Institute for Machine Tools and Industrial Management (*iwb*)
TUM School of Engineering and Design
Technical University of Munich

Development and Evaluation of a Reinforcement Learning Framework for CAM Parameter Identification

Entwicklung und Evaluierung eines Reinforcement Learning Frameworks für die Identifikation von CAM-Parametern

Scientific work for obtaining the academic degree

Bachelor of Science (B.Sc.)

at the TUM School of Engineering and Design of the Technical University of Munich

Examined by Prof. Dr.-Ing. Michael F. Zäh
Chair of Machine Tools and Manufacturing Technology

Submitted by Cornelius Gruss
Matriculation No.: 03738649

Submitted on 05.05.2025 in Garching

Institute for Machine Tools and Industrial Management (*iwb*)
TUM School of Engineering and Design
Technical University of Munich

Task Description

English Title of the Bachelor's Thesis:

Development and Evaluation of a Reinforcement Learning Framework for CAM Parameter Identification

German Title of the Bachelor's Thesis:

Entwicklung und Evaluierung eines Reinforcement Learning Frameworks für die Identifikation von CAM-Parametern

Written by: Cornelius Gruss
Matriculation-No.: 03738649
Advised by: M. Sc. Moritz Goeldner
Time period: 04.11.2024 to 05.05.2025

Initial Situation

Reinforcement learning (RL), as an encouraging machine learning method, is becoming increasingly important in optimizing manufacturing processes and enhancing the product quality. Over the past few decades, the primary areas of application for reinforcement learning have spanned across different areas such as autonomous driving, financial investments, medical diagnostics, and intelligent control systems. (Gupta et al., 2021)

With the continuous rise in manufacturing costs and an increasing demand for automation, the application of RL in production and computer-aided manufacturing has become increasingly important.

Computer-aided manufacturing (CAM) is a computational manufacturing method that utilizes computer software to simulate the machining production process and program automated CNC machines to manufacture products with a high degree of complexity and accuracy. CAM systems often help to simplify the planning effort of multiaxial milling processes and offer an opportunity to investigate various process parameters. To effectively select the proper value of these parameters ensuring optimal cutting conditions, reinforcement learning is introduced and thereby regulates the machining process. As the researchers Weiye Li et al. (2022), Lu et al. (2023), Samsonov et al. (2020) and Kaneko et al. (2023) pointed out, reinforcement learning proves applicable to various production domains and mechanical machining processes.

Objective

The primary objective of this thesis is to utilize reinforcement learning to reduce the dependence of CAM planning on expert knowledge and to achieve an automated CAM planning process. Within this thesis, the process parameters such as cutting depth and cutting speed are varied within the framework of reinforcement learning in order to optimize e.g. the processing time and cutting forces. This serves as a theoretical foundation for subsequent research endeavors.

Approach and Working Methodology

The following approach is planned to achieve the desired objectives:

- Induction into machining and the associated process parameters
- Systematic preparation of literature review findings.
- Investigation of the theoretical operation of CAM systems, investigating key indicators which offer potential for optimization.
- Exploration and study of the suitability of reinforcement learning methods for the machining process.
- Evaluation of the suitability of existing reinforcement learning models for the research problem of optimizing parameters for CAM simulation
- Identification of appropriate parameters and relevant metrics to formulate the cost function and the state-action space for the training environment.
- Development of an appropriate interface to ensure the implementation of reinforcement learning within a CAM-system.
- Integration of a suitable reinforcement learning model within a simulated environment.
- Prototypical training of the reinforcement learning framework

Agreement

With the supervision of Mr. Cornelius Gruss, the intellectual property of the *iwb* is incorporated into this work. Publication of the work or disclosure to third parties requires the permission of the chair holder. I agree to the archiving of the thesis in the *iwb*‘s library, which is only accessible to *iwb* employees, and in the *iwb*‘s digital thesis database as a PDF document.

Garching, den 04.11.2024

Prof. Dr.-Ing. Michael F. Zäh Cornelius Gruss

Abstract

This thesis addresses the challenge of parameter identification in Computer-aided Manufacturing (CAM) systems, which traditionally relies on expert knowledge rather than systematic approaches. Manufacturing efficiency and product quality depend critically on appropriate parameter selection for machining operations. To address this challenge, this research develops and evaluates a reinforcement learning (RL) framework for automated parameter identification in milling operations, integrated with the professional CAM software *hyper*MILL. The framework employs a comprehensive approach that includes state space representation, action space definition, and a reward function balancing machining time and accumulated work. Two state-of-the-art RL algorithms, Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC), are implemented and compared using a sequence of six predefined machining operations. Results demonstrate that SAC outperforms PPO in terms of convergence speed (24% faster), decision stability (standard deviation of 0.77 versus 7.46), and job completion rate (100% versus 95.20%). The framework successfully identifies parameter combinations that meet specified objectives, with SAC developing more nuanced, operation-specific parameter selections compared to PPO, which tends toward generalization. Comparison with industry benchmark parameters highlights the potential of RL approaches for systematizing parameter identification while revealing limitations in the current implementation, particularly regarding surface quality considerations. This research establishes a methodological foundation for applying RL to CAM parameter identification that can serve as a baseline for future work in automated manufacturing planning.

Zusammenfassung

Diese Bachelorarbeit untersucht die Herausforderungen der Parameteridentifikation in Computer-Aided Manufacturing (CAM)-Systemen, die traditionell eher auf Expertenwissen als auf systematischen Methoden basiert. Die Effizienz der Fertigung und die Produktqualität hängen maßgeblich von der geeigneten Wahl der Bearbeitungsparameter ab. Um dieser Herausforderung zu bewältigen, wird in dieser Arbeit ein Framework auf Basis von Reinforcement Learning (RL) zur automatisierten Parameteridentifikation bei Fräsprozessen entwickelt und evaluiert, das in die professionelle CAM-Software *hyper*MILL integriert ist. Das Framework umfasst die Modellierung des Zustandsraums, die Definition möglicher Aktionen und eine Belohnungsfunktion, die eine Balance zwischen Bearbeitungszeit und aufgewendeter Arbeit herstellt. Zwei moderne RL-Algorithmen, Proximal Policy Optimization (PPO) und Soft Actor-Critic (SAC), werden implementiert und anhand einer festen Abfolge von sechs vordefinierten Fräsoperationen verglichen. Die Ergebnisse zeigen, dass SAC den PPO-Algorithmus hinsichtlich der Konvergenzgeschwindigkeit (24% schneller), der Entscheidungsstabilität (Standardabweichung von 0,77 gegenüber 7,46) und der Erfolgsrate (100% gegenüber 95,2%) übertrifft. Das Framework identifiziert erfolgreich Parameterkombinationen für die definierten Ziele, wobei SAC differenziertere, operationsspezifische Parametereinstellungen entwickelt, während PPO eher zu Generalisierungen tendiert. Der Vergleich mit industriellen Standardparametern verdeutlicht das Potenzial von RL-Ansätzen zur Systematisierung der Parameterbestimmung und zeigt gleichzeitig Einschränkungen der aktuellen Implementierung auf, insbesondere bezüglich der Berücksichtigung der Oberflächenqualität. Diese Forschungsarbeit schafft eine methodische Grundlage für die Anwendung von RL zur CAM-Parameterbestimmung, die als Ausgangspunkt für zukünftige Arbeiten im Bereich der automatisierten Fertigungsplanung dienen kann.

Table of Content

List of Figures

List of Tables

List of Equations

List of Abbreviations

Abbreviation	Description
API	Application Programming Interface
BP	Backpropagation
CAD	Computer-aided Design
CAM	Computer-aided Manufacturing
CNC	Computer Numerical Control
DDQN	Double Deep Q-Network
DQN	Deep Q-Network
Iwb	Institute for Machine Tools and Industrial Management
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SAC	Soft Actor-Critic
TD3	Twin Delayed Deep Deterministic Policy Gradient

List of Indices

Latin Symbols

Symbol	Unit	Description
t	-	Discrete time step
s_t	-	State at timestep t
a_t	-	Action taken at timestep t
r_t	-	Reward received at timestep t
S	-	Set of all possible states in the Markov Decision Process
A	-	Set of all possible actions available to the agent
P	-	State transition function
$P(s_{t+1}\|s_t, a_t)$	-	Probability of transitioning to state s_{t+1} given state s_t and action a_t
R	-	Reward function
$R(s_t, a_t, s_{t+1})$	-	Reward expected when going to s_{t+1} from s_{t+1} via action a_t
$v_\pi(s)$	-	Value of state s under the policy π
$q_\pi(s, a)$	-	Value of taking action a in state s under the policy π
F_t	N	Tangential force component
F_f	N	Feed force component
F_r	N	Radial force component
K_{tc}	N/mm^2	Tangential cutting force coefficient
K_{fc}	N/mm^2	Feed cutting force coefficient
K_{rc}	N/mm^2	Radial cutting force coefficient
K_{te}	N/mm	Tangential edge force coefficient
K_{fe}	N/mm	Feed edge force coefficient
K_{re}	N/mm	Radial edge force coefficient
b	mm	Axial depth of cut
h	mm	Chip thickness
f_z	$mm/tooth$	Feed per tooth
$W_{segment}$	J	Work performed during a single cutting segment
d	mm	Distance travelled during a single cutting segment
$a_{i,t}$	-	i-th component of action vector at timestep t, where $a_{i,t} \in [-1,1]$ and $i \in [1,2,3,4,5]$
p_i	varies	Actual machining parameter after denormalization
$p_{min,i}$	varies	Minimum allowable value for a specific parameter
$p_{max,i}$	varies	Maximum allowable value of a specific parameter
R_{time}	-	Normalized time reward component, where $R_{time} \in [-3,3]$
R_{work}	-	Normalized work reward component, where $R_{work} \in [-3,3]$
$R_{performance}$	-	Performance reward component
$R_{progression}$	-	Non-linear incentive job progression reward
$R_{completion}$	-	Reward bonus for completing all jobs
$\boldsymbol{R_{total}}$	-	Total Episode Reward

Greek Symbols

Symbol	Unit	Description
γ	-	Discount Factor for future rewards
π	-	Policy (decision-making rule)
$\pi(s\|a)$	-	Probability of selecting action a in state s
π_*	-	Optimal policy
ϕ	rad	Engagement angle of cutting tool
α	-	Weighting factor for balancing the time and work objectives

Chapter 1 Introduction

1.1 Background and Motivation

Manufacturing has undergone several transformative phases throughout history, evolving from manual craftsmanship to mechanized production, and eventually to the digitally-controlled processes that define modern industrial production. This industrial evolution has been marked by technological innovations that increasingly automate and enhance manufacturing processes, with computer-aided manufacturing (CAM) representing a crucial development in this progression. CAM emerged in the 1960s as a revolutionary approach to industrial production (Kalpakjian & Schmid, 2022), enabling the translation of design specifications into precise machining instructions that control computerized-machine tools. Since its widespread adoption in the following decades, CAM has become an indispensable component in manufacturing workflows, particularly for complex parts requiring high precision (Rekow, 2006).

Despite the technological sophistication of CAM systems, the effectiveness of machining processes remains heavily dependent on the selection of appropriate process parameters. These parameters - including cutting depth (how much material is removed in a single pass), feed rate (how fast the tool moves through the material) and stepover width (the distance between adjacent cutting paths) - directly influence production efficiency, part quality, tool wear, and manufacturing costs (Ma et al., 2020). Currently, this critical parameter selection process relies substantially on expert knowledge accumulated through years of practical experience. According to Li et al. (2018), most process planning knowledge still remains in the minds of experienced engineers or in engineering handbooks, with manufacturing experts traditionally solving process-planning problems using their accumulated expertise and intuition.

This dependency on expert knowledge creates several challenges for modern manufacturing operations. First, it introduces bottlenecks in production planning, as parameter decisions must often pass through a limited number of qualified personnel (Leo Kumar, 2019). Second, it creates inconsistency in process planning, as Leo Kumar (2019) notes that different typically operators select different parameters for similar tasks based on their individual experiences and preferences. Third, and perhaps most concerning for the industry's future, it poses a significant knowledge transfer problem as experienced operators retire without systematic methods to capture and transfer their expertise (Leo Kumar, 2019).

Three converging trends in the manufacturing landscape further amplify the importance of addressing these challenges. First, there is an increasing demand for manufacturing automation across all process stages, driven by competitive pressures to reduce costs and increase productivity (Y. Lu et al., 2020). Second, the industry faces an increasing shortage of experienced CAM operators and process planners, creating a knowledge gap that threatens manufacturing capabilities (Jarosz et al., 2023). Third, there is the emergence and maturation of machine learning techniques that can potentially address complex manufacturing challenges in parameter optimization and decision-making processes (Lu et al., 2020).

Reinforcement learning (RL), a paradigm within machine learning, offers a promising approach to the challenge of parameter selection in CAM systems. Unlike supervised learning which requires labeled examples, RL learns through interaction with an environment, making it well-suited for process optimization problems where the optimal solution emerges from a sequence of decisions. RL agents learn by receiving rewards or penalties based on the outcomes of their actions, gradually improving their decision-making strategies through experience which is conceptually similar to how human operators develop expertise.

The application of RL to CAM parameter selection is particularly appropriate because it can potentially systematize and automate a process that has historically relied on individual expertise. By formulating parameter selection as a RL problem, it becomes possible to create systems that can identify optimal parameters based on specific objectives such as minimizing production time, maximizing surface quality, or extending tool life. Such systems could not only reduce dependency on human expertise but also potentially discover parameter combinations that human experts might overlook due to the vast parameter space involved in complex machining operations.

1.2 Problem Statement

This thesis addresses the lack of systematic approaches for parameter identification in CAM systems. Despite advances in CAM software and machine tool capabilities, determining optimal machining parameters remains largely unsystematic, relying on the operators experience rather than algorithmic approaches. This can create a significant gap between the potential capabilities of modern manufacturing equipment and the realized performance in practice.

Parameter selection in machining processes presents a challenging optimization problem due to several key factors:

1. The parameter space is complex and multidimensional (cutting depth, cutting speed, stepover width, feed rate, tool path strategy, etc.).
2. Parameters interact in non-linear ways, making their combined effects difficult to predict.
3. Optimization objectives often compete as an increase in cutting speed may reduce production time but also increase tool wear and potentially decrease surface quality.

The complexity increases further when considering workpiece materials, cutting tool characteristics, and machine-specific capabilities, each requiring specific parameter adaptations for optimal performance.

From this problem space, the following research questions emerge:

1. How can the parameter selection process in CAM systems be systematized to reduce dependency on implicit expert knowledge?
2. To what extent can machine learning approaches address the multi-objective nature of machining parameter optimization?
3. What evaluation metrics and validation approaches can effectively assess the quality of automatically generated parameter sets?

This thesis focuses specifically on milling operations, allowing for a deeper exploration of the challenges particular to this widely used machining process. By addressing the research questions within this defined scope, this work aims to establish fundamental approaches that could potentially extend to other machining processes in future research.

1.3 Thesis Objectives

The primary objective of this thesis is to develop and evaluate a RL framework for automated parameter identification in CAM systems, with a specific focus on milling operations.

To ensure industrial relevance and practical applicability, this framework is integrated with *hyper*MILL, a professional CAM software system developed by OPEN MIND Technologies AG and widely used in manufacturing industries. *hyper*MILL provides industrial-grade capabilities for toolpath generation and cutting data calculation, offering a realistic environment for developing and testing RL approaches. By creating a bridge between advanced machine learning techniques and established CAM software, this research aims to address parameter selection challenges within the actual tools used by industry professionals.

To achieve this primary objective, the thesis pursues the following specific goals:

1. Design appropriate state and action representations for the milling parameter identification problem that capture relevant process characteristics while remaining computationally manageable.
2. Develop reward functions that effectively guide the RL agent toward optimal parameter selections, balancing competing manufacturing objectives.
3. Implement and compare multiple RL algorithms within this framework to determine their relative strengths and limitations for this application domain.
4. Establish evaluation methodologies and metrics that meaningfully assess the performance of the parameter selection approach compared to conventional methods.

Through these objectives, the thesis aims to establish a methodological foundation that can serve as a baseline for future research in automated CAM planning.

1.4 Thesis Structure

This thesis is organized into six chapters that systematically address the research problem, present the developed methodology, and evaluate the results.

Chapter 1 introduces the research context, defines the problem statement, and outlines the research objectives.

Chapter 2 provides the fundamental background on CAM systems, machining processes (particularly milling operations), and RL concepts.

Chapter 3 presents a review of the state of the art in applying RL to machining processes, categorizing existing approaches and identifying research gaps.

Chapter 4 details the methodology developed in this thesis, including system architecture, environment design (state/action spaces and reward functions), and training strategy.

Chapter 5 presents the results, comparing different RL algorithms, analyzing the training process, and evaluating the framework.

Chapter 6 discusses the implications of the research findings, summarizes contributions, acknowledges limitations, and suggests directions for future work.

Chapter 2 Fundamentals

2.1 Computer-Aided Manufacturing (CAM)

2.1.1 Principles and Workflow

Computer-aided manufacturing (CAM) refers to the use of software systems that help transform digital designs into physical products by generating instructions for automated machine tools. While Computer-aided design (CAD) software allows engineers the creation of detailed 3D models, CAM software determines how machine tools should operate to produce the physical component from raw material.

The typical CAM workflow consists of several sequential steps as illustrated in Figure 2.1:

1. **CAD Model Import**: A 3D model created in CAD software is imported into the CAM system, establishing the target geometry for manufacturing.

2. **Stock Definition**: The programmer specifies the initial state of the raw material (the "stock"), including its dimensions and material properties.

3. **Operation Selection**: Appropriate machining operations are selected based on the geometric features to be created (various operation types are discussed in detail in Section 2.2).

4. **Parameter Selection**: For each operation, the programmer must define numerous process parameters. This critical step directly influences part quality, production efficiency, and tool life.

5. **Toolpath Generation**: The CAM software calculates precise toolpaths based on the selected operations and parameters, determining the exact movements the cutting tool will follow.

6. **G-code Generation**: Toolpaths are converted into machine-specific instructions in G-code format, a standardized numerical control programming language that specifies tool coordinates, movement speeds, spindle rotation rates, and other machine functions.

As shown in Figure 2.1, the process flow includes an important repetition through steps 3-5 for each operation necessary to manufacture a part. Modern components typically require multiple machining operations, each demanding its own operation selection, parameter configuration, and toolpath generation, based on its specific geometric requirements, preceding operations, and quality objectives.

This multi-operation nature of CAM programming significantly increases the complexity of the parameter selection task. Parameters must not only be optimized for individual operations but also coordinated across the entire manufacturing sequence to ensure consistency, efficiency, and quality.

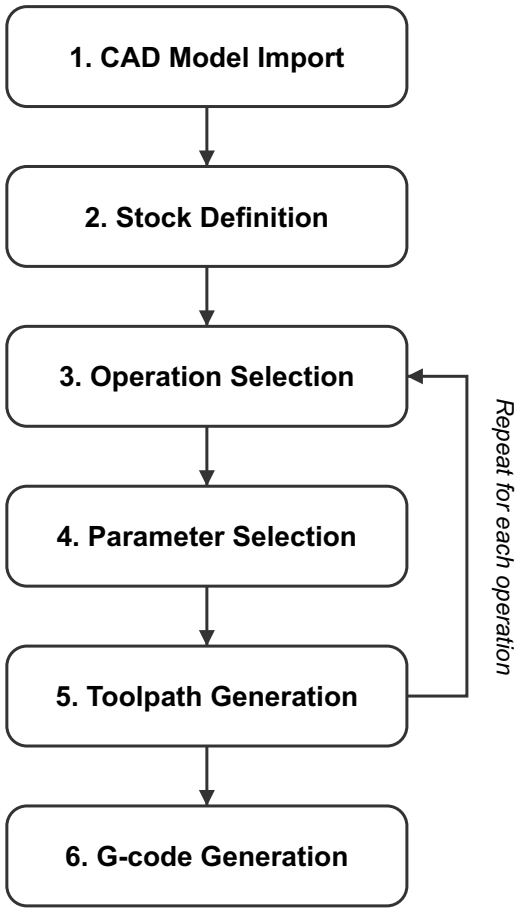


Figure 2.1: CAM workflow showing the sequential process from CAD model import to G-code generation

2.1.2 Current Challenges

Despite significant advances in CAM technology, several persistent challenges impact manufacturing efficiency and quality outcomes:

Parameter selection complexity: Modern CAM systems require configuring numerous interdependent parameters for each machining operation. A typical milling operation may involve selecting appropriate values for cutting speeds, feed rates, tool engagement, cutting depths, entry/exit strategies, and toolpath patterns. The parameter space increases dramatically with each

additional parameter, making manual optimization increasingly difficult. For example, if each parameter has just 10 possible values, a five-parameter optimization problem already involves 10^5 possible combinations. There exist few standardized methodologies for parameter selection, with manufacturers often relying on conservative defaults that prioritize reliability over performance.

Knowledge dependency: Successful parameter selection typically depends on the expertise of individual CAM programmers, knowledge acquired through years of experience. An experienced programmer might instinctively know that certain combinations of speed and feed work better for specific materials, but this knowledge is difficult to formalize, document, or transfer between operators. This situation creates vulnerability in manufacturing operations when experienced staff leave, potentially resulting in capability gaps and reduced manufacturing efficency.

Multi-objective optimization challenges: Manufacturing operations involve inherent trade-offs between competing objectives. For example, increasing material removal rates through higher cutting speeds and feed rates typically reduces machining time but may negatively impact surface quality, dimensional accuracy, tool life, and energy consumption. CAM systems provide limited support for systematically balancing these competing objectives, requiring operators to make subjective judgments about acceptable compromises rather than employing quantitative optimization approaches.

Context sensitivity: Optimal parameter values exhibit high sensitivity to the specific manufacturing context, including workpiece material properties, cutting tool characteristics, machine capabilities, and the geometric features being machined. For instance, parameters that work effectively when milling aluminum may cause catastrophic failure in titanium, even with identical tools and machines. Similarly, parameters suitable for open-faced operations often require adjustment when machining confined spaces or deep cavities due to chip evacuation and tool deflection concerns. This context dependency necessitates continuous adaptation of parameters even within a single part program, further complicating the selection process.

Limited algorithmic support: While modern CAM systems incorporate sophisticated algorithms for collision detection, toolpath optimization, and remaining material calculation, they provide comparatively limited algorithmic support for parameter selection. Most systems offer basic parameter suggestions based on material-tool pairing tables rather than dynamic optimization that accounts for the specific machining context and operational constraints.

These challenges highlight the need for more systematic approaches to parameter identification in CAM systems. The following section examines milling processes in detail, as they represent the specific manufacturing context in which this thesis will develop a reinforcement learning framework for parameter identification.

2.2 Machining Processes

2.2.1 Types of Machining Operations

Machining represents a fundamental manufacturing technique that transforms raw materials into finished components through controlled material removal processes. Kalpakjian and Schmid (2022) describe machining as encompassing various processes that remove material and modify workpiece surfaces after initial forming. This subtractive approach enables precise dimensional control, high-quality surface finishes, and complex geometrical features that would be difficult or impossible to achieve through primary forming processes alone.

While various machining operations exist, including turning, drilling, and grinding, this thesis focuses specifically on milling operations. Milling involves using rotary cutters to remove material from a workpiece, with the cutting tool typically rotating around a fixed axis while the workpiece or tool moves along multiple axes to create the desired geometry. According to Kalpakjian and Schmid (2022), milling operations can be categorized into three primary types based on the tool orientation and cutting strategy:

Peripheral milling (also called plain milling) involves a cutting tool whose axis of rotation is parallel to the workpiece surface. The cutter body has multiple teeth along its circumference, with each tooth acting like a single-point cutting tool. When the cutter is longer than the width of the cut, the operation is referred to as slab milling. Peripheral milling cutters may have either straight or helical teeth, with helical teeth generally preferred as they engage with the workpiece gradually, resulting in smoother cutting action and reduced chatter.

Face milling removes material from flat surfaces, primarily using the cutting edges on the face (bottom) of the tool. The cutter diameter is typically relativily large in comparison to the width of the workpiece, allowing for efficient material removal across wide surfaces in fewer passes. Face milling tools have cutting inserts arranged around the outer edge of the face of the tool body.

End milling employs cylindrical cutters (end mills) with cutting edges on both the end face and periphery. End mills can remove material with both their end faces and cylindrical cutting edges, enabling the creation of a wide range of features including slots, pockets, profiles, and three-dimensional contours. Kalpakjian and Schmid (2022) identify several specialized end mill types, including ball-nose mills with hemispherical tips for sculptured surfaces.

For each milling operation type, programmers must distinguish between roughing and finishing strategies, which represent fundamentally different approaches with distinct parameter requirements. Roughing operations prioritize material removal efficiency, rapidly reducing the workpiece to near-final dimensions while leaving a small amount of excess material called machining allowance. This excess material, controlled through different parameters, serves as a buffer that protects the final geometry from potential irregularities during the high-material-removal roughing phase.

In contrast, finishing operations prioritize dimensional accuracy and surface quality, removing the remaining material to achieve the final specifications with minimal deviation. Finishing typically employs lower cutting depths, higher spindle speeds, and optimized tool paths to achieve the required surface characteristics. The transition between roughing and finishing operations represents a critical decision point in the manufacturing process, requiring careful parameter selection to balance productivity with quality objectives.

2.2.2 Process Parameters and Optimization

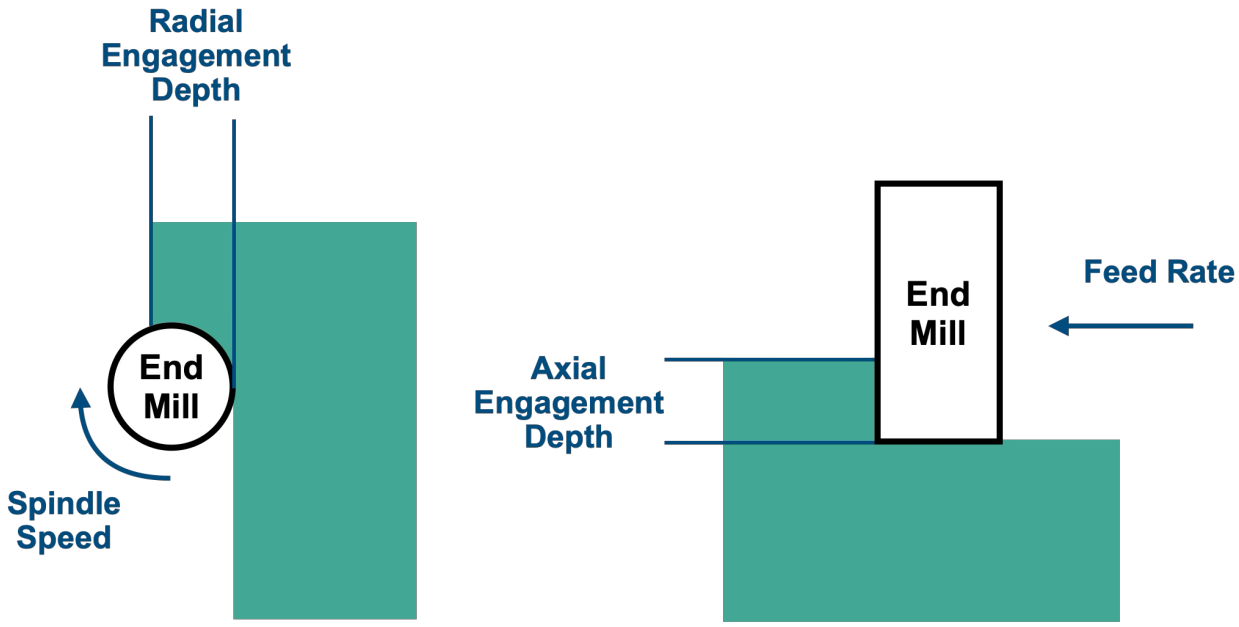


Figure 2.2: Illustriation of key milling process parameters from the Top and Side view.

The performance of milling operations depends significantly on the selection of process parameters, which directly influence productivity, surface quality, tool life, and energy consumption. Essential parameters are described below:

Feed rate represents the speed at which the cutting tool advances through the material, typically measured in millimeters per minute (mm/min).

Spindle speed defines the rotational velocity of the cutting tool, measured in revolutions per minute (RPM).

Axial engagement depth (also called cutting depth) represents the distance the tool penetrates into the material along its axial direction in a single pass.

Radial engagement depth (also called stepover) determines the lateral distance the tool engages with the material during cutting operations, often expressed as a percentage of the tool diameter.

Machining Allowance parameters establish deliberate dimensional buffer between roughing and finishing operations.

Parameter interdependencies create complex optimization challenges in machining operations. For instance, increasing cutting depth typically requires reducing feed rate to maintain acceptable cutting forces. Similarly, higher spindle speeds may allow faster feed rates but can generate excessive heat that affects both tool life and workpiece quality. These interdependencies create a complex multi-dimensional optimization space where parameters cannot be considered in isolation.

The optimization of milling parameters represents a challenging multi-objective problem, requiring balance between competing goals:

- Maximizing material removal rate to reduce cycle time
- Minimizing tool wear to extend tool life and reduce replacement costs
- Achieving desired surface quality specifications
- Reducing energy consumption and environmental impact

These objectives often conflict, for example, increasing feed rate typically improves productivity but may reduce surface quality and accelerate tool wear. This complex trade-off space, combined with the high dimensionality of parameter selection and process uncertainty, creates an opportunity for advanced approaches like RL that can systematically explore parameter combinations and learn from outcomes across multiple objectives.

2.3 Reinforcement Learning

2.3.1 Basic Concepts

Reinforcement Learning (RL) represents a machine learning approach focused on how agents should take actions within an environment to maximize cumulative reward. Unlike supervised learning, which requires labeled training data, RL learns optimal behaviors through trial-and-error interactions with the environment. This approach makes RL particularly suitable for complex sequential decision-making problems where optimal solutions are difficult to specify in advance, such as parameter identification in manufacturing processes.

The core framework of RL consists of several key components. An agent interacts with an environment over a series of discrete time steps. At each time step t, the agent observes the current state s_t of the environment, selects an action a_t based on its policy, and receives a reward r_t along with a transition to a new state s_{t+1}. This interaction cycle, illustrated in Figure 2.3, forms the foundation of the RL process.

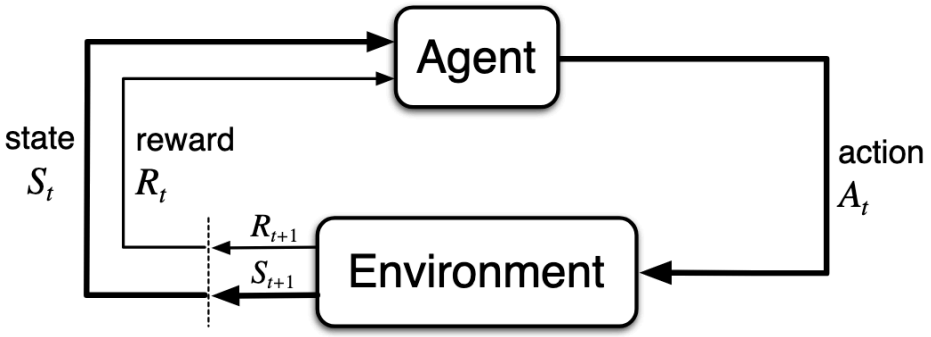


Figure 2.3: The standard RL framework showing the agent-environment interaction loop (Sutton & Barto, 2020, p.48)

Formally, RL problems are typically modeled as Markov Decision Processes (MDPs). The term "Markov" refers to the property that the future depends only on the current state and not on the sequence of events that preceded it. An MDP consists of a state space *S*, an action space *A*, a transistion function $P(s_{t+1}|s_t, a_t)$ that defines the probability of moving to state s_{t+1} after taking action a in state s, and a reward function $R(s_t, a_t, s_{t+1})$. A discount factor $\gamma \in [0,1]$ determines how much the agent values future rewards compared to immediate ones.

The primary goal in RL is to learn a policy π where $\pi(s|a)$ defines the probability of selecting action a in state s. The agent aims to find an optimal policy π_* that maximizes the expected cumulative reward.

To achieve this goal, RL algorithms often estimate value functions that predict the expected cumulative reward from a given state (state-value function $v_\pi(s)$) or from taking a specific action in a given state (action-value function $q_\pi(s, a)$). These value functions gudie the agent's decision-making process: when an agent knows the value of each state or state-action pair, it can make informed choices by selecting actions that lead to states with higher values or directly choosing the action with the highest action-value.

A central challenge in RL is the balance between exploration (trying new actions to discover potentially better strategies) and exploitation (selecting actions known to yield high rewards based on current knowledge). This trade-off is critical for effective learning, as too much exploitation may lead to suboptimal solutions, while excessive exploration can be inefficient.

2.3.2 Algorithms and Approaches

Reinforcement learning algorithms can be categorized into three main approaches:

Value-based methods focus on learning value functions and then deriving policies from them. Q-learning, a fundamental algorithm in this category, learns the optimal action-value function without requiring a model of the environment. Deep Q-Network (DQN) extends this approach by using neural networks to approximate the Q-function, enabling RL to handle high-dimensional state spaces.

Policy-based methods directly optimize the policy without necessarily learning a value function. These methods are particularly effective for continuous action spaces and can learn stochastic policies. Policy gradient methods update the policy parameters in the direction of greater expected return.

Model-based methods explicitly learn a model of the environment's dynamics and use this model for planning. These methods can be data-efficient but may suffer from model errors if the learned model doesn't accurately represent the actual environment.

Two advanced policy-based algorithms that will be considered in this thesis are:

Proximal Policy Optimization (PPO) addresses the challenge of making meaningful policy improvements without causing destructive updates that destabilize the training process. It achieves this by limiting how much the policy can change in a single update step, implementing a form of "trust region" that prevents drastic changes. This conservative update approach provides stability that helps prevent the selection of potentially harmful actions as we could have in manufacturing contexts.

Soft Actor-Critic (SAC) is an off-policy actor-critic algorithm that incorporates entropy maximization to encourage exploration while learning from all collected experiences. SAC combines value function estimation with direct policy optimization. It is sample-efficient and stable for complex environments with continuous action spaces, making it particularly suitable for CAM parameter optimization where interactions with the environment may be computationally expensive.

RL methods can also be classified based on how they interact with the environment:

- **Online learning** involves learning from real-time interactions with the environment.
- **Offline learning** (also called batch RL) learns from fixed datasets of previously collected experiences without new environment interaction.
- **Simulation-based learning** trains agents in virtual environments that model the real-world system dynamic.

The application of RL to complex domains like manufacturing parameter optimization faces several common challenges, including sparse rewards, high-dimensional continuous action spaces, safety concerns during exploration, sample efficiency when interactions are costly, and generalization across different conditions.

For CAM parameter identification, PPO and SAC are selected for implementation based on their demonstrated effectiveness in continuous control problems. PPO offers stability during training that helps avoid extreme parameter selections, while SAC's entropy-based exploration and off-policy learning provide sample efficiency important for CAM integration where simulations are computationally expensive.

Having established the fundamental concepts of CAM and RL, the next chapter examines how these approaches have been combined in existing research. Chapter 3 will review the current state of the art in applying RL to machining processes, identify the progress made in this emerging field, and highlight the research gaps that this thesis aims to address through its novel framework for CAM parameter identification.

Chapter 3 State of the Art

3.1 Reinforcement Learning Approaches in Machining

Building on the RL concepts established in Chapter 2, this chapter examines how researchers have applied RL specifically to machining parameter optimization. The literature reveals two primary application contexts: pre-process parameter planning (determining optimal parameters before manufacturing begins) and in-process control (adjusting parameters during operation). This thesis focuses on the former, which presents unique challenges in translating theoretical RL frameworks to practical CAM implementations.

Within these application contexts, researchers have implemented the three training approaches introduced in Chapter 2, online learning, offline learning, and simulation-based learning, each with distinct advantages when applied to manufacturing optimization:

Online learning implementations have demonstrated the ability to adapt directly to physical manufacturing environments. Zhang et al. (2022) implemented the Rainbow algorithm, a model-free deep reinforcement learning approach combining several DQN improvements, for constant force control in robotic grinding operations (notably not milling, but demonstrating parameter control principles). Their implementation achieved improved surface quality with a 26.54% reduction in surface roughness compared to empirically adjusted parameters through real-time parameter adjustments. Jiang et al. (2022) developed a Time-Series Deep Q-Network (TS-DQN) approach for contour error compensation in milling operations, which extends traditional DQN by incorporating temporal dependencies through recurrent neural networks. This modification enabled the system to achieve 99% precision in tracking error prediction and 60-85% reduction in contour error. Gulde et al. (2019) applied RL for vibration compensation during milling processes, autonomously adjusting parameters to maintain stability. Despite their demonstrated effectiveness, these online learning implementations face significant limitations including safety risks during exploration phases, potential for equipment damage, high operational costs, and inefficient use of production equipment for training purposes.

Offline learning implementations leverage historical manufacturing data to train RL agents without physical equipment interaction. Wang et al. (2022) developed a Double Deep Q-Network (DDQN) approach for milling operations, trained on experimental data collected using the Taguchi method. Their two-step pipeline used DDQN first to optimize Support Vector Regression parameters for surface roughness prediction (DDQN-SVR), then applied DDQN again to optimize four key machining parameters (spindle speed, feed rate, width of cut, and depth of cut). This approach achieved 91.18% accuracy in predicting surface quality while finding optimal trade-offs between surface roughness and material removal rate. Their work demonstrated how manufacturers could utilize process data for parameter optimization without experimental disruptions. However, offline learning methods remain inherently limited by the quality, comprehensiveness, and distribution of available historical data, potentially struggling to generalize to machining conditions not well-represented in the training dataset.

Simulation-based implementations include Li et al. (2022), who combined back-propagation (BP) neural networks with the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm (creating a hybrid approach called BP-TD3) to optimize machining parameters for energy efficiency and machining cost during turning operations. Their simulation environment used a neural network to model machine tool power consumption, avoiding the complexity of empirical formulas. This approach achieved results comparable to traditional particle swarm optimization while reducing optimization time by 95%. Xiao et al. (2021) applied model-agnostic meta-learning across 24 different simulated milling configurations, achieving an 85.90% reduction in optimization time. Kaneko et al. (2023) implemented a DQN-based system for optimizing feed rate in end milling operations, successfully controlling cutting torque across different scenarios. While these simulation-based approaches avoid the safety and cost concerns of online learning, they face challenges in bridging the reality gap between simulated and physical machining environments, particularly in accurately modeling complex phenomena like chatter, tool deflection, and material inconsistencies.

The diversity of RL algorithms applied to machining parameter optimzation reflects the various approaches researchers have taken to address manufacturing challenges. Having examined the different learning contexts, the next section will explore the specifc algorithms that have been implemented for milling parameter optimization.

3.2 Algorithm Selection in Milling Parameter Optimization

The choice of RL algorithms for milling parameter optimization depends largely on whether the control parameters are treated as discrete or continous variables. This section examines how researchers have selected different algorithm types to address the unique requirements of manufacturing environments.

Value-based methods have been effectively applied when milling parameters are represented as discrete values. In their offline learning approach, Wang et al. (2022) selected Double Deep Q-Network (DDQN) to handle the surface quality and material removal rate optimization. DDQN's architecture helps produce more stable and accurate value estimates than standard DQN, which proved beneficial for balancing competing manufacturing objectives. Similarly, Kaneko et al. (2023) utilized DQN for feed rate optimization in end milling, demonstrating how value-based methods can successfully maintain target torque levels across different cutting scenarios without requiring continuous torque monitoring.

Lu et al. (2023) selected Soft Actor-Critic (SAC) for their system that simultaneously optimized energy efficiency and workpiece deformation. Their approach leveraged SAC's ability to balance exploration of new parameter combinations with exploitation of known effective parameters, a characteristic that aligns well with manufacturing optimization where finding global optima among many local optima is challenging. Similarly targeting continuous control problems, Li et al. (2022) adopted the TD3 algorithm, enhanced with a simulation-based learning environment as detailed earlier. Their approach, BP-TD3, illustrates how predictive neural models can be integrated into the RL pipeline to support stable and efficient optimization, particularly when direct interaction with physical systems is costly or impractical.

Beyond selecting algorithms for individual optimization tasks, researchers have begun exploring approaches that transfer knowledge between different manufacturing contexts. Xiao et al. (2021) implemented model-agnostic meta-learning to develop parameter optimization systems that can quickly adapt to new machining configurations without extensive retraining. This approach addresses the practical manufacturing requirement of flexibility across different part geometries, materials, and tooling.

The pattern emerging from the literature shows that algorithm selection is guided primarily by the nature of the parameter space: discrete parameter representations favor value-based methods, while continuous parameter spaces benefit from actor-critic approaches. Within this latter category, researchers have also begun to integrate predictive neural models into the learning loop, as seen in simulation-based methods like BP-TD3, to enable stable policy learning when direct physical interaction is infeasible. Researchers have also explored algorithms that can effectively transfer knowledge across different manufacturing tasks.

Table 3.1 summarizes the key characteristics of the RL applications for milling parameter optimization discussed in this chapter, including their learning approaches, algorithms, optimization objectives, and results. This synthesis provides a foundation for identifying the remaining research gaps in the field.

Table 3.1: Key characteristics of recent RL applications for milling parameter optimization

Authors (Year)	Learning Approach	Algorithm	Parameters Optimized	Optimization Objectives	Key Results
Zhang et al. (2022)	Online	Rainbow	CNC controller parameters	Surface quality, Controller robustness	26.54% reduction in surface roughness
Jiang et al. (2022)	Online	TS-DQN	CNC machining parameters (compensation cycle, compensation rate)	Tracking accuracy, Contour precision	99% precision in tracking error prediction, 60-85% reduction in contour error
Gulde et al. (2019)	Online	Various RL Algorithms	Feed-drive velocity commands	Process stability, Vibration compensation	Achieved vibration suppression without dynamics model
Wang et al. (2022)	Offline	DDQN	Cutting speed, Feed rate, Depth of cut, Width of cut	Surface roughness, Material removal rate	91.18% accuracy in surface quality prediction

Authors (Year)	Learning Approach	Algorithm	Parameters Optimized	Optimization Objectives	Key Results
Li et al. (2022)	Simulation	BP-TD3	Spindle speed, Feed speed	Energy efficiency, Machining cost	95% reduction in optimization calculation time
Xiao et al. (2021)	Simulation	Meta-learning	Feed Rate, Cutting depth, Turning Speed	Generalization across machining configurations, Energy and cost-efficient productivity	Outperformed heuristic methods; fast adaptation and convergence across machining tasks
Kaneko et al. (2023)	Simulation	DQN	Feed rate	Cutting torque stability	Maintained target torque across varying conditions
Lu et al. (2023)	Simulation	SAC	Feed rate, spindle speed, cutting width	Energy efficiency, Workpiece deformation	45.71% improvement in specific cutting energy, 32.27% enhancement in material removal rate

3.3 Research Gaps and Thesis Focus

The preceding analysis of current literature and the synthesis presented in Table 3.1 reveal several critical gaps in the application of RL to machining parameter optimization. This section identifies these gaps and establishes the specific focus areas where this thesis aims to make contributions, thereby positioning the current work within the broader research landscape.

Integration with Professional CAM Systems: Current research primarily uses custom simulation environments or simplified models of the machining process, with a notable lack of integration between RL approaches and professional CAM software systems that are commonly used in industry. This integration gap limits the practical applicability of research findings, as parameter optimizations that work in simplified environments may not transfer effectively to the complex, feature-rich environments of professional CAM systems. This thesis addresses this gap by developing a framework that integrates directly with *hyper*MILL, providing access to industrial-grade toolpath generation and cutting data calculation. This integration ensures that the RL

framework as outlined in Section 1.3 operates within the actual constraints and capabilities of production CAM systems rather than simplified approximations.

Sequential Operation Optimization: Existing approaches typically focus on optimizing specific aspects of single machining operations, treating each operation in isolation. Although Xiao et al. (2021) made progress in adaptability across different machining configurations, their approach does not address the interdependencies between sequential machining operations. This limitation becomes particularly apparent in complex parts requiring multiple operations, where the surface condition or material state resulting from one operation can significantly influence the optimal parameters for subsequent steps. This thesis explores the potential for identifying parameters across sequential operations, considering the interdependencies that exist in real manufacturing workflows.

Comprehensive Parameter Space Exploration: Research implementations often focus on a limited subset of machining parameters, typically addressing feed rate and spindle speed while giving less attention to parameters like depth of cut, stepover width, and tool path strategies. As shown in Table 3.1, most studies consider only three to four parameters , whereas professional CAM systems offer significantly more configurable parameters for a single milling operation. The full parameter space in modern CAM systems is far more extensive, with dozens of interrelated parameters affecting process outcomes.

By developing a RL framework that operates within an industrial-grade CAM system, considers sequential operations and explores a more comprehensive parameter space, this work aims to bridge the gap between theoretical RL approaches and practical industrial applications. The framework developed in this thesis represents a step toward more automated, efficient, and knowledge-driven manufacturing processes, where parameter selection leverages the power of RL while respecting the constraints and requirements of real manufacturing environments.

Having identified these research gaps, the following chapter presents the methodology developed to address them. Chapter 4 details the system architecture, environment design, and training strategy of the proposed RL framework.

Chapter 4 Methodology

4.1 Framework Design

This section presents the technical architecture and workflow of the RL framework for CAM parameter identifcation. The design choices address the research gaps identified in Chapter 3.

4.1.1 System Architecture

The RL framework is directly integrated with *hyper*MILL through its Python Application Programming Interface (API). This integration approach creates a bridge between industrial-standard manufacturing software and advanced machine learning techniques, leveraging the strengths of both domains. Rather than developing a custom simulation environment from scratch, the framework utilizes *hyper*MILL's built-in capabilities for toolpath generation and cutting data calculation, ensuring industrial relevance and practical applicability.

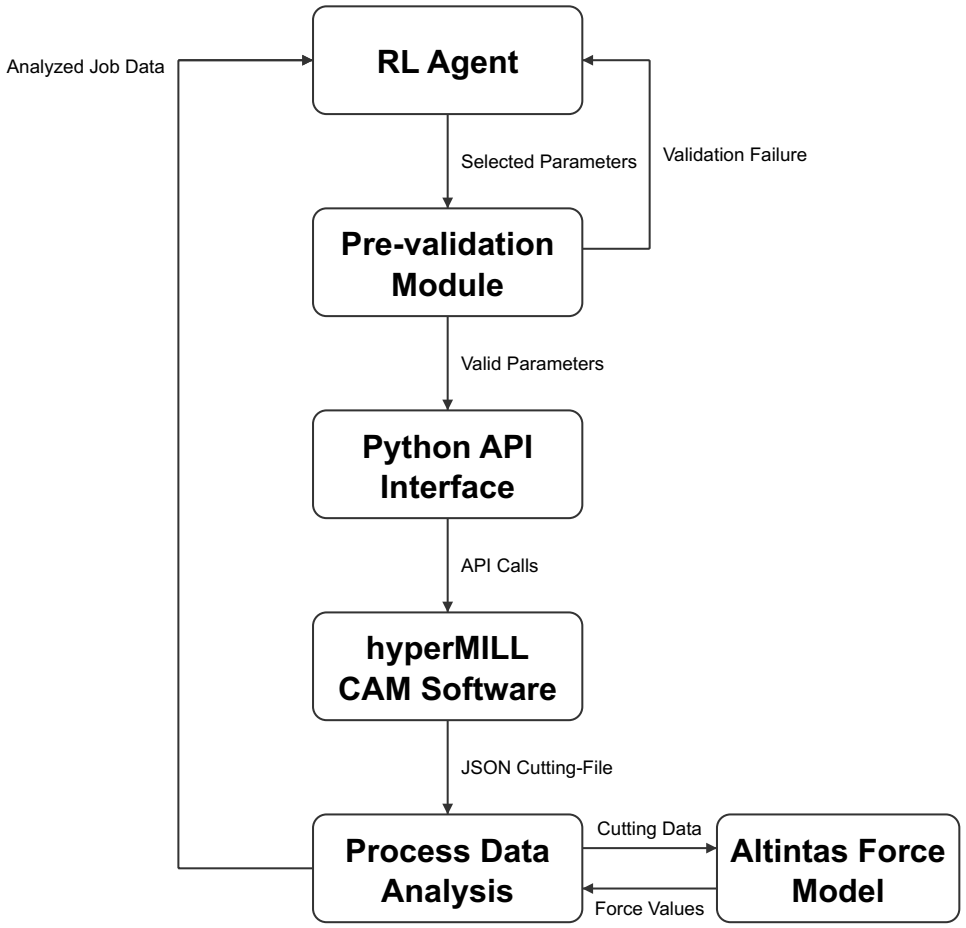


Figure 4.1: System architecture and information flow between the framework components.

The system architecture consists of the following key modules:

RL Agent: The central component that makes decisions regarding machining parameters based on the current state and learned policy. The agent is implemented using the Stable-Baselines3 (SB3) library (Raffin et al., 2021), which supports various algorithms. It operates according to the standard RL paradigm of state observation, action selection, and reward processing. The detailed design of this component, including state and action spaces, is further elaborated in the environment design section (Section 4.2).

Pre-validation Module: Checks the validity of selected parameters before executing the full job simulation, significantly reducing unnecessary computations for invalid parameter combinations. In *hyper*MILL terminology, a "job" refers to a discrete machining operation (such as face milling or contour milling). This module performs quick validation checks such as ensuring selected tools meet the minimum requirements for the current job.

Python API Interface: Establishes the connection between the RL framework and *hyper*MILL through the software's Python API. This interface enables the framework to modify parameters programmatically and execute job calculations.

CAM Software (*hyper*MILL): Provides the industrial-grade toolpath generation and simulation capabilities. When a job is executed with a given set of parameters, *hyper*MILL calculates the toolpath based on the part geometry, tool selection, and machining parameters, then generates a detailed JSON cutting-file containing segment-wise information.

Process Data Analysis: After *hyper*MILL executes a job simulation, this module extracts and processes the JSON cutting data to calculate machining time based on path length and feed rates, and evaluate the overall machining performance. It then calls the force model to calculate process forces based on the cutting trajectory data.

Force Model: The force model developed by Altintas (2012) is implemented to calculate process forces based on the cutting data extracted from *hyper*MILL. These force calculations provide physical validation of the selected machining parameters and are used in the reward function to evaluate different parameter choices. The detailed mathematical formulation and implementation of this force model including the segment-wise integration are elaborated in Section 4.1.4.

4.1.2 Framework Workflow and Data Exchange

The interaction between the components follows a defined sequence for each machining job:

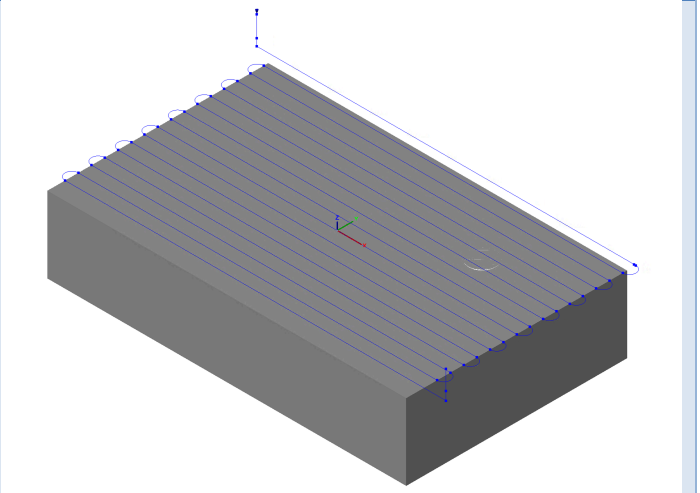
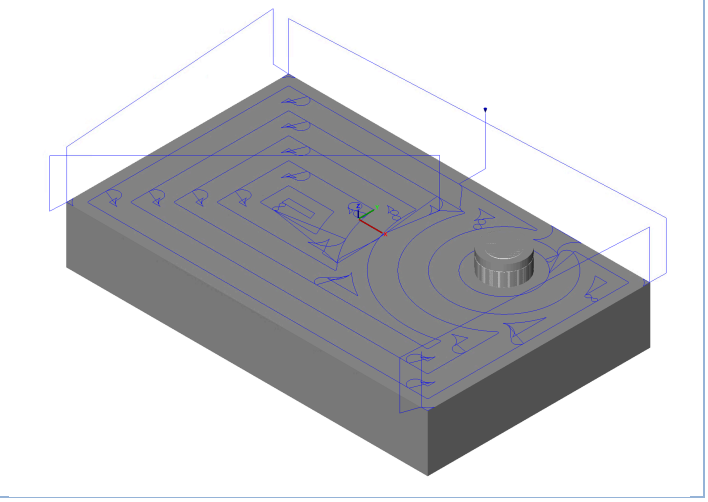
1. The RL agent observes the current state, which includes information about the machining job requirements and available tools.
2. Based on its learned policy, the agent selects machining parameters (tool, feed rate, step factor, etc.).
3. These parameter selections are sent to the pre-validation module.
4. The pre-validation module checks the parameter selection against basic constraints.
5. If the parameters pass pre-validation, they are passed to the Python API interface.
6. The Python API interface applies the valid parameters to the current job in *hyper*MILL.
7. *hyper*MILL simulates the job, generates the toolpath and produces a JSON cutting-file.

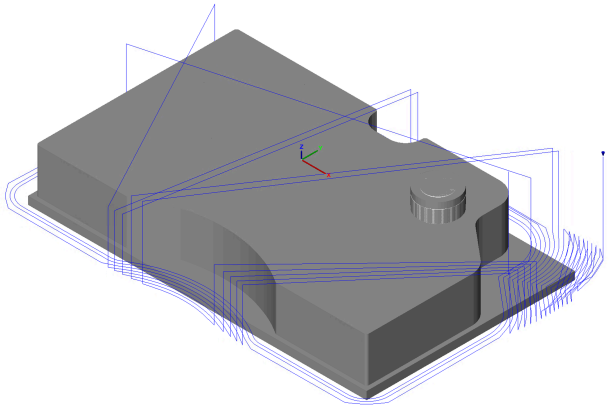
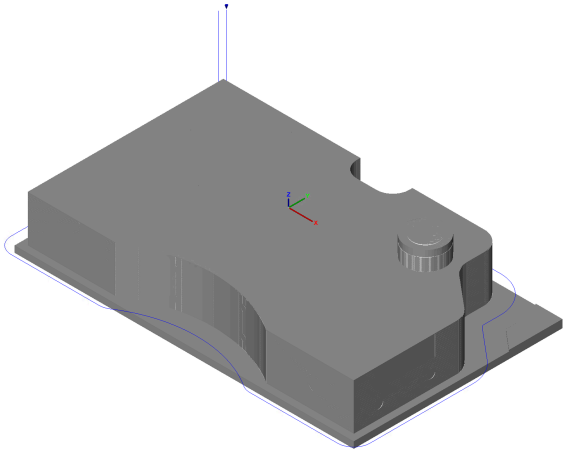
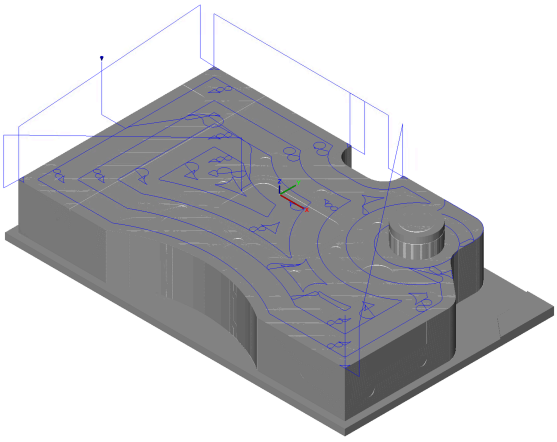
8. The process data module analyzes the cutting data to calculate machining time and process forces.
9. The calculated results of the job are given back to the agent who then calculates the reward.
10. The process repeats for the next job or the next training episode.

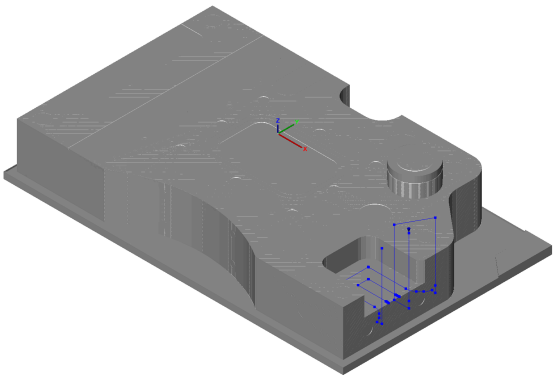
4.1.3 Machining Test Bed Configuration

The framework was evaluated using a comprehensive test bed consisting of six sequential machining jobs on a single workpiece. This configuration presents diverse manufacturing challenges with varying geometric constraints. The sequence progresses from initial stock reduction to final feature creation, testing the framework's ability to adapt parameters across different operation types and to account for the effects of previous machining states on subsequent operations. For this prototypical approach, the test bed intentionally focuses on representative machining jobs rather than a complete part production sequence.

Table 4.1: Sequential Machining Jobs in the Test Bed

Job ID	Job Type	Description	Visual Representation
1	Face Milling (Roughing)	Initial stock material removal that establishes the primary reference surface.	
2	Pocket Milling (Roughing)	Material removal from the top surface leaving over a boss on the right side.	

Job ID	Job Type	Description	Visual Representation
3	Contour Milling (Roughing)	Rough cutting of the external profile.	
4	Contour Milling (Finishing)	Finishing passes for the external profile to achieve final dimensions.	
5	Pocket Milling (Finishing)	Finishing passes on the top surface to get the final dimensional accuracy.	

Job ID	Job Type	Description	Visual Representation
6	Pocket Milling (Roughing and Finishing)	Single-operation milling of smaller pocket.	

4.1.4 Force Calculation Model

A critical component of the framework is the force calculation model based on the cutting force approach developed by Altintas (2012). This model is essential for evaluating the physical implications of the selected machining parameters and serves as a key input to the reward function.

The force model implementation utilizes a segment-wise approach, analyzing each cutting segment in the toolpath generated by *hyper*MILL. For each segment where material is removed, the framework performs the following operations:

1. Extracts cutting parameters (cutting depth, feed rate, engagement angle)
2. Calculates force components using the Altintas force model
3. Computes the work performed during that segment
4. Accumulates the total work across the entire toolpath

The Altintas force model used in this implementation combines both cutting and edge coefficients in the force calculations:

$$\begin{aligned} \boldsymbol{F_t} &= \boldsymbol{K_{tc} \cdot b \cdot h + K_{te} \cdot b} \\ \boldsymbol{F_f} &= \boldsymbol{K_{fc} \cdot b \cdot h + K_{fe} \cdot b} \\ \boldsymbol{F_r} &= \boldsymbol{K_{rc} \cdot b \cdot h + K_{re} \cdot b} \end{aligned} \tag{4.1}$$

Where:

- F_t, F_f, and F_r are the tangential, feed, and radial force components (N)
- K_{tc}, K_{fc}, and K_{rc} are the cutting force coefficients (N/mm^2)
- K_{te}, K_{fe}, and K_{re} are the edge force coefficients (N/mm)
- b is the axial depth of cut (mm)
- h is the chip thickness (mm)

The chip thickness h is a critical parameter that varies with the tool rotation angle and determines the instantaneous cutting forces. It is calculated as:

$$\boldsymbol{h = f_z \cdot sin(\phi)} \tag{4.2}$$

Where:

- f_z is the feed per tooth ($mm/tooth$), derived from the feed rate and spindle speed
- ϕ is the engagement angle, representing the angular position of the cutting edge

The framework implements the force calculation using standard coefficient values for aluminum machining, applying these consistently across all tools to maintain computational efficiency. For varying engagement conditions, the framework performs discrete integration over the engagement angles with 20 integration steps. The integration approach accommodates different milling strategies:

Conventional Milling: When the right side of the tool is engaged, forces are integrated from angle $\phi = 0$ to $\phi = \phi_{eng}$, where ϕ_{eng} is the engagement angle.

Climb Milling: When the left side of the tool is engaged, forces are integrated from angle $\phi = 2\pi - \phi_{eng}$ to $\phi = 2\pi$, with a negative sign applied to the sine function in the chip thickness calculation to account for the opposite cutting direction.

Combined Engagement: When both sides of the tool are engaged simultaneously, the framework calculates forces for both engagement regions separately and then averages the results to obtain the final force components.

The segment-wise work calculation follows:

$$\boldsymbol{W_{segment} = \frac{F_t \cdot d}{1000}} \tag{4.3}$$

Where:

- $W_{segment}$ is the work performed in a single segment (Joules)
- F_t is the tangential force (N)
- d is the distance traveled by the tool along that segment (mm)
- Division by 1000 converts from Nmm to Nm

The accumulated work across the entire toolpath provides a comprehensive metric for energy consumption and tool wear:

$$\boldsymbol{W_{total} = \sum_{i=1}^{n} W_{segment,i}} \tag{4.4}$$

Where n is the number of cutting segments in the toolpath.

The current framework implementation focuses on parameter identification for a specific part with six fixed machining operations. This specialized approach provides a controlled environment for evaluating the capabilities of RL for parameter identification. While this limits immediate generalization to other parts, it establishes a methodological foundation that can be extended to more diverse manufacturing scenarios in future work.

4.2 Environment Design

The environment follows a standard MDP as described in Section 2.3, with states, actions, transitions, rewards, and termination conditions specifically adapted for the CAM Parameter identification problem. The interaction flow between the RL agent and the machining environment follows the system architecture illustrated in Figure 4.1, where parameter selections from the agent are validated, executed through the CAM software, and the resulting performance is analyzed to provide feedback to the agent.

The environment dynamics include:

- Transitions from state s_t to s_{t+1} determined by current job status, parameter validity and job completion
- Termination conditions that include selection of inappropriate tools (early termination) or successful completion of all six machining jobs
- Detailed state and action representations that enable the agent to make informed decisions
- A comprehensive reward function that balances multiple objectives

The following sections provide a detailed description of the key components of this environment design.

4.2.1 State Space Representation

The state space has been designed to provide the RL agent with sufficient information about the machining context while maintaining computational efficiency. The state representation comprises three primary components:

Job Properties provide essential information about the current machining operation:

- Job Identifier: A unique ID for each of the six machining operations.
- Operation Type: A binary indicator (0 for roughing, 1 for finishing) denoting the operation category.
- Minimum Radius Constraint: The smallest feature radius (mm) that must be accommodated in the current job.
- Maximum Cutting Depth: The available depth (mm) for material removal, which updates dynamically based on the previous operations' outcomes. For instance, the z-axis machining allowance selected in job 2 directly affects the available cutting depth for subsequent jobs 3 and 4.

Tool Information represents the complete set of available tools in the library. The state space contains information about all 11 tools simultaneously, enabling the agent to select the most appropriate tool for each operation. For each tool in the library, the following attributes are included:

- Tool Identifier: A unique ID for each tool in the library
- Cutting Length: The effective cutting portion of the tool (mm)
- Tool Diameter: The diameter of the cutting tool (mm)
- Tool Type Classification: An enumerated value representing the tool category (end mill, face mill, etc.)
- Validity Indicator: A binary value (0 for invalid, 1 for valid) indicating whether the tool meets the requirements for the current job

Last Cutting Tool Used caputes the tool ID from the previous operation, allowing the agent to consider tool change implications when making new parameter selections.

The complete tool library consists of standard end mills and face milling cutters with varying geometric properties as detailed in Table 4.2. By including the validity indicator for each tool, the state representation provides an efficient pre-validation mechanism that helps the agent focus on selecting compatible tools for specific operations.

Table 4.2: Tool library available to the Agent, showing 11 tools with different diameters and cutting lengths

Tool ID	Name	Diameter (mm)	Cutting Length (mm)
1	D8 Schaftfräser	8	19
2	D12 Schaftfräser	12	30
3	D16 Schaftfräser	16	38
4	D25 Schaftfräser	25	50
5	D4 Schaftfräser	4	11
6	D20 Schaftfräser	20	38
7	D40 Stahl Eckmesserkopf	40	11
8	D100 Stahl Eckmesserkopf	100	7
9	D32 Alu Eckmesserkopf	32	16
10	D2 Schaftfräser	2	6
11	D20 Schaftfräser	20	60

4.2.2 Action Space Definition

The action space is designed as a continuous space with normalized dimensions. This action vector $a_t \in R^5$ consists of five components, with each dimension $a_i \in [-1,1]$ normalized to this range. These normalized values are then mapped to specific parameter ranges relevant to the machining process as detailed in Table 4.3.

This normalization approach standardizes diverse parameter scales, improving neural network training efficiency and exploration strategies while allowing the RL algorithm to operate in a consistent space regardless of the underlying physical parameter ranges.

Table 4.3: Action Space Parameters and Ranges

Parameter	Description	Range	Unit
Tool ID	Selection of cutting tool	$1,2,\dots,11$	-
Feed rate XY	Cutting feed rate in XY plane	$[40,200]$	mm/min
Step factor	Radial engagement width	$[0.30,1]$	-
Cutting depth	Axial depth of cut	$[10,51]$	mm
Z-axis machining allowance	Machining left on workpiece in Z direction	$[0.50,4]$	mm

The mapping from normalized action values to machining parameters follows:

$$\boldsymbol{p_i = p_{min,i} + \frac{(a_{i,t}+1)}{2} \times \left(p_{max,i} - p_{min,i}\right)} \tag{4.5}$$

Where p_i is the actual parameter value, $p_{min,i}$ and $p_{max,i}$ are the minimum and maximum allowable values for that parameter, and a_i is the normalized action space component within the range $[-1,1]$.

This continuous action space formulation allows the agent to select precise parameter values within the allowable ranges, providing flexibility in optimization while respecting physical constraints of the machining process.

4.2.3 Reward Function Design

The design of this reward function acknowledges that parameter optimization in manufacturing represents a multi-objective problem with inherent trade-offs. Rather than attempting to incorporate all possible manufacturing considerations, the function strategically focuses on two fundamental dimensions that capture the essence of machining performance and provide clear learning signals:

Machining Time serves as a direct measure of process efficiency and productivity, quantified in seconds. This metric captures the economic dimension of machining operations, where shorter cycle times typically translate to higher throughput and lower production costs. By including machining time in the reward function, the framework encourages the agent to discover parameter combinations that maximize productivity within the constraints of other requirements.

Work Required provides a physics-based measure of process quality and tool interaction, calculated in Joules using the Altintas Force Model described in Section 4.1.3. This metric embodies multiple quality considerations: (1) energy efficiency of the cutting process, (2) mechanical load on the cutting tool which correlates with tool wear and potential failure, and (3)

indirect indication of surface quality, as excessive forces often lead to poor surface finish. By optimizing for lower work values, the agent learns to select parameters that promote sustainable machining practices and longer tool life.

The reward function integrates several components to provide comprehensive feedback to the agent:

Performance Reward Component normalizes performance metrics against predetermined bounds:

$$\boldsymbol{R_{performance} = \alpha \cdot R_{time} + (1 - \alpha) \cdot R_{work}} \quad (4.6)$$

Where:

- α is a weighting factor to balance time and work objectives (set to 0.5)
- R_{time} is the normalized time reward component
- R_{work} is the normalized work reward component

The normalization scales each metric to a range of $[-3,3]$:

$$\boldsymbol{R_{metric} = -3 + 6.0 \cdot \frac{metric - metric_{best}}{metric_{worst} - metric_{best}}} \quad (4.7)$$

Where $metric_{best}$ and $metric_{worst}$ represent the predetermined best and worst values for each job, as specified in Table 4.4.

Table 4.4: Job-specific performance metric bounds

Job ID	Time Min (min)	Time Max (min)	Work Min (J)	Work Max (J)
1	0.4	220	18	800
2	4	220	720	9000
3	4	220	1350	9000
4	1	100	450	4000
5	4	165	126	2000
6	0.4	16	36	1000

Progression Reward provides a non-linear incentive based on job progression:

$$\boldsymbol{R_{progression} = 0.5 \cdot (j^{1.2})} \quad (4.8)$$

Where j is the current job identifier (1-6). This provides a gradually increasing reward as the agent progresses through more complex jobs.

Episode Completion Bonus $\boldsymbol{R_{completion}}$ of +6.0 is awarded for successfully completing all six jobs, encouraging complete machining sequences.

Validity Handling terminates episodes when invalid parameters are selected, particularly inappropriate tools, providing clear feedback without explicit penalties

This reward structure balances immediate performance feedback with longer-term learning signals, guiding the agent to develop effective parameter identification policies.

The final reward signal combines these components:

$$R_{total} = R_{performance} + R_{progression} + R_{completion} \tag{4.9}$$

As shown in Table 4.4, the maximum achievable reward is approximately TODO for successfully completing all six machining operations with excellent parameter selections. This includes the episode completion bonus of 6.0, which is added upon successful completion of the final job. The reward structure creates a progression that increasingly incentivizes the agent to advance through all operations while maintaining high-quality parameter selections throughout the process.

Job ID	Performance Reward (Min)	Performance Reward (Max)	Progression Reward	Cumulative Reward (Min)	Cumulative Reward (Max)
1	-3	3	0.50	-2.50	3.50
2	-3	3	1.15	-4.35	7.65
3	-3	3	1.87	-5.48	12.52
4	-3	3	2.64	-5.84	18.16
5	-3	3	3.45	-5.39	24.61
6	-3	3	4.30 (+6)	1.90	37.90

Table 4.5: Potential reward breakdown by job for different performance levels

4.3 Training Strategy

The training strategy addresses the challenge of applying RL to parameter identification within the constrains of industrial CAM software integration, balancing learning efficency with practical constraints inherent to manufacturing software integration.

4.3.1 Implementation Approach

The framework implements the two RL algorithms PPO and SAC using the Stable-Baselines3 library. As discussed in Section 2.3.2, these algorithms were selected for their strengths in handling continuous parameter spaces like those found in machining processes, with PPO offering stability and SAC providing efficient experience utilization while balancing exploration and exploitation.

Several key hyperparameters influence the learning behaviour of the RL algorithms:

Learning rate determines how quickly the model adjusts its parameters during training. Higher values enable faster learning but may cause instability, while lower values provide more stable but slower learning.

Batch size represents the number of training samples used for each update of the model parameters. This affects both the stability and computational efficiency of the training process.

Entropy coefficient controls how much the algorithm encourages exploration of new parameter combinations. Higher values promote more diverse exploration, while lower values focus on exploiting known good parameters. Even with a coefficient of 0.0, agents still maintain some exploration capability through their underlying policy mechanisms. SAC can automatically adjust this coefficient during training.

For the SAC algorithm specifically:

Learning starts specifies the number of initial interactions with the environment before the algorithm begins learning. This allows the agent to collect a diverse set of experiences before making policy updates.

Buffer size defines the capacity of the replay memory that stores previous experiences. This memory allows the agent to learn from past interactions, improving sample efficiency.

For the PPO algorithm:

N steps determines the number of environment steps collected before each policy update. This parameter affects the trade-off between data collection and learning frequency.

These hyperparameters were configured based on standard values recommended in the Stable-Baselines3 documentation, with adjustments made to achieve convergence. The specific values used in the training experiments are detailed in Table 5.1 in the Results chapter.

4.3.2 Training Process

The training infrastructure integrates the RL framework with *hyper*MILL CAM software through its Python API, which introduces several unique constraints compared to typical RL applications:

Sequential Processing: Due to the industrial CAM software's architecture, only a single instance of *hyper*MILL can be run at a time, preventing parallel training. Each training step requires approximately 12 seconds on average, with the majority of computation time spent on toolpath generation within the CAM software.

Training Duration: Each training run was configured to execute 10,000 environment interactions, resulting in approximately 20 hours of continuous training time per model. Training was conducted on a dedicated workstation where computing resources were sufficient for the RL algorithms, but the primary bottleneck remained the *hyper*MILL execution time rather than computational limitations.

Sequential Job Structure: Each episode consists of six sequential machining operations of various complexity. The agent must select appropriate parameters for each operation in sequence, with the episode terminating if invalid parameters are selected for any operation. This creates a challenging learning environment where early mistakes prevent the agent from experiencing later operations.

4.3.3 Monitoring and Evaluation

Training progress is monitored using TensorFlow's logging capabilities, a standard tool for tracking machine learning experiments, which captured key metrics including episode rewards, selected parameters, total machining time, total work performed. The primary indicators of successful training include increasing mean rewards, improving job completion rates and decreasing total machining time or work.

The training approach focuses on optimizing the complete manufacturing process rather than individual operations in isolation. This global optimization perspective recognizes that the optimal parameter set for the entire workpiece may involve trade-offs between individual operations. Some operations might use slightly sub-optimal parameters if this enables better overall performance across the complete machining sequence. This holistic approach more accurately reflects real manufacturing scenarios where total production time and resource utilization across all operations determine overall efficiency.

4.3.4 Validation Approach

To evaluate the effectiveness of the RL approach, the trained agent's parameter selections are compared against industry-standard benchmark parameters obtained from the Hoffmann Group tool catalog, which represents common industrial practice. This comparison focuses on key machining parameters including tool selection, feed rates, axial engagement depths, and resulting machining time and work metrics.

The validation methodology includes:

- Execution of the complete six-operation sequence using the industry-standard benchmark parameters
- Execution of the complete sequence using parameters selected by trained RL agent
- Quantitative comparison of performance metrics focusing on total machining time and process forces (work)
- Analysis of training progress and convergence behavior to evaluate the stability and reliability of the learned policies

This validation step measures the practical value of the framework against established methods and addresses whether RL can systematize the parameter selection process while maintaining or improving upon the performance achieved through conventional industrial guidelines. The results

also provide insights into which aspects of machining parameter selection benefit most from the RL approach versus traditional methods.

The comparative performance of the different algorithms, specific hyperparameter configurations, and their impact on training outcomes will be analyzed in detail in the next chapter.

Chapter 5 Results and Analysis

This chapter presents the results of implementing and evaluating the RL framework for CAM parameter identification developed in Chapter 4. The following sections detail the experimental outcomes, learning dynamics, and performance of the implemented algorithms. The analysis provides insights into the framework's capability to systematically identify machining parameters and demonstrates the integration of RL techniques within a professional CAM system environment.

5.1 Training Configuration

The implementation of RL for CAM parameter identification required careful selection of training configurations and hyperparameters to ensure effective learning while accommodating the constraints of industrial CAM software integration. To facilitate a fair comparison between algorithms, common basic configurations were established for both PPO and SAC implementations. The hyperparameters were initially based on default values recommended in the Stable-Baselines3 implementation, with practical adjustments made through preliminary testing to ensure both algorithms could successfully complete the machining tasks.

Table 5.1 summarizes the key training hyperparameters used for both algorithms. These configurations reflect a pragmatic approach to the parameter identification problem, balancing theoretical recommendations with the practical constraints of the *hyper*MILL environment, where each environment interaction requires significant computation time for toolpath generation and force calculation.

Table 5.1: Summary of Training Configurations

Parameter	SAC	PPO
Total Timesteps	10000	10000
Learning Rate	0.0004	0.0004
Batch Size	256	256
Gamma	0.99	0.99
Learning Starts	400	N/A
Buffer Size	10000	N/A
N Steps	N/A	512
Entropy Coefficient	Auto-adjusted	0.0

5.2 Training Performance and Algorithm Comparison

This section presents a comprehensive analysis of the training processes for both PPO and SAC algorithms, focusing on learning dynamics, convergence properties, and parameter selection behavior throughout the training process. The comparative analysis provides insights into the strengths and limitations of each algorithm when applied to the CAM parameter identification problem.

5.2.1 Learning Curves and Convergence Analysis

Figure 5.1 illustrates the learning progress of both algorithms across four key metrics: episode reward, job completion rate, total machining time, and accumulated work. These metrics provide complementary perspectives on how each algorithm learned to identify CAM parameters over time.

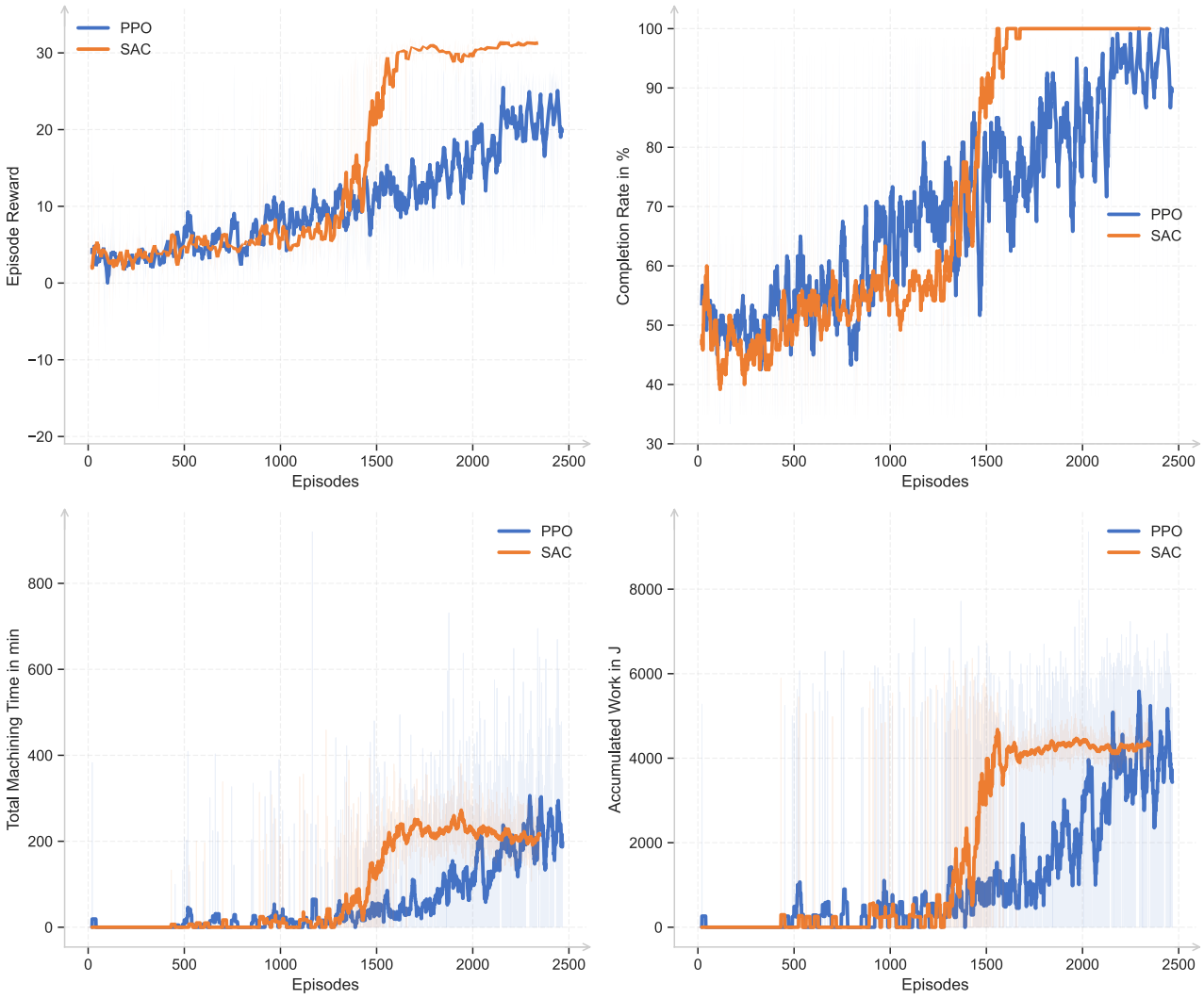


Figure 5.1: Learning curves showing episode reward, completion reward, machining time and accumulated work for the PPO and SAC algorithm across training episode

The episode reward curves (top-left) reveal distinctly different learning patterns between the two algorithms. Both PPO and SAC show similar reward levels during the initial 1000 episodes, indicating comparable exploration behaviors in the early stages of training. However, beyond episode 1000, SAC demonstrates a rapid increase in reward, achieving consistently high values around episode 1500. This suggests that SAC discovered effective parameter combinations that generalized well across all machining jobs at this point. In contrast, PPO shows a more gradual improvement throughout the entire training process, continuing to make incremental progress even in the later episodes, though never matching SAC's performance level.

This divergence in learning patterns aligns with the theoretical properties of these algorithms: SAC's off-policy learning approach enables more efficient use of past experiences, allowing it to more quickly identify and exploit effective parameter combinations once discovered. PPO's on-policy approach, while potentially more stable in some environments, appears to result in slower convergence for this specific parameter identification task.

The job completion rate (top-right) further highlights these differences. SAC achieves a consistent 100% completion rate (successfully completing all six machining operations) after approximately 1700 episodes. PPO, however, demonstrates higher variability in completion rate, fluctuating between 80-100% even in the later stages of training. This suggests that PPO continued to explore parameter combinations that occasionally resulted in invalid tool selections.

Examining the total machining time (bottom-left), both algorithms initially generate similar values as they explore the parameter space. Around episode 1400, the curves begin to diverge significantly. SAC settles into a relatively stable range of machining times, indicating it had identified consistent parameter combinations. In contrast, PPO exhibits greater variability and generally higher machining times toward the end of training, suggesting less stability in its parameter identification strategy.

The accumulated work metric (bottom-right) reveals perhaps the most interesting distinction. SAC shows a characteristic spike in accumulated work around episode 1500 before settling into a stable pattern. This coincides with the period when SAC's reward and completion rate reached their peak values, suggesting that SAC initially identified parameters that successfully completed all jobs but required refinement to optimize work values. PPO, meanwhile, shows a more gradual increase in accumulated work that continues throughout training, without achieving the same level of stability observed in SAC.

To quantitatively confirm the observations from the learning curve, key convergence metrics were calculated for both algorithms, as shown in Table 5.2.

Table 5.2: Convergence Metrics Comparison

Metric	PPO	SAC
Total Training Time (hh:mm:ss)	20:28:27	17:49:18
Total Training Episodes	2467	2348
Total Environment Steps	10239	10000
Episodes to Convergence	2207	1685
Final Mean Reward (last 100 eps)	21.63	31.19
Reward Stability (std last 100 eps)	7.46	0.77
Final Job Completion Rate (%)	95.20	100
Max Achieved Reward	30.62	33.16
Reward Consistency (% of max in last 100)	70.61	94.13

These metrics confirm the qualitative observations from the learning curves. SAC converged approximately 24% faster than PPO and achieved a higher final mean reward with substantially better stability. The order-of-magnitude difference in standard deviation (0.77 vs 7.46) particularly highlights SAC's consistent performance.

These learning curves demonstrate that both algorithms can learn effective parameter identification strategies, but with significant differences in efficiency and reliability. SAC more quickly discovers and exploits effective parameter combinations, achieving a perfect 100% completion rate in the final training phase compared to PPO's 95.20%. PPO continues exploring the parameter space more extensively throughout training, resulting in less consistent performance. This difference in convergence efficiency is particularly significant given the computational constraints of the CAM environment, where each episode represents a substantial time investment. The quantitative results clearly indicate that SAC's off-policy approach is better suited to this specific CAM parameter identification framework.

5.2.2 Parameter Selection Analysis

To better understand how each algorithm explored and converged on different machining parameters, Figure 5.2 illustrates the evolution of key parameters for Job 2 throughout the training process. The scattered points show the individual parameter selections made during each episode, while the solid lines are moving averages highlighting the underlying trend. Job 2 (contour milling roughing operation) serves as a representative example that demonstrates characteristic differences in parameter selection strategies between the algorithms.

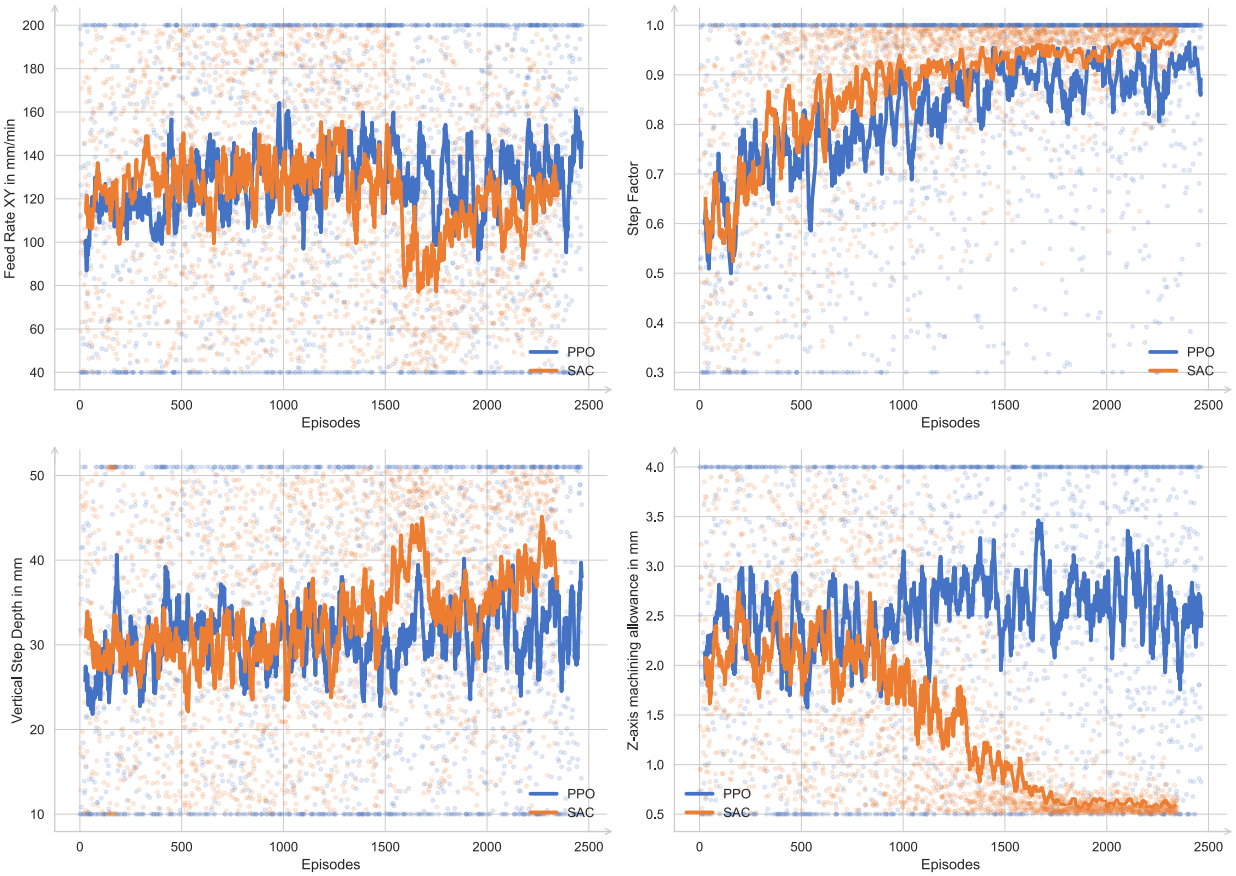


Figure 5.2: Parameter evolution for Job 2 for both PPO and SAC

The parameter evolution reveals notable differences in exploration and exploitation patterns between the algorithms. For feed rate XY (top-left), PPO frequently selects values at the extremes of the allowable range, exhibiting a characteristic bimodal distribution pattern with many selections at both the minimum and maximum values. In contrast, SAC shows a more continuous exploration pattern, gradually narrowing its focus to mid-range values around 120-140 mm/min.

The step factor parameter (top-right) shows convergence toward high values (0.80-1) for both algorithms, reflecting the discovery that higher step factors generally lead to more efficient machining removal for contour operations. However, PPO demonstrates greater variability throughout training, while SAC converges more consistently to near-maximum values.

For the vertical step depth parameter (bottom-left), both algorithms show ongoing fluctuations throughout training without clear convergence. Unlike the other parameters, neither algorithm appears to settle on a consistent strategy for vertical step depth. PPO again displays a tendency toward more extreme values, while SAC fluctuates more centrally, but both algorithms maintain similar levels of variation even in late training stages. This suggests that both algorithms struggled to identify consistent relationships between vertical step depth and overall performance.

The Z-axis machining allowance parameter (bottom-right) reveals the most dramatic difference between the algorithms. PPO fails to converge on any consistent strategy, fluctuating around an average of approximately 2.50 mm throughout the entire training process. In contrast, SAC exhibits a strong shift around episode 1000, progressively reducing this parameter near the

minimum value of 0.50 mm by the end of training. This demonstrates SAC's ability to discover and commit to a specific approach for tool positioning, identifying that minimal allowance values result in more efficient machining for this specific contour operation and subsequent jobs, while PPO continues to sample across a broader range without developing a clear preference.

5.2.3 Tool Selection Evolution During Training

Figure 5.3 illustrates the tool selection evolution over training episodes for both algorithms across all six machining jobs, revealing distinct patterns in how each algorithm approaches tool selection.

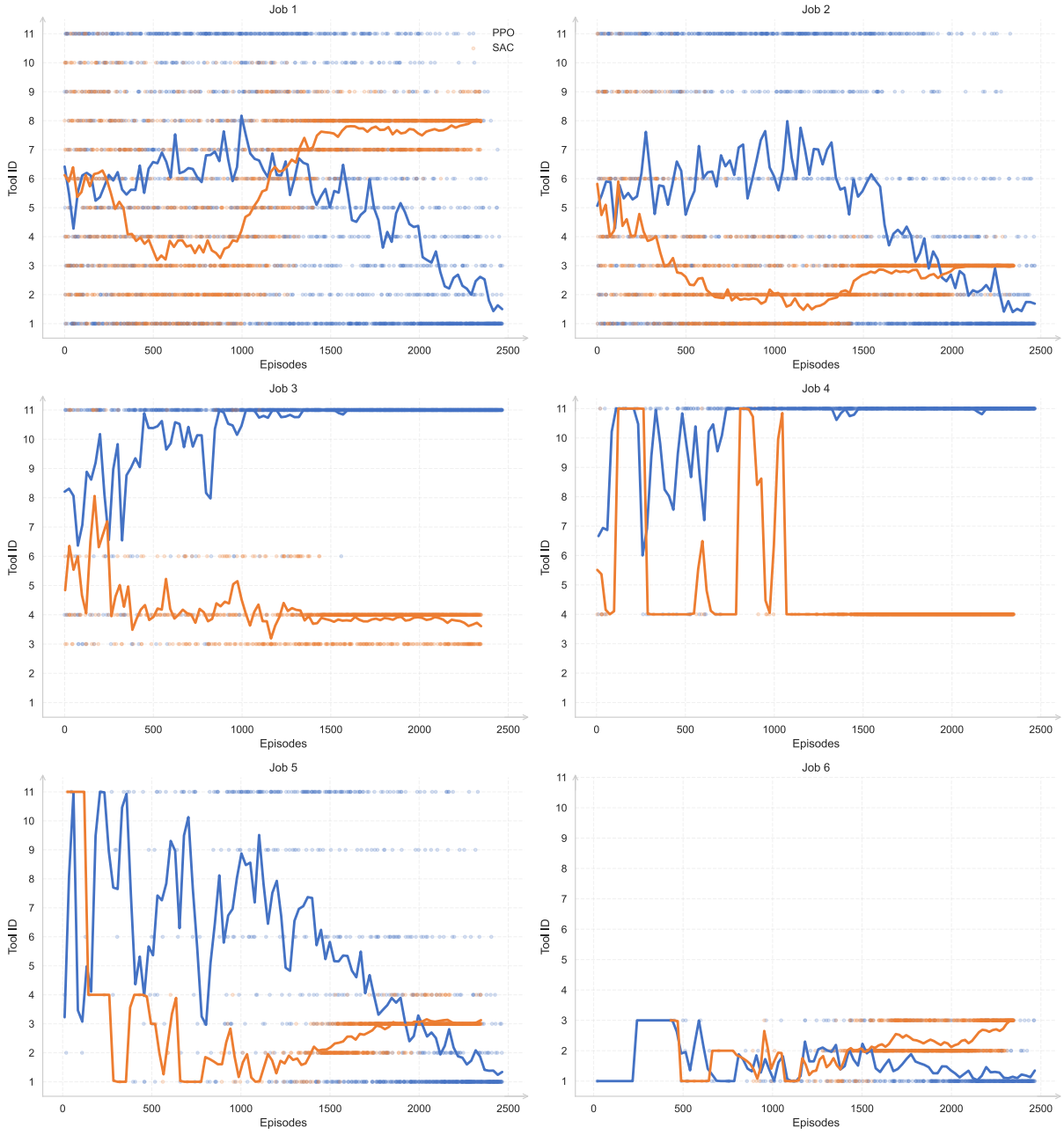


Figure 5.3: Tool selection evolution across training episodes for both algorithms on all six machining jobs

A striking observation is PPO's tendency to converge toward a small subset of tools across different jobs. For Jobs 1, 2, 5, and 6, PPO eventually settles on Tool 1 (D8 end mill) despite initially exploring various alternatives. This suggests that PPO develops a generalized preference for this particular tool rather than identifying job-specific optimal tools. For Jobs 3 and 4, PPO consistently selects Tool 11 (D20 end mill with 60mm cutting length), the tool with the longest cutting length in the library. This pattern indicates that PPO may be overgeneralizing, identifying certain tool characteristics (small diameter or long cutting length) as broadly beneficial rather than developing nuanced, job-specific selection strategies.

SAC, in contrast, demonstrates more deliberate and differentiated tool selection. For Job 1 (face milling), SAC displays an interesting learning trajectory, initially selecting smaller tools before discovering the benefits of larger face milling tools (moving from Tool 3 to Tool 7 around episode 500), then briefly experimenting with other options before finally settling on Tool 8 (D100 Corner face mill). This evolution suggests SAC effectively learns the appropriate tool type for face milling operations.

These tool selection patterns reveal that beyond differences in convergence speed and stability, the algorithms develop fundamentally different approaches to parameter identification. PPO tends toward generalization across similar job types, while SAC develops more specialized selections tailored to the specific requirements of each machining operation.

The training performance analysis establishes SAC as the more effective algorithm for CAM parameter identification in this specific implementation. The next section will examine how these differences in training characteristics compared against benchmark parameters and their machining performance.

5.3 Comparison with Benchmark Parameters

This section compares the parameter selections identified by the RL agents against industry benchmark parameters to assess the practical relevance and potential benefits of the developed framework.

5.3.1 Benchmark Parameter Selection

The benchmark parameters were systematically derived following industry-standard guidelines from the Hoffmann Group machining handbook (Hoffmann Group & Fraunhofer-Institut für Werkzeugmaschinen und Umformtechnik, 2014). Tool selection was performed manually based on the geometric requirements of each operation, while cutting depths were set to 1.00 x tool diameter as recommended in the handbook. Step factor values were directly adopted from the handbook's recommendations: 0.50 for roughing operations and 0.10 for finishing operations, with the latter specifically designed to achieve superior surface quality. Feed rates were determined using the Hoffmann Group ToolScout (*ToolScout*, n.d.), an online calculation tool for machining parameters, for S275JR steel, which approximates the material properties used in the force calculation model. Table 5.3 presents the complete set of resulting benchmark parameters that represent typical industry practice.

Table 5.3: Benchmark Parameters for Machining Operations

Job ID	Operation	Tool Selection	Feed Rate (mm/min)	Step Factor	Cutting Depth (mm)	Z-Axis Machining Allowance (mm)
1	Face Milling (Roughing)	8 (CornerMill_D100_L7)	1150	1	1	0
2	Contour Milling (Roughing)	4 (EndMill_D25_L50)	1600	0.50	25	0.50
3	Contour Milling (Roughing)	4 (EndMill_D25_L50)	1590	0.50	25	0
4	Contour Milling (Finishing)	4 (EndMill_D25_L50)	125	0.10	25	0
5	Contour Milling (Finishing)	3 (EndMill_D16_L38)	111	0.10	16	0
6	Pocket Milling (Combined)	3 (EndMill_D16_L38)	111	1	15	0

5.3.2 Implementation Challenges and Limitations

A significant challenge in this comparison is that the RL environment implemented in this thesis limited feed rates to a maximum of 200 mm/min, substantially below industrial recommendations for roughing operations. This limitation was initially established to create a controlled learning environment but unintentionally created a significant divergence from industrial practice.

For a fair comparison, a "modified benchmark" was created by constraining the industrial benchmark parameters to the same limitations as the RL environment. Specifically, any feed rates above 200 mm/min were capped at 200 mm/min, while maintaining all other parameter values.

It is equally important to note a fundamental limitation of the current comparison: the reward function used for training the agents prioritizes machining time and work (energy consumption) without explicitly considering surface quality. In industrial practice, surface quality is a critical consideration, particularly for finishing operations, and is one of the primary reasons for the low step factors (0.10) used in the benchmark parameters for Jobs 4 and 5. This limitation must be acknowledged when interpreting the performance improvements demonstrated by the RL agents.

5.3.3 Comparison of Parameter Selection Stategies

Figure 5.4 reveals noticeable differences in parameter selection strategies between the industrial benchmark and the RL agents, particulary regarding tool selection and step factors.

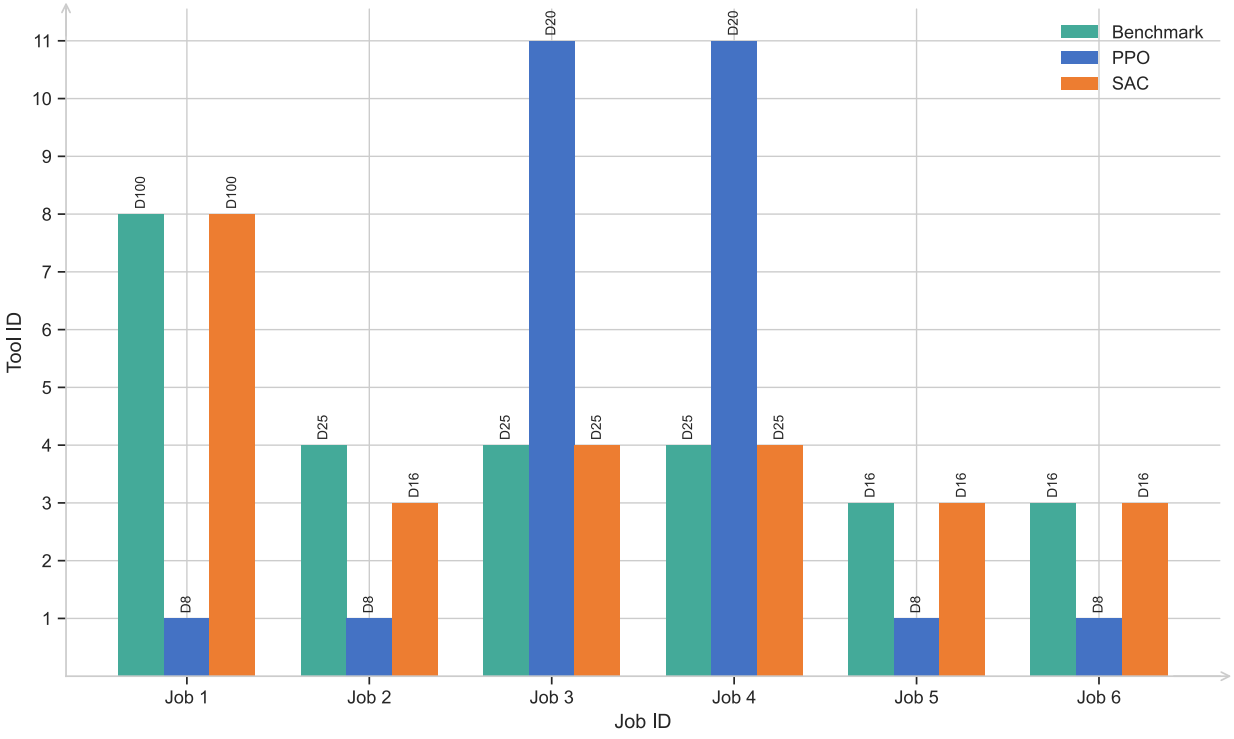


Figure 5.4: Comparison of Tool Selections Between Benchmark and RL Agent Across All Jobs

The tool selection comparison (Figure 5.4) reveals notable patterns across the different approaches. The SAC agent independently selected the same D100 corner mill for Job 1 as the benchmark, and showed similar choices to the benchmark for several other operations. In contrast, the PPO agent demonstrated a consistent preference for smaller diameter tools—particularly the D8 end mill (tool ID 1) for multiple jobs.

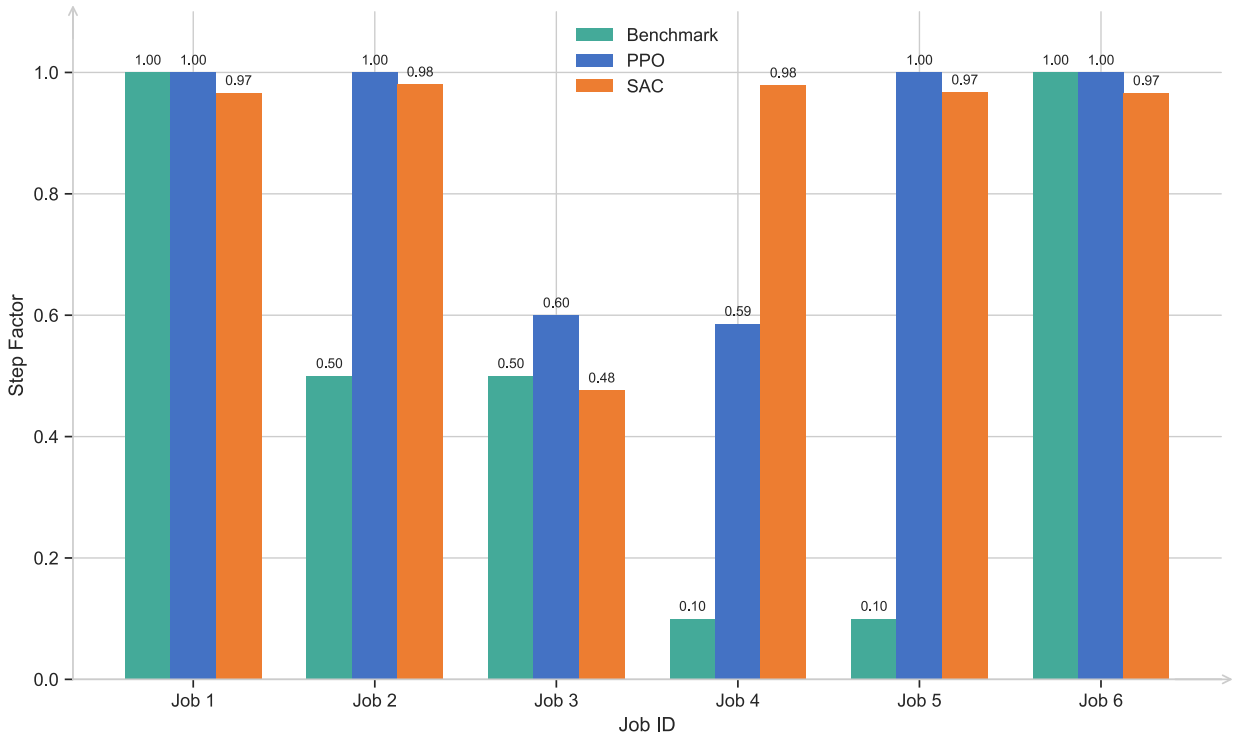


Figure 5.5: Comparison of Step Factors Between Benchmark and RL Agents

The step factor comparison (Figure 5.5) reveals even more significant differences in strategy. For Jobs 4 and 5 (finishing operations), the benchmark parameters use very low step factors (0.10) to ensure high surface quality. In contrast, both RL agents discovered that much higher step factors (0.60-1) improved their reward function outcomes. This stark divergence underscores the limitation of not incorporating surface quality into the reward mechanism, as the agents optimize for time and energy efficiency at what would likely be the expense of surface finish in a real manufacturing scenario.

5.3.4 Performance Comparison

Given the significant constraints in this comparison, particularly the feed rate cap and lack of surface quality consideration, direct performance comparisons have limited practical applicability. Nevertheless, examining the relative performance patterns across different jobs provides some insight into the agents' behavior.

Figure 5.6 shows the performance metrics across individual jobs:

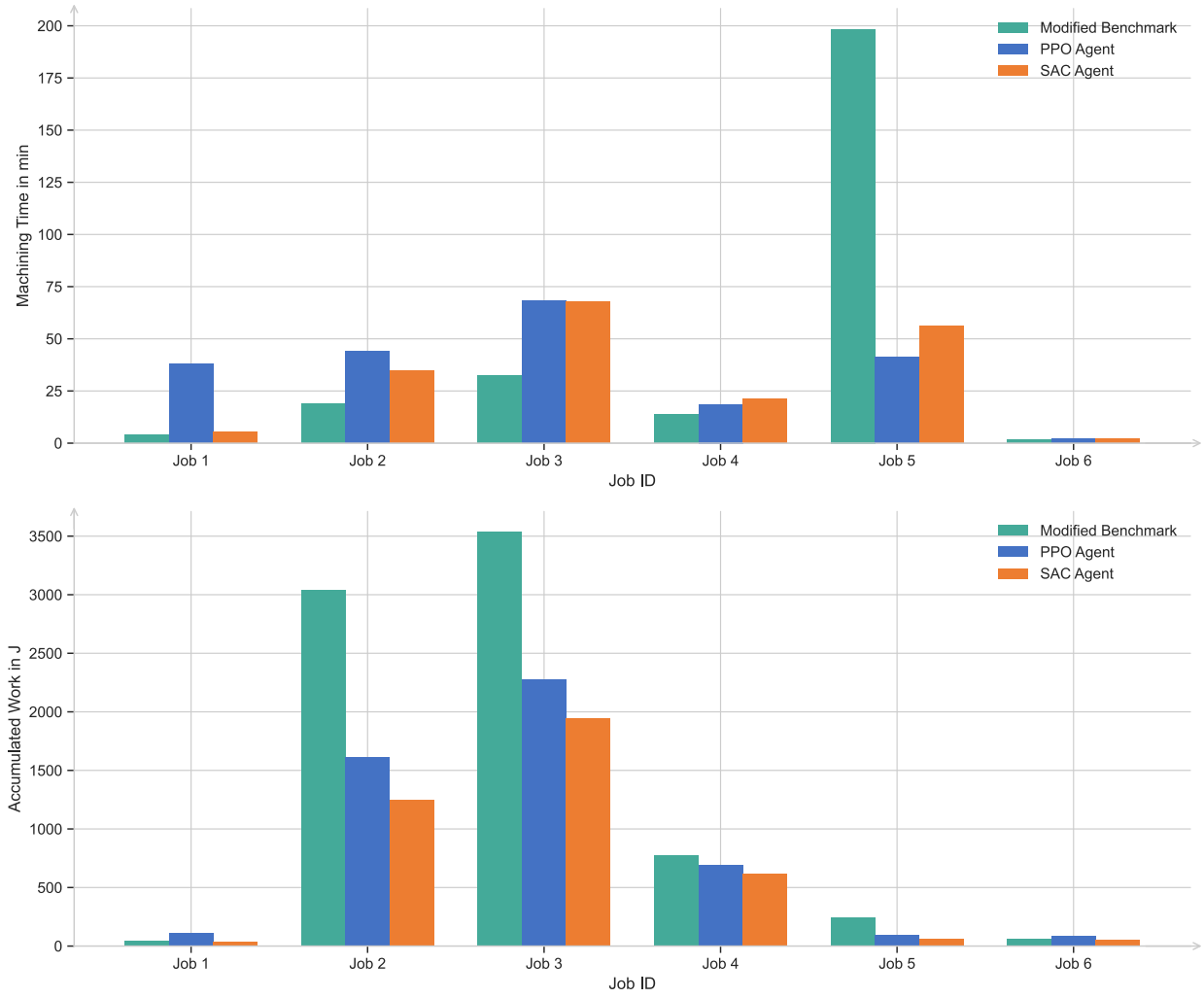


Figure 5.6: Comparison Machining Performance Benchmark and RL Agents Across Jobs

The visualization reveals that Jobs 2, 3, and 5 show the most pronounced differences between the benchmark and RL agents, particularly in machining time. These differences primarily come from the parameter selection strategies already discussed, especially the step factor values used for finishing operations.

While both agents demonstrate performance improvements over the modified benchmark, these results must be interpreted cautiously. The absence of surface quality considerations, artificial feed rate limitations, and disregard for tool life implications significantly constrain the practical relevance of this comparison. These limitations highlight the need for expanded evaluation criteria in future work.

Despite these constraints, the comparison demonstrates that RL approaches can effectively identify parameter combinations that optimize specified objectives within given constraints. Future implementations that incorporate additional manufacturing considerations, particularly surface quality metrics, would provide more meaningful comparisons against industrial benchmarks.

5.4 Summary of Key Findings

This chapter evaluated the developed reinforcement learning framework for automated CAM parameter identification. By systematically assessing how the PPO and SAC algorithms identify optimal process parameters, this chapter demonstrated the framework's capability to automate parameter selection while optimizing processing time and cutting forces. Key findings include:

Within the framework, SAC demonstrated superior capabilities, converging 24% faster than PPO while achieving 100% job completion rate compared to PPO's 95.20%. SAC also exhibited substantially higher decision-making stability (standard deviation of 0.77 versus 7.46), suggesting greater reliability for automated planning applications.

The framework revealed distinct parameter selection strategies between algorithms. PPO generated bimodal distributions with preference for extreme values and generalized tool selection (predominantly using the D8 end mill across operations). In contrast, SAC developed nuanced, operation-specific parameter selections that better approximated conventional machining practices despite having no prior domain knowledge.

Comparison with industrial benchmark parameters demonstrated the framework's ability to systematize parameter identification within its evaluation criteria. However, the current implementation's focus on machining time and accumulated work, without incorporating surface quality considerations, highlighted the importance of comprehensive objective functions, particularly evident in the selection of higher step factors for finishing operations than typically recommended.

The results confirm that the developed framework successfully addresses the objective of automating parameter identification within specified constraints. The SAC algorithm proved particularly effective due to its stability and sophisticated selection strategies. These findings establish a foundation for Chapter 6, which will discuss broader implications for manufacturing planning, address limitations, and propose future research directions to enhance the practical applicability of RL for CAM parameter identification.

Chapter 6 Discussion and Outlook

This chapter reflects on the developed RL framework for CAM parameter identification, examining its limitations, and potential improvements for manufacturing planning. The discussion contextualizes the findings presented in Chapter 5 while identifying promising directions for future research.

6.1 Framework Limitations

While the developed framework successfully demonstrated the potential of RL for parameter identification, several limitations should be addressed in future implementations.

Single-Part Design and Knowledge Transfer: A significant limitation of the current framework is its specialization for a single workpiece with predefined sequential operations. The environment design inherently constrains the agent's learning to this specific manufacturing context, creating a fundamental generalization challenge. Knowledge acquired during training cannot be directly transferred to different parts, materials or operation sequences, effectively requiring retraining for each new manufacturing scenario. This limitation contrasts with human expertise, which naturally generalizes across different parts with similar features.

Reward Function Constraints: The reward function was deliberately designed to focus on machining time and accumulated work, intentionally exluding surface quality metrics to simplify the initial framework. This design choice predictably influenced the agent's behavior, particularly regarding step factor selection for finishing operations (as shown in Figure 5.5). Despite the state representation identifying operations as either "roughing" or "finishing", the agent had no incentive to treat finishing operations differently without surface quality considerations in the reward mechanism.

Tool-Specific Parameter Constraints: The framework implemented uniform parameter ranges across all tools, regardless of their geometric characteristics or mechanical properties. While the conservative maximum feed rate limit (200 mm/min) effectively prevented potential tool failures in the current implementation, this approach doesn't account for the complex relationships between tool properties and appropriate parameter values. In real manufacturing scenarios, smaller diameter tools (2-4mm) require substantially different feed rates than larger tools (25-100mm) to prevent breakage and ensure effective cutting. Any future implementation allowing higher feed rates would necessarily require tool-specific parameter constraints.

Additional Parameter Considerations: Future implementations could expand the parameter space to include additional considerations such as rounding corners, tool entry/exit strategies, and different tool path patterns. These parameters significantly influence manufacturing outcomes but were beyond the scope of the current implementation. Inclusion of these parameters would more comprehensively reflect the full complexity of CAM programming decisions.

6.2 Conclusion

This thesis has developed and evaluated a reinforcement learning framework for CAM parameter identification that successfully integrates with professional CAM software and systematizes parameter selection for milling operations. While the current implementation demonstrates the feasibility and potential of this approach, significant opportunities exist for enhancing its capabilities and practical relevance.

The limitations identified in this chapter should not be viewed as fundamental shortcomings but rather as natural progression opportunities for a novel approach to a complex manufacturing challenge. Future research addressing these limitations, particularly regarding generalization across parts and materials and more comprehensive reward functions, would substantially advance the practical application of RL in CAM systems.

Bibliography

Altintas, Y. (2012). *Manufacturing automation: Metal cutting mechanics, machine tool vibrations, and CNC design* (2nd ed). Cambridge university press.

Gulde, R., Tuscher, M., Csiszar, A., Riedel, O., & Verl, A. (2019). Reinforcement Learning Approach to Vibration Compensation for Dynamic Feed Drive Systems. *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, 26–29. https://doi.org/10.1109/AI4I46381.2019.00015

Hoffmann Group & Fraunhofer-Institut für Werkzeugmaschinen und Umformtechnik (Eds.). (2014). *Garant ToolScout Zerspanungshandbuch: Bohren, Gewinde, Senken, Reiben, Sägen, Fräsen, Drehen, Spannen, Präzisionsschleifen* (2., veränderter Nachdruck 12/2014). Hoffmann Group.

Jarosz, K., Chen, Y.-T., & Liu, R. (2023). Investigating the differences in human behavior between conventional machining and CNC machining for future workforce development: A case study. *Journal of Manufacturing Processes*, *96*, 176–192. https://doi.org/10.1016/j.jmapro.2023.04.037

Jiang, Y., Chen, J., Zhou, H., Yang, J., Hu, P., & Wang, J. (2022). Contour error modeling and compensation of CNC machining based on deep learning and reinforcement learning. *The International Journal of Advanced Manufacturing Technology*, *118*(1–2), 551–570. https://doi.org/10.1007/s00170-021-07895-6

Kalpakjian, S., & Schmid, S. R. (2022). *Manufacturing engineering and technology, EBook, SI Units* (Eighth edition in SI units). Pearson Education, Limited.

Kaneko, K., Komatsu, T., Zhou, L., Onuki, T., Ojima, H., & Shimizu, J. (2023). Autonomous optimization of cutting conditions in end milling operation based on deep reinforcement learning (Offline training in simulation environment for feed rate optimization). *Journal of*

Advanced Mechanical Design, Systems, and Manufacturing, *17*(5), JAMDSM0064–JAMDSM0064. https://doi.org/10.1299/jamdsm.2023jamdsm0064

Leo Kumar, S. P. (2019). Knowledge-based expert system in manufacturing planning: State-of-the-art review. *International Journal of Production Research*, *57*(15–16), 4766–4790. https://doi.org/10.1080/00207543.2018.1424372

Li, W., Li, B., He, S., Mao, X., Qiu, C., Qiu, Y., & Tan, X. (2022). A novel milling parameter optimization method based on improved deep reinforcement learning considering machining cost. *Journal of Manufacturing Processes*, *84*, 1362–1375. https://doi.org/10.1016/j.jmapro.2022.11.015

Li, X., Zhang, S., Huang, R., Huang, B., Xu, C., & Zhang, Y. (2018). A survey of knowledge representation methods and applications in machining process planning. *The International Journal of Advanced Manufacturing Technology*, *98*(9–12), 3041–3059. https://doi.org/10.1007/s00170-018-2433-8

Lu, F., Zhou, G., Zhang, C., Liu, Y., Chang, F., & Xiao, Z. (2023). Energy-efficient multi-pass cutting parameters optimisation for aviation parts in flank milling with deep reinforcement learning. *Robotics and Computer-Integrated Manufacturing*, *81*, 102488. https://doi.org/10.1016/j.rcim.2022.102488

Lu, Y., Xu, X., & Wang, L. (2020). Smart manufacturing process and system automation – A critical review of the standards and envisioned scenarios. *Journal of Manufacturing Systems*, *56*, 312–325. https://doi.org/10.1016/j.jmsy.2020.06.010

Ma, H., Liu, W., Zhou, X., Niu, Q., & Kong, C. (2020). An effective and automatic approach for parameters optimization of complex end milling process based on virtual machining. *Journal of Intelligent Manufacturing*, *31*(4), 967–984. https://doi.org/10.1007/s10845-019-01489-6

Rekow, E. D. (2006). Dental CAD/CAM systems. *The Journal of the American Dental Association*, *137*, 5S-6S. https://doi.org/10.14219/jada.archive.2006.0396

Sutton, R. S., & Barto, A. (2020). *Reinforcement learning: An introduction* (Second edition). The MIT Press.

ToolScout. (n.d.). [Computer software]. Hoffmann SE. Retrieved April 17, 2025, from https://toolscout.com

Wang, Z., Lu, J., Chen, C., Ma, J., & Liao, X. (2022). Investigating the multi-objective optimization of quality and efficiency using deep reinforcement learning. *Applied Intelligence*, *52*(11), 12873–12887. https://doi.org/10.1007/s10489-022-03326-5

Xiao, Q., Li, C., Tang, Y., & Li, L. (2021). Meta-Reinforcement Learning of Machining Parameters for Energy-Efficient Process Control of Flexible Turning Operations. *IEEE Transactions on Automation Science and Engineering*, *18*(1), 5–18. https://doi.org/10.1109/TASE.2019.2924444

Zhang, T., Yuan, C., & Zou, Y. (2022). Online Optimization Method of Controller Parameters for Robot Constant Force Grinding Based on Deep Reinforcement Learning Rainbow. *Journal of Intelligent & Robotic Systems*, *105*(4), 85. https://doi.org/10.1007/s10846-022-01688-z

Affidavit

I hereby certify that I have written this thesis independently and have not used any sources or aids other than those specified. In case of doubt, the German version is legally binding.

Garching, 05.05.2025

Cornelius Gruss

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Garching, den 05.05.2025

Cornelius Gruss