coupling model-driven and data-driven paradigm,” *Expert Systems with Applications*, vol. 262, p. 125604, 2025.
- [12] J.-S. Shaw and Y.-H. Huang, “Virtual modeling of an industrial robotic arm for energy consumption estimation,” *Adv. Technol. Innov*, vol. 69, pp. 300–305, 2023.

-
- [13] Q. Chang, T. Yuan, H. Li, Y. Chen, X. Wang, S. Gao, H. Ren, X. Zhao, and L. Wang, "A data-driven method for predicting and optimizing industrial robot energy consumption under unknown load conditions," in *Actuators*, vol. 13, no. 12, 2024.
 - [14] M.-H. Haghbayan, A. Miele, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "Performance/reliability-aware resource management for many-cores in dark silicon era," *IEEE Transactions on Computers*, vol. 66, no. 9, pp. 1599–1612, 2017.
 - [15] H. Haghbayan, A. Miele, O. Mutlu, and J. Plosila, "Run-time resource management in cmps handling multiple aging mechanisms," *IEEE Transactions on Computers*, vol. 72, no. 10, pp. 2872–2887, 2023.
 - [16] J. Zhai, C. Bai, B. Zhu, Y. Cai, Q. Zhou, and B. Yu, "Mcpat-calib: A microarchitecture power modeling framework for modern cpus," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.
 - [17] E. Del Sozzo, G. C. Durelli, E. Trainiti, A. Miele, M. D. Santambrogio, and C. Bolchini, "Workload-aware power optimization strategy for asymmetric multiprocessors," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 531–534.
 - [18] A. K. Singh, A. Prakash, K. R. Basireddy, G. V. Merrett, and B. M. Al-Hashimi, "Energy-efficient run-time mapping and thread partitioning of concurrent opencl applications on cpu-gpu mpsoes," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–22, 2017.
 - [19] A. Pathania, Q. Jiao, A. Prakash, and T. Mitra, "Integrated cpu-gpu power management for 3d mobile games," in *Proceedings of the 51st Annual Design Automation Conference*, 2014, pp. 1–6.
 - [20] S. Wegener, K. K. Nikov, J. Nunez-Yanez, and K. Eder, "Energyanalyzer: Using static wcet analysis techniques to estimate the energy consumption of embedded applications," *arXiv preprint arXiv:2305.14968*, 2023.
 - [21] F. Xia, L. Ma, W. Zhao, Y. Sun, and J. Dong, "Enhanced energy-aware feedback scheduling of embedded control systems," *arXiv preprint arXiv:0809.4917*, 2008.
 - [22] X. Mei, Q. Wang, X. Chu, H. Liu, Y.-W. Leung, and Z. Li, "Energy-aware task scheduling with deadline constraint in dvfs-enabled heterogeneous clusters," *arXiv preprint arXiv:2104.00486*, 2021.
 - [23] A. Muralidharan and Y. Mostofi, "Communication-aware robotics: Exploiting motion for communication," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 115–139, 2021.
 - [24] S. A. Mohamed, M.-H. Haghbayan, A. Miele, O. Mutlu, and J. Plosila, "Energy-efficient mobile robot control via run-time monitoring of environmental complexity and computing workload," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7587–7593.

-
- [25] S. Shahidinejad, E. Bibeau, and S. Filizadeh, “Statistical development of a duty cycle for plug-in vehicles in a north american urban setting using fleet information,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 8, pp. 3710–3719, 2010.
 - [26] Q. Yan, B. Zhang, and M. Kezunovic, “Optimization of electric vehicle movement for efficient energy consumption,” in *2014 North American Power Symposium (NAPS)*. IEEE, 2014, pp. 1–6.
 - [27] D. Angioletti, F. Bertani, C. Bolchini, F. Cerizzi, and A. Miele, “A runtime resource management policy for opencl workloads on heterogeneous multicores,” in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1385–1390.
 - [28] Y.-J. Chen, B.-G. Jhong, and M.-Y. Chen, “A real-time path planning algorithm based on the markov decision process in a dynamic environment for wheeled mobile robots,” in *Actuators*, vol. 12, no. 4. MDPI, 2023, p. 166.
 - [29] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
 - [30] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, “Geometric a-star algorithm: An improved a-star algorithm for agv path planning in a port environment,” *IEEE access*, vol. 9, pp. 59 196–59 210, 2021.
 - [31] D. Li, W. Yin, W. E. Wong, M. Jian, and M. Chau, “Quality-oriented hybrid path planning based on a* and q-learning for unmanned aerial vehicle,” *Ieee Access*, vol. 10, pp. 7664–7674, 2021.
 - [32] W. Chi, Z. Ding, J. Wang, G. Chen, and L. Sun, “A generalized voronoi diagram-based efficient heuristic path planning method for rrts in mobile robots,” *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 4926–4937, 2021.
 - [33] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
 - [34] M. L. Puterman, “Markov decision processes,” *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.
 - [35] A. Rao and T. Jelvis, *Foundations of reinforcement learning with applications in finance*. Chapman and Hall/CRC, 2022.
 - [36] J. M. G. B. Feijão, “Deep q-learning with pca and prioritized experience replay for trading in the forex market,” 2018.
 - [37] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into deep learning*. Cambridge University Press, 2023.
 - [38] B. d. V. D. Aroso, “Using machine learning to improve the performance of flying networks,” Master’s thesis, Universidade do Porto (Portugal), 2019.

-
- [39] M. Han, X. Zhang, L. Xu, R. May, S. Pan, and J. Wu, “A review of reinforcement learning methodologies on control systems for building energy,” 2018.
 - [40] Y. Tang and S. Agrawal, “Discretizing continuous action space for on-policy optimization,” in *Proceedings of the aaai conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 5981–5988.
 - [41] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” *arXiv preprint arXiv:1506.02438*, 2015.
 - [42] C. Harris, M. Stephens *et al.*, “A combined corner and edge detector,” in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.
 - [43] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.
 - [44] J. Redmon, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.