Multi-Temporal Predictive Coding for Robotic Arm Control

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Robotics and Autonomous Systems
July 2025
Azeez Akinjide Jimoh

Supervisors:
Adjunct Prof. Hashem Haghbayan
Prof. Tomi Westerlund

UNIVERSITY OF TURKU
Department of Computing

Azeez Akinjide Jimoh: Multi-Temporal Predictive Coding for Robotic Arm Control

Master of Science (Tech) Thesis, 57 p.
Robotics and Autonomous Systems
July 2025

Predictive coding is a theoretical framework inspired by the brain's mechanisms for processing information. It holds significant potential for enhancing robotic control systems, particularly in dynamic and uncertain environments. Traditional control methods often rely on reactive error correction, which can lead to inefficiencies and degraded performance under changing conditions. In contrast, predictive coding manages these corrections proactively by minimizing prediction errors, resulting in more stable and resilient robotic behavior. This study develops and evaluates a predictive coding framework for the Franka Emika Panda robotic arm, enabling adaptive and robust performance in response to environmental changes. The proposed approach continuously updates the system's internal model by comparing predicted and actual sensory inputs in real time.

In this thesis, a hierarchical predictive model for robotic arm behavior is introduced, operating across multiple temporal scales (e.g., 0.2s, 0.3s, and 0.5s horizons) to anticipate future states and adapt motion accordingly. Complementing this structure is a vision-in-the-loop system, which incorporates real-time camera feedback not only for perception but also for predicting how the target will appear in future views. This joint prediction of both physical movement and visual perception is combined with an error minimization mechanism to ensure smooth and robust control, particularly during repetitive tasks and in environments with inherent uncertainty. By continuously minimizing discrepancies between predicted and actual camera views, the system refines its motion strategy to maintain optimal target visibility and centering when possible, while adaptively avoiding obstacles based on system confidence estimates derived from the error minimization mechanism.

Simulation-based experiments demonstrate that the framework improves control accuracy and robustness, especially in scenarios with high uncertainty and dynamic changes. These findings highlight the potential of predictive coding to advance adaptive robotic control, with future work aimed at real-world deployment and broader generalization.

Keywords: Predictive Coding, Robotic Control, State Estimation, Error Minimization, Vision in the Loop, Adaptive Control

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background Study

Predictive coding is a theoretical framework that models how the brain processes and predicts sensory information through hierarchical neural networks [1]. This framework has been extensively studied across various domains, including language comprehension where studies show the brain generates predictions about upcoming words and phrases [2]. The left temporal cortex handles lexico-semantic processing during language comprehension, managing prediction errors related to word usage and understanding. This region becomes particularly active when encountering unrecognized words, while the Inferior Frontal Cortex processes these prediction errors by updating and adjusting the brain's internal models [2]. Several comprehensive reviews have explored the mathematical foundations and algorithmic implementations of predictive coding [3], [4].

The brain continuously operates in a state of prediction, comparing expected outcomes based on prior knowledge against incoming sensory information. This predictive mechanism serves multiple purposes: reducing the cognitive load on sensory systems, anticipating future events to enhance decision-making, and facilitating learning through prediction error minimization.

Prediction error is the mismatch between predicted and actual sensory information. In robotics, this concept is crucial for developing adaptive control systems. By continuously monitoring and minimizing prediction errors, robotic systems can maintain accurate internal models of their environment, leading to more precise and reliable control [1]. This is particularly important in dynamic environments where sensory information may be noisy or incomplete.

Predictive coding is an important aspect for robotic control for tasks that requires precise control such as industrial automation and robotic manipulation, it does this by predicting sensory inputs and minimizing prediction error which allows robots to adjust their actions in real-time leading to improved performance in dynamic and uncertain environment [5]. It is also used in action inference and recognition, that is cognitive robots are able to infer and recognizes actions by reusing common circuits for both movement generation and action estimations just like the brain [6].In robot cognitive development predictive coding enables robots to build world models similar to human cognitive development which is important to develop adaptive and social perception in robots, enabling them to predict outcome of actions and refine their understand of their environment [6].

The application of predictive coding in robotics extends beyond basic control systems. It has been successfully implemented in various domains including autonomous navigation [7], human-robot interaction [6], and complex manipulation tasks [5]. The integration of vision-in-the-loop and visual servoing has been foundational for enabling robots to use visual feedback for real-time control and manipulation [8], [9]. The historical development of predictive coding in robotics traces back to early work on hierarchical control systems in the 1990s, with significant advancements in the last decade driven by improvements in computational power and machine learning techniques [1]. The importance of hierarchical learning for sensory-motor

systems and adaptive robot behavior has been extensively explored by Tani [10]. Recent work by Baltieri and Buckley [7] has demonstrated that traditional PID (Proportional-Integral-Derivative) control can be reinterpreted as a process of active inference with linear generative models, providing a theoretical bridge between classical control theory and modern predictive coding approaches.

1.2 Problem Statement

Traditional robotic control systems often struggle with dynamic and uncertain environments, where sensory information may be noisy, incomplete, or rapidly changing [1]. Studies have shown that conventional control approaches, such as PID controllers and model-based methods, can exhibit significant performance degradation in the presence of environmental uncertainty [7]. The work of Baltieri and Buckley [7] has shown that while PID controllers are effective in many scenarios, their performance can be enhanced by incorporating predictive coding principles, particularly in dealing with non-linear dynamics and environmental uncertainty.

Existing solutions typically rely on predefined models or reactive control strategies, which lack the ability to adapt to novel situations or learn from experience [6]. This limitation becomes particularly evident in complex, real-world scenarios where the system must continuously update its internal models and predictions.

This project addresses these challenges by implementing a predictive coding framework for the Franka Emika robotic arm. The system incorporates real-time perception and learning-based adaptation, enabling it to predict environmental changes and optimize control actions accordingly. This approach has the potential to improve robotic performance in various applications, from industrial automation to collaborative human-robot interaction.

1.3 Aim and Objectives

The aim of this study is to develop and evaluate an advanced hierarchical predictive coding control system for the Franka Emika Panda robotic arm. This system is designed as a multi-component architecture that integrates a hierarchical forward predictor with multiple prediction horizons for temporal abstraction, an adaptive memory system that learns from past experiences to improve future predictions, a comprehensive error detection and a state estimation system that maintains confidence-weighted predictions across multiple timescales. The predictive coding framework is intended to minimize prediction errors through real-time state prediction and error minimization, experience-based memory influence on predictions, hierarchical information flow between different temporal horizons, adaptive confidence-based prediction optimization, and dynamic error scaling based on prediction horizons.

Objectives

1. **Design a hierarchical predictive coding framework** that enables multi-horizon prediction and integrates error detection, state estimation, and memory across temporal scales.

2. **Develop an adaptive memory-augmented learning system** that retrieves and weights past experiences to enhance future predictions across different horizons.

3. **Create a robust error minimization and confidence estimation mechanism** using horizon-based confidence metrics, and optimization guided by memory integration.

4. **Evaluate system performance** using multi-horizon error analysis and memory retrieval metrics to validate predictive accuracy, confidence adaptation, and hierarchical influence.

1.4 Scope of Study

This research focuses specifically on the implementation of predictive coding principles for robotic arm control using the Franka Emika Panda platform. The study encompasses the development and implementation of predictive coding control within the ROS framework, with particular emphasis on obstacle avoidance to target tasks in simulated environments. The research concentrates on the integration of multi-horizon predictions at 0.2s, 0.3s, and 0.5s intervals, alongside the implementation of memory-based learning for trajectory optimization. While the study provides comprehensive evaluation of system performance through defined metrics and benchmarks, it excludes physical hardware implementation and real-world testing. The scope does not extend to complex manipulation tasks beyond obstacle avoidance to target operations, multi-robot coordination scenarios, or human-robot collaboration. Additionally, deep learning and neural network-based implementations fall outside the current research boundaries.

1.5 Significance of Study

This research makes substantial contributions to the field of robotic control through both technical implementations and practical applications. From a technical perspective, the study advances the understanding of predictive coding in robotics by demonstrating the effective integration of multi-horizon predictions in control systems.

The developed framework implements memory-based learning by storing and retrieving successful trajectories, a strategy that has been shown to improve adaptability in robotic manipulation and goal-oriented behavior [11], [12]. These implementations are particularly significant as they address fundamental challenges in robot control under uncertainty, using confidence-based heuristics and safety margins as described in classical robotics literature [13], [14].

The practical significance of this research extends to real-world applications in industrial automation and robotic manipulation. The developed system demonstrates improved performance in dynamic environments while reducing the traditional requirement for precise environmental modeling. This advancement is particularly valuable in industrial settings where environmental conditions may vary and precise modeling is challenging. Furthermore, the enhanced adaptability to changing task conditions provides a foundation for more sophisticated robot control systems, potentially influencing future developments in autonomous robotics and industrial automation.

The results presented in this thesis confirm the effectiveness of the proposed approach in simulation, highlighting its potential for future real-world applications.Techniques such as domain randomization have been proposed to bridge the gap between simulation and real-world deployment, enabling more robust sim-to-real transfer [15].

1.6 Thesis Structure

This thesis is organized into six comprehensive chapters. Chapter 1 serves as an introduction, establishing the foundational concepts of predictive coding, presenting the research motivation, and outlining the study's objectives, scope, and significance. Chapter 2 provides an extensive literature review, examining the theoretical foundations of predictive coding, analyzing existing implementations in robotics,

and identifying current challenges and limitations in the field.

Chapter 3 presents the methodology, detailing the experimental setup including the Franka Emika Panda platform, Intel RealSense depth camera configuration, simulation environment, and data collection processes. The chapter also outlines the system requirements, workflow procedures, and configuration parameters essential for the research implementation. Chapter 4 focuses on implementation, providing an in-depth examination of the system design including the state estimator, forward predictor, error detector, prediction controller, system memory, and other core components.

Chapter 5 presents the results and analysis, offering a detailed evaluation of the system's performance through trajectory analysis, hierarchical prediction performance, memory system effectiveness, and comprehensive error analysis. The thesis concludes with Chapter 6, which discusses the research implications, addresses limitations, and proposes directions for future work in predictive coding for robotic control.

This structure ensures a logical progression from foundational concepts to practical implementation, experimental validation, and critical reflection on the system's broader impact.

2 Literature Review

2.1 Introduction

The concept of predictive coding originated from Helmholtz's work in the 19th century. He proposed that the brain acts as a predictive model of sensory input. In other words, the brain makes predictions about what it expects to see, hear, or feel and compares these predictions with actual incoming signals [16]. This idea was further developed by Rao and Ballard [17]. Their research proposed a hierarchical predictive model where the higher brain areas send predictions to the lower area of the brain, allowing the brain to minimize prediction error by comparing what is expected with what is actually perceived. Clark [16] expands on this by presenting a broader framework for predictive coding that goes beyond just explaining perception. His framework allows for how actions are selected, attention is allocated, and how beliefs are updated. Recent reviews, such as Spratling [3], provide a comprehensive overview of the evolution of predictive coding algorithms from their neuroscientific origins to their computational implementations.

The core principles of predictive coding includes hierarchical prediction, where each level of the neural hierarchy generates predictions about the expected activity of the level directly below it; error minimization, which involves adjusting predictions based on the difference between expected and actual sensory input; and top-down/bottom-

up processing, where predictions (top-down) and sensory information (bottom-up) interact to refine perception [16], [17].

Empirical evidence supporting predictive coding has been shown in various studies which involve different aspects of the brain. For instance, in neuroimaging, studies have shown that areas such as the visual cortex exhibit activity patterns consistent with the predictions made by higher cognitive processes, validating the hierarchical model proposed by Rao and Ballard [17]. In language comprehension, studies show that the left temporal cortex is involved in generating words and phrases, while the inferior frontal cortex handles prediction errors for semantically inconsistent inputs [2].

The transition of predictive coding from neuroscience to robotics has been facilitated by its computational efficiency and adaptability. Clark [16] highlights how predictive coding serves as a unifying framework for understanding both perception and action, making it particularly relevant for robotic control systems. By mimicking the brain's predictive mechanisms, robots can better anticipate environmental changes and adjust their actions accordingly. Computational frameworks for predictive coding, as outlined by Bogacz [18], have facilitated the application of these principles in artificial systems, including robotics.

Predictive coding is particularly suitable for robotic control systems due to its ability to handle uncertainty and variability in real-world environments. By employing a predictive model, robots can continuously update their internal representations based on sensory feedback, leading to more robust and flexible behavior. This approach addresses the limitations of traditional robotics, which often rely on rigid, pre-programmed responses that fail to adapt to dynamic conditions.

2.2 Theoretical Foundations

The theoretical foundations of predictive coding provide the essential framework for our robotic control implementation. These principles, drawn from neuroscience and control theory, inform our multi-horizon prediction approach and memory-based learning system. Understanding these foundations is crucial for developing robust predictive control systems that can adapt to dynamic environments while maintaining stable performance.

2.2.1 Hierarchical Prediction in Predictive Coding System

In predictive coding, the brain is viewed as a hierarchical system that predicts sensory inputs at every level. Each level of the hierarchy generates predictions about the level below it, while the lower level computes prediction errors by comparing actual input with the predicted input. These errors are then passed upward to update the higher-level predictions. This process enables efficient processing and integration of sensory information [1], [7], [16], [19].

2.2.2 Multi-Horizon Prediction

The concept of hierarchical prediction in predictive coding, as proposed by Rao and Ballard [17], has been extended to robotic control through multi-horizon approaches. Hwang et al. [20] demonstrated how different temporal scales can be integrated for improved control, while Tani and Nolfi [21] showed how hierarchical learning can be implemented in sensory-motor systems. These works provide the theoretical foundation for implementing predictive coding across multiple time horizons in robotic control systems such as those explored by Yamashita and Tani [22], demonstrate the emergence of functional hierarchies for prediction and control in robotic systems.

Our multi-horizon prediction system implements this hierarchical approach, with

longer-horizon predictions influencing shorter ones, similar to the temporal hierarchy demonstrated by Hwang et al. [20] in their human-robot interaction system.

2.2.3 Error Minimization

Error minimization refers to the brain's process of reducing the difference between predicted and actual sensory inputs, known as prediction error. Error minimization is a central mechanism in predictive coding, as explored by Spratling [23], who describes how prediction errors are incrementally reduced through recurrent processing. This is achieved through a mechanism similar to gradient descent, where predictions are adjusted incrementally to reduce error over time [7]. Importantly, this process occurs at every level of the neural hierarchy, allowing each layer to refine its predictions and contribute to a more accurate understanding of the environment. Computational mechanisms for error correction, as reviewed by Franklin and Wolpert [24], are fundamental for adaptive control in both biological and robotic systems.

We implement this through the error detector and base prediction components, which continuously update model parameters based on prediction errors, following principles similar to those demonstrated by Lanillos et al. [25] in their adaptive robot learning system.

2.2.4 Connection to Bayesian Inference

Predictive coding is closely related to Bayesian inference, where the brain uses a probabilistic models to update beliefs about the world. The Bayesian brain hypothesis, as reviewed by Knill and Pouget [26], provides a theoretical framework for understanding how uncertainty and probabilistic inference are encoded in neural systems, closely aligning with predictive coding principles. Whenever the brain is

minimizing prediction error, it is actually doing Bayesian inference, updating its internal models to better handle uncertainty and make better predictions [7]. Bayesian inference mechanisms, such as those described by Deneve [27], offer computational models for how prediction errors are used to update beliefs in both biological and artificial systems.

Our memory system implements this Bayesian approach by weighting past experiences based on their relevance and success, similar to the experience-based learning demonstrated by Baltieri and Buckley [7] in their adaptive control architecture.

2.3 Predictive Coding for Robot State Estimation

State estimation is fundamental to our predictive coding implementation, as it forms the basis for generating accurate multi-horizon predictions and confidence-weighted control actions. This section explores how predictive coding enhances traditional state estimation approaches through continuous prediction and error minimization. Classical and modern state estimation techniques, such as those described by Barfoot [28], provide the foundation for integrating predictive coding with sensor fusion in robotic systems.

2.3.1 State Estimation

State estimation is the process of inferring the internal condition or state of a system based on observable outputs and available inputs. It enables informed decisions and control actions by providing the best possible guess about unobserved variables [29]. In robotics, this involves determining the robot's current status, such as joint positions, velocities, and other relevant parameters necessary for motion control.

In our implementation, state estimation is handled by the State Estimator com-

ponent, which integrates both proprioceptive (joint states), exteroceptive (camera states) and predictive information to maintain accurate robot state estimates. This approach builds on the work of Lanillos et al. [25], who demonstrated how predictive coding can enhance state estimation accuracy in robotic systems. Predictive coding principles have been successfully applied to sensorimotor integration and state estimation, as demonstrated by Adams et al. [30].

2.3.2 Body Schema Learning and Predictive Coding

Body schema learning refers to the process by which a robot develops an internal representation of its own body, similar to how humans and animals understand their own physical state or bodies. In traditional robotic systems, without predictive coding, robots typically rely on predefined mathematical models or methods, such as **inverse kinematics**. Inverse kinematics calculates the required joint parameters (such as angles and lengths) based on the desired position and orientation of the end-effector, using sensors like encoders to measure joint positions directly.

Another approach is **Direct Sensor Fusion**, where the robot integrates both **proprioceptive information** (internal sensors, such as joint encoders) and **exteroceptive information** (external sensors, such as cameras for visual input) to build and update its body schema. This fusion of internal and external sensory data enhances the robot's ability to understand its state and the environment in real-time. In our *Pick and Place* project [31], we employed this approach, allowing the robot's proprioceptive and exteroceptive sensors to work together, improving its comprehension of both its own movements and the surrounding environment.

With the integration of **predictive coding**, body schema learning becomes more dynamic and adaptable. The robot continuously updates its internal model based on prediction errors, which leads to a more accurate representation of its own body

in space.

The importance of body schema learning for **state estimation** lies in the robot's ability to more accurately estimate its position and orientation within its environment. By constantly updating its internal representation based on sensory inputs and minimizing prediction errors, the robot can better estimate its own state, such as joint angles, velocity, and spatial configuration. This results in more precise and reliable movement control, which is crucial for tasks requiring fine motor skills, like grasping and manipulation, and is key for effective autonomous behavior in dynamic environments.

2.4 Predictive Coding for Robot Control

Robot control through predictive coding represents a fundamental shift from reactive to predictive control paradigms. Our implementation demonstrates this transition through a multi-horizon prediction system that combines traditional control principles with advanced predictive coding mechanisms. This section explores various control approaches and their relationship to our predictive coding framework.

2.4.1 Control Methods in Predictive Coding

Traditional control methods, such as Proportional-Integral-Derivative (PID) and Model Predictive Control (MPC), are widely used in robotics [7]. These methods can be reinterpreted within a predictive coding framework, providing a theoretical bridge between classical and modern approaches.

PID Controllers operate by correcting errors between a desired setpoint and the actual state, using a weighted combination of the current error (Proportional), the accumulated past errors (Integral), and the rate of change of the error (Derivative).

While effective in many situations, PID controllers rely on fixed rules and are limited in adaptability to dynamic or uncertain environments [25].

Model Predictive Control (MPC) uses a mathematical model of the system to predict future states over a defined time horizon, similar to our multi-horizon prediction approach. However, as highlighted by Oliver et al. [32], traditional MPC lacks the adaptive learning capabilities inherent in predictive coding systems.

2.4.2 Model Reference Adaptive Controller

MRAC is a type of adaptive control scheme that adjusts controller parameters in real-time to force the system's behaviour to follow that of a predefined model. The controller works by comparing the actual performance of the system with what the reference model predicts, then adjusts the control gains to minimize error [33]. This approach shares similarities with our confidence-weighted control system.

Hwang et al. [20] demonstrated how adaptive control can be enhanced through predictive coding principles, particularly in handling temporal dependencies. Our implementation builds on this work by incorporating multiple prediction horizons and memory-based adaptation.

Table 2.1: Comparison of Control Methods

Controller	Description	Adaptability	Model Requirement	Computational Cost	Key References
PID	Error-based rule controller using proportional, integral, and derivative terms	Low	Low	Low	[7]
MPC	Predicts future system states using a model and optimizes control inputs over a time horizon	Medium	High	High	[32]
MRAC	Adapts control policy to match the behavior of a predefined reference model	High	Medium	Medium	[33]
AIC	Uses generative models to minimize prediction error through perception and action	Very High	Probabilistic Model	Medium–High	[25]

2.5 Learning and Adaptation in Predictive Coding Systems

Learning in predictive coding systems is driven by an ongoing feedback loop: the system generates predictions, compares them with actual sensory input, minimizes the resulting prediction error, and updates its internal state accordingly. This cyclical process represents a single experience. Over time, as the system repeats this loop across numerous interactions, it accumulates experiences that refine its internal representations. With each iteration, the system becomes increasingly adept at modeling the environment, mirroring the way biological organisms learn from experience.

2.5.1 Memory Systems in Predictive Coding

Memory systems in predictive coding frameworks have evolved from simple error storage to complex experience-based learning. Tani and Nolfi [21] proposed a hierarchical learning approach using recurrent neural networks, while Hwang et al. [20] demonstrated how temporal patterns can be learned and applied in robotic control. These works show how memory systems can be implemented to improve prediction accuracy and control performance.

Our memory system implementation diverges from traditional RNN-based approaches to better suit the requirements of real-time robotic control. Instead of maintaining a continuous state representation through neural networks.

2.6 Multi-modal and Visual Predictive Coding

Multi-modal integration is essential for robust robotic control, particularly in our predictive coding implementation where both proprioceptive (joint states) and exteroceptive (camera) information guide the Franka Panda's actions. This section explores how different sensory modalities are integrated within a predictive coding framework to enhance prediction accuracy and control performance, with particular emphasis on the fusion of visual and proprioceptive information that our system employs.

2.6.1 Multiple Modalities

Multi-modal predictive coding involves processing and integrating predictions across multiple sensory modalities such as vision, touch, sound, and proprioception simultaneously. This integration allows a system to form a more robust and coherent internal representation of the environment.

Proprioception is the body's ability to sense its own position, movement, and balance, such as how humans know where their limbs are and what they are doing without needing to look. In robotics, proprioception refers to internal sensing mechanisms, including joint positions, motor angles, force/torque, velocity, and acceleration. These are typically measured using joint encoders (for angles), inertial measurement units (IMUs) for orientation, and force/torque sensors for physical interaction.

Utilizing multiple sensory modalities is crucial for robust prediction. When one sensor fails or becomes unreliable, another modality can compensate. This approach was adopted by Hwang et al. [20], where their model integrates both vision and proprioception through separate but interconnected pathways. Each sensory stream is structured hierarchically: lower levels encode detailed features (e.g., pixel intensities for vision, joint angles for proprioception), while higher levels abstract these into patterns or system-level interpretations. This hierarchical coupling ensures that predictions in one modality can inform and influence predictions in another, leading to improved cross-modal consistency and error correction.

2.6.2 Visual Predictive Coding

Visual predictive coding enables a system to utilize visual inputs, such as images from a camera, to infer and predict its own body state. In the framework proposed by Sancaktar et al. [34], the robot generates a predicted visual input based on its current internal body estimate and compares this with the actual camera image. The discrepancy between the predicted and observed images is then used to update its internal model, allowing it to refine its understanding of body position and posture over time. This vision-based approach allows the robot to visually monitor and correct its bodily estimates in a self-supervised, closed-loop manner.

2.6.3 Vision in Loop Control System

The implementation of vision-in-loop control in this system draws from several key developments in robotic vision and predictive control. The architecture follows a hierarchical structure similar to the predictive coding framework described by [35], where visual feedback is continuously integrated into the control loop. The system employs a multi-object detection and tracking approach inspired by [36] and [37], but adapted for real-time robotic applications with specific focus on target and obstacle detection.

The visual servoing system incorporates predictive control principles, following the framework established by [38]. This is achieved through a three-tiered processing pipeline: the Multi Object Detector for raw visual data processing, the Camera Processor for real-time state maintenance, and the Camera View Predictor for future state estimation. This structure enables the system to not only react to current visual information but also anticipate future visual states.

The integration of these components allows for robust target tracking and obstacle avoidance, even in dynamic environments. The implementation follows the control principles described in [13], but extends them with predictive visual capabilities, creating a more anticipatory control system.

our vision-in-loop system incorporates several key innovations that enhance its performance and capabilities. The system implements real-time multi-object tracking with color-based segmentation, enabling efficient object detection and tracking. A sophisticated predictive camera view estimation system allows for anticipatory control actions. The dynamic obstacle tracking system incorporates confidence metrics to improve reliability, while integrated visual state prediction ensures smooth and accurate control. These features work together to maintain target visibility while

avoiding obstacles, demonstrating the practical application of theoretical frameworks in real-world robotic control scenarios.

2.7 Uncertainty in Predictive Coding

Uncertainty handling in predictive coding systems has been approached in various ways. Lanillos et al. [25] discussed different approaches to managing uncertainty in robotic control, while Oliver et al. [32] demonstrated practical implementations of uncertainty-aware control. These works provide insights into how uncertainty can be managed in predictive coding systems.

2.7.1 Confidence-Based Control

Confidence-based control approaches in predictive coding systems have been developed to handle varying levels of prediction certainty. Oliver et al. [32] showed how confidence metrics can be used to adjust control actions, while Lanillos et al. [25] demonstrated how uncertainty can be incorporated into control decisions. These approaches provide frameworks for implementing confidence-aware control in robotic systems like we have done in our own system.

3 Methodology

3.1 Introduction

This chapter outlines the methodological framework employed to implement and evaluate predictive coding for robot arm movement using the Franka Emika Panda robotic arm. We begin by introducing the hardware components, namely the Panda arm and the Intel RealSense D435 depth camera, followed by a detailed description of the camera-to-robot transformation used in our eye-in-hand configuration. We then describe the simulation environment built using ROS and Gazebo, the real-time control system enabled by MoveIt Servo, and the full data processing pipeline, from perception to motion execution. A system architecture diagram illustrates how these components interact.

3.2 The Franka Emika Panda

The Franka Emika Panda stands out as our top choice for predictive coding research in simulation. This 7-DOF arm comes with detailed simulation models that capture its real-world dynamics, including the joint torque sensing that's crucial for our prediction-error feedback approach.

We chose the Panda over other options mainly because of practical software con-

Figure 3.1: Franka Emika[39].

siderations. The Franka ROS packages provide ready-to-use simulation tools with accurate models and well-documented interfaces, which saved us implementation time. Panda has a large community support for simulation resources and also numerous GitHub repos and academic implementations we could build upon.

The Panda's key specifications make it particularly suitable for our work[39]:

- 7 degrees of freedom with a 3kg payload capacity
- Joint torque sensors at all joints with 1kHz sampling rate
- $\pm$0.1mm position repeatability
- 0.85m reach with a compact footprint
- Cartesian impedance control with configurable stiffness (0.1-3000 N/m)

Gazebo works with the Panda, giving us physics-accurate simulations that reliably reproduce the arm's behavior essential when we're training and validating our predictive models. The MoveIt library integration is another time-saver, offering a clean

interface for executing trajectories without getting bogged down in low-level control details.

The active developer community around the Panda means simulation tools keep improving, and techniques developed in simulation transfer more reliably to hardware. This practical combination of open-source tools, accurate models, and community support lets us focus on our predictive coding algorithms rather than fighting with platform-specific quirks.

3.3 Intel RealSense Depth Camera D435

Figure 3.2: Intel Realsense Depth Camera D435[40].

The D435 is Intel's depth-sensing camera built for robotics and computer vision tasks. It merges traditional RGB imaging with stereo depth sensing to capture both color images and precise depth measurements. At its core, the camera pairs infrared stereo vision with Intel's D4 processor to create depth maps in real-time.

This makes the D435 especially useful for robotic systems that need to understand their surroundings and detect objects [40].

Key specifications of the D435 include a depth field of view of 87° × 58° × 95° (Horizontal × Vertical × Diagonal) and an RGB field of view of 69.4° × 42.5° × 77°. The depth camera operates at up to 90 frames per second with a resolution of 1280 x 720, while the RGB sensor captures at 1920 x 1080 resolution at 30 fps. The camera's effective depth range extends from 0.2 meters to approximately 10 meters, with optimal performance between 0.3 and 3 meters [40]. In our simulation environment, these specifications are maintained without the physical limitations and noise patterns typically present in real-world deployments.
For collecting data from the camera we setup a ROS node called Multi-Object Detector, this node processes real-time feeds from the camera utilizing both RGB and depth sensors. The node subscribe to the raw RGB and depth image topics, where it performs color-based detection to identify blue targets and green obstacles using HSV color thresholding. When objects are detected, the node combines the 2D pixel locations from the RGB image with corresponding depth data to calculate precise 3D positions in world coordinates. This transformation from camera to world coordinates uses the camera's intrinsic parameters and TF transformations. The detected object positions are then published as ROS messages on the `/blue_object_detection` and `/green_object_detection` topics, which are used by the predictive coding controller for target tracking and obstacle avoidance.

3.4 Camera To Franka Arm Transformation

The RealSense D435 camera is mounted on the Franka robot's end-effector (eye-in-hand) using a fixed transform defined in the URDF XACRO file of the robot arm. The camera is positioned 5 centimeters forward and 2 centimeters to the right of

the hand's center, with no vertical offset. Its orientation is set by rotating -90 degrees around the Y-axis (pitch) and 180 degrees around the Z-axis (yaw), effectively pointing the camera's view perpendicular to the robot's hand. This transform configuration is automatically integrated into ROS's Transform (TF) system, allowing the robot to accurately convert between camera coordinates and world coordinates when detecting objects.

3.5 Simulation Environment

For the totality of this project we use Gazebo simulator along side ROS(Real-time operating system) to perform the action of controlling the arm and moving towards the target from the camera feed while avoiding the obstacles along it's path using the predictive coding model. We make use of libraries such as `libfranka`, `franka_ros` and MoveIt for getting joint state of the arm and sending move commands, for the camera we used `realSense` provided library for getting all readings. see Figure 3.3

3.6 MoveIt Servo

The predictive coding control system interfaces with the Panda robot through MoveIt Servo's velocity control framework, chosen for its ability to handle real-time control based on continuous model predictions. The control flow begins with the joint processor node subscribing to `/joint_states` to monitor the robot's current state at 100Hz. This state information feeds into our predictive coding model, which generates velocity commands based on the difference between predicted and actual states, particularly when tracking detected objects from the RealSense D435 camera mounted on the end-effector.

The velocity commands are published through the `/servo_server/delta_twist_cmds` topic at 10Hz, as configured inside the controller node. MoveIt Servo internally han-

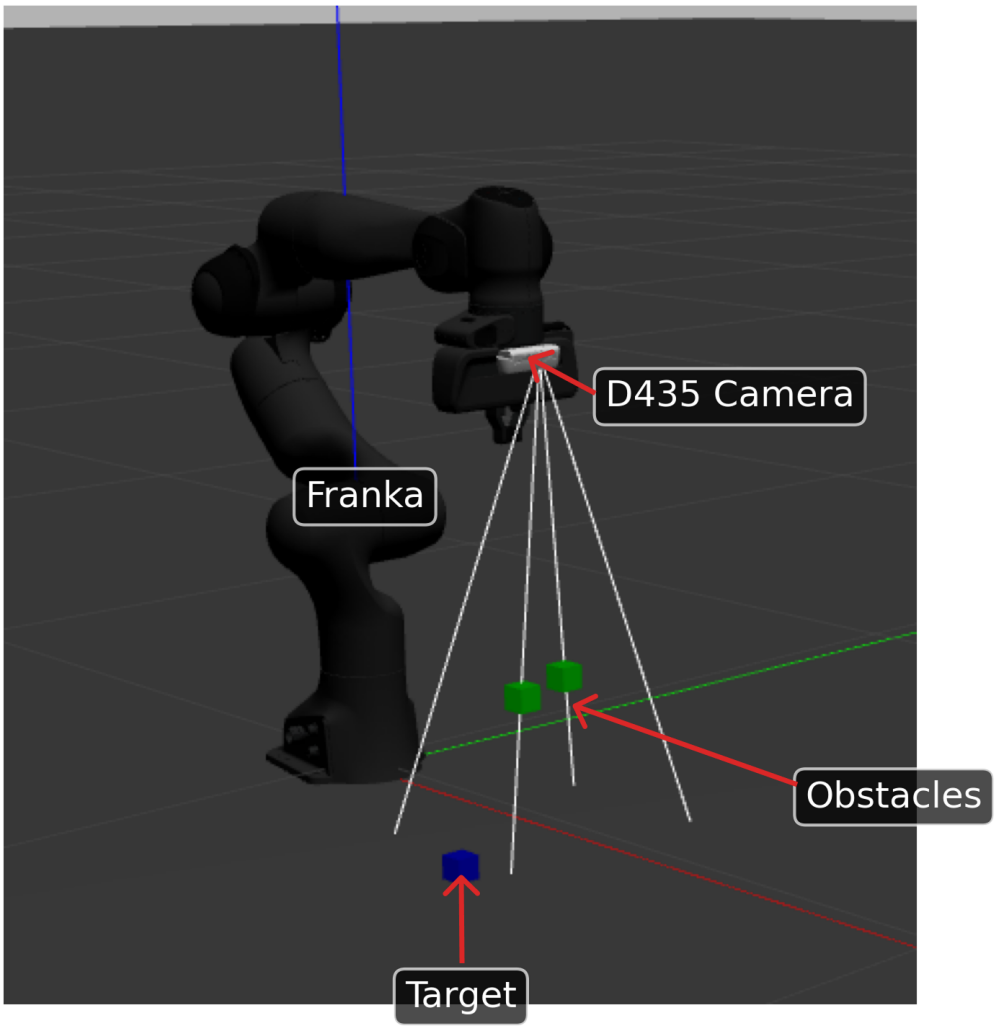


Figure 3.3: Simulation Environment

dles the inverse kinematics through Jacobian calculations, converting our Cartesian velocity commands into joint velocities while respecting the robot's kinematic constraints. This approach provides more responsive control compared to position-based planning, which would require complete trajectory generation, or torque control, which is more suited for force-sensitive tasks.

For Servo parameter tuning, we maintain the default velocity scaling factor of 0.5 m/s for Cartesian movements, balancing between responsive motion and safe operation. The command update rate of 10Hz was chosen experimentally to allow sufficient time for the predictive model to process sensor data and generate mean-

ingful predictions while maintaining smooth motion. This configuration provides a practical balance between computational load and control responsiveness, though higher rates could be achieved with further optimization.

3.7 Workflow Diagram

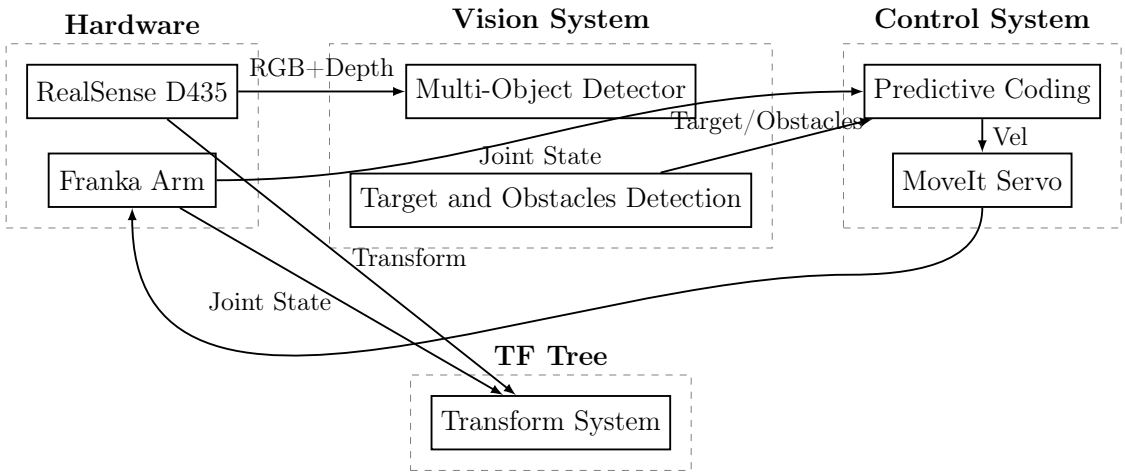


Figure 3.4: System Architecture Overview

The system architecture integrates three main components: the vision pipeline, transform system, and control loop. The vision pipeline begins with the RealSense D435 camera providing RGB and depth images, which are processed by the Multi-Object Detector to identify targets and obstacles in the environment. The transform system plays a crucial role in maintaining spatial relationships, converting camera detections to world coordinates, and tracking the robot's state through joint positions. These processed positions feed into the control loop, where the Predictive Coding controller generates velocity commands based on its predictions. These commands are then executed on the Franka Arm robot through MoveIt Servo, enabling real-time control based on visual feedback and predictive modeling.

3.8 Data Collection Process

The performance of the predictive coding control system was primarily evaluated within a simulation environment using ROS (Robot Operating System) and Gazebo, specifically with the Franka Emika Panda robot model and RealSense depth camera attached to it. The system implemented comprehensive performance measurement through multiple metrics tracked by the DataLogger component, including prediction accuracy (position and velocity errors across different horizons), system confidence levels, memory system effectiveness (success rates, retrieval efficiency, and influence metrics), and hierarchical performance indicators (inter-horizon influences and constraint strengths). These measurements were captured through sophisticated error detection mechanisms, confidence estimation algorithms, and memory system analytics, all integrated within the ROS framework. The implementation shows that testing was conducted exclusively in simulation, utilizing simulated sensor data and state estimation, suggesting that future work could benefit from real-world testing on the physical Franka Emika Panda robot to validate the system's performance in actual operating conditions and compare it with simulation results.

3.9 System Requirements

The project requires Ubuntu 20.04 LTS (Focal Fossa) as the operating system, paired with ROS Noetic (ros-noetic-desktop-full) for the robotics framework. The simulation environment is built on Gazebo 11.x, which integrates with MoveIt 1.1.x for motion planning and RViz for visualization. Essential ROS packages include `franka_ros` for the Panda robot interface, `moveit_servo` for real-time control, `realsense2_camera` with Intel RealSense SDK 2.0 for vision, and supporting packages like `tf2_ros`, `cv_bridge`, and `image_geometry`. The software stack relies on Python 3.8 or higher, with key dependencies including NumPy for numerical com-

putations, OpenCV for image processing.

4 Implementation

The predictive coding implementation for the Franka Panda robot integrates multi-horizon predictions with memory-based learning to enhance robot control capabilities. The system is designed to continuously predict and update its internal models while generating confidence-weighted control actions, building upon established predictive coding principles.

4.1 State Estimator

The State Estimator fuses multi-modal sensor data to construct a temporally coherent representation of the robot's state. It integrates input from the Joint Processor, which extracts end-effector position and velocity from joint sensors, and the Camera Processor, which identifies target and obstacle positions using RGB-D and point cloud data. The estimator maintains a history of the 50 most recent states to support robust prediction and analysis.

The estimator outputs a Robot State object containing the current position, velocity, target position, distance to target, target visibility, and detected obstacles. A feature vector, including the current position, velocity, target coordinates, and prediction time horizon, is extracted from the latest state for use in predictive modeling.

4.2 Predictive State Manager

The Predictive State Manager system employs a hierarchical predictive coding architecture that integrates physics-based predictions with learned experiential patterns. It is composed of three key components: the Forward Predictor as the central coordinator, the Hierarchical Predictor for multi-horizon predictions, and the Base Predictor for foundational state estimation.

4.2.1 Forward Predictor

The Forward Predictor orchestrates interactions among the prediction modules, error detection system, and memory network. It maintains predictions across multiple horizons—specifically at 0.2s, 0.3s, and 0.5s—and continuously tracks errors and performs verification. It also generates future camera view predictions to assess visual consistency.

This component operates in a continuous loop defined by a prediction-verification-adjustment cycle:

$$\text{State}_{t+1} = \text{Predict}(\text{State}_t) + \text{Verify}(\text{Prediction}_t) + \text{Adjust}(\text{Error}_t) \tag{4.1}$$

4.2.2 Base Predictor

The Base Predictor handles the core prediction process, combining multiple velocity sources into a single desired output:

$$\vec{v}_{desired} = \vec{v}_{target} \cdot w_{target} + \vec{v}_{view} \cdot w_{view} + \vec{v}_{center} \tag{4.2}$$

Weighting factors for target and view-based velocities are dynamically computed.

In particular, w_{target} is defined as $1.0 - (e_{view} \cdot w_{view})$, with w_{view} being a predefined constant reflecting the weight of camera view error. The centering term $\vec{v}_{center}$ provides an additional corrective force to stabilize movement.

Error prediction aggregates multiple components, each weighted according to its importance:

$$\text{Total Error} = \sum_i w_i \cdot e_i \tag{4.3}$$

Over time, correction factors are updated adaptively using:

$$\text{Correction}_{new} = \text{Correction}_{old} \cdot (1 - \alpha \cdot \text{Error}) \tag{4.4}$$

where α determines the learning rate.

The base predictor continuously refines its predictions by updating a set of correction factors based on recent prediction errors and experience. After each prediction, the system compares the predicted and actual outcomes, calculating errors in position, velocity, camera view, and centering. These errors are used to adjust internal correction factors, which act as scaling multipliers for the next prediction cycle.

When generating the desired velocity for the robot, these correction factors are applied to the base velocity components (target direction, view-based adjustments, and centering). As a result, the final velocity command is not only directed toward the target but is also adaptively scaled and corrected to account for recent errors and learned behaviors, leading to more accurate and robust motion in dynamic environments.

The base predictor leverages the memory system by retrieving similar past experiences for the current state and prediction horizon. These experiences are used to dynamically adjust the weighting factors for target- and view-based velocities. Successful past experiences contribute to the calculation of a scalar memory influence, which modulates the final velocity output, allowing the system to adapt its predictions based on learned behaviors and outcomes.

Obstacle avoidance is incorporated by introducing an avoidance correction vector, which is computed based on the predicted error and the presence of obstacles in the environment. This vector is normalized and blended with the target and view velocities, with the blending factor determined by the predicted error. This mechanism enables the base predictor to generate safe and adaptive trajectories, even in the presence of dynamic or unexpected obstacles.

4.2.3 Confidence Calculation

Prediction confidence is determined by multiple factors, including time horizon, distance from the current state, predicted error (which incorporates camera view quality), and the base confidence. In implementation, these factors are combined as a weighted sum:

$$\text{Confidence} = 0.2 \cdot \text{base\_conf} + 0.3 \cdot \exp(-t) + 0.3 \cdot \exp(-d) + 0.2 \cdot \exp(-e) \quad (4.5)$$

where t is the prediction time horizon, d is the spatial distance to the current state, and e is the estimated prediction error (including camera view quality).

4.2.4 Hierarchical Predictor

The Hierarchical Predictor performs multi-horizon forecasting while satisfying multiple constraints. The satisfaction of constraints is quantified using:

$$\text{Satisfaction} = \sum_i w_i \cdot c_i \tag{4.6}$$

Here, each w_i denotes a weight assigned to a particular constraint, and c_i reflects the satisfaction score for that constraint. These include position consistency ($\exp(-\|\Delta p\|)$), velocity consistency ($\exp(-\|\Delta v\|)$), confidence consistency ($(1 - |\Delta c|)$), and camera view consistency ($(1 - |\Delta q|)$).

Constraint satisfaction values influence the blending factor used for adaptive corrections:

$$\text{Blend Factor} = (1 - \text{Satisfaction}) \cdot \text{Error Adjustment} \cdot \frac{1}{\text{Movement Factor}} \tag{4.7}$$

This mechanism allows the system to modulate corrections based on the degree to which constraints are violated.

4.2.5 Error Propagation

To maintain temporal consistency, the system propagates error between prediction horizons. The propagated error is computed as:

$$\text{Error Propagation} = \sum_i w_i \cdot |\Delta e_i| \tag{4.8}$$

where Δe_i represents the difference in error between consecutive horizons and w_i is the corresponding component weight.

The forward predictor continuously learns from prediction discrepancies by adjusting its internal parameters and preserving a history of past predictions for validation. The magnitude and speed of learning are scaled according to the prediction horizon and the prevailing error metrics, ensuring both short-term responsiveness and long-term stability.

4.3 Error Detector

The Error Detector component is responsible for computing and tracking various types of prediction errors in the predictive coding system. It takes a predicted state, actual state, predicted camera view, and actual camera view as inputs, and calculates several error metrics: position errors (both overall Euclidean distance and directional vectors), velocity errors (overall and directional vectors), and camera view errors (including target visibility, view quality, and centering error). These errors are combined using adaptive weights that are updated based on error trends.The camera view errors are further decomposed into view quality (70%) and centering error (30%).

In addition to these scalar error metrics, the Error Detector also computes an obstacle avoidance vector. This vector indicates the optimal direction for the robot to move in order to avoid obstacles while still progressing toward the target. The avoidance vector is included in the error metrics and is used in the Base Predictor to influence the desired velocity of the arm to generate safe and adaptive trajectories.

The total error is computed as:

$$E_{total} = w_{pos}E_{pos} + w_{vel}E_{vel} + w_{cam}(0.7E_{view} + 0.3E_{center}) \tag{4.9}$$

where E_{pos}, E_{vel}, E_{view}, and E_{center} are the position, velocity, view quality, and centering errors respectively, and w_{pos}, w_{vel}, and w_{cam} are their corresponding adaptive weights. The component maintains a history of these errors (up to 100 samples) and computes error trends using a moving average of the last 10 samples. These trends are used to adaptively update the error weights, allowing the system to prioritize different types of errors based on their recent performance. The error metrics it produces are used by other components, particularly the Hierarchical Predictor, to adjust their predictions and learn from mistakes, forming a key part of the predictive coding feedback loop.

4.4 Prediction Controller

The Predictive Coding Controller acts as the central processor of the system, employing a hierarchical control structure that integrates multiple components into a continuous prediction–error–learning loop. It coordinates four core modules: the State Estimator for processing sensor data and the current robot state, the Forward Predictor for generating hierarchical predictions, the Error Detector for computing prediction errors, and the Memory System for experience storage and retrieval.

Velocity commands are generated through a layered prediction mechanism, formalized as:

$$\vec{v}_{\text{cmd}} = \vec{v}_{\text{immediate}} + \vec{v}_{\text{trajectory}} + \vec{v}_{\text{strategic}} \tag{4.10}$$

Each term in this equation corresponds to a specific control signal: $\vec{v}_{\text{immediate}}$ re-

flects short-term predictions; $\vec{v}_{\text{trajectory}}$ accounts for trajectory adjustments based on medium-term predictions; $\vec{v}_{\text{strategic}}$ handles long-term strategic planning.

To ensure safe and stable motion, the controller enforces several constraints: the total command velocity is capped at $\|\vec{v}_{\text{cmd}}\| \leq 0.5$ m/s.

The system operates across multiple prediction horizons, each with its own confidence threshold. Short-term predictions guide immediate responses, medium-term predictions adjust trajectories when the confidence exceeds 0.6, and long-term predictions steer strategic behavior when confidence exceeds 0.8. This layered structure allows the controller to respond effectively across temporal scales, balancing responsiveness with long-term planning.

4.5 System Memory

The Memory System implements an experience-based learning mechanism that stores and retrieves robot movement experiences. Operating with a fixed-size buffer of 1000 experiences, it maintains three primary memory structures: a general experience queue, an obstacle-specific experience queue, and a trajectory memory for successful movements. Each experience encapsulates the robot's state, prediction, actual outcome, and error metrics at a given time. The system employs a similarity-based retrieval mechanism where:

$$S(s_1, s_2) = 0.5 \cdot e^{-\|\vec{p}_1 - \vec{p}_2\|} + 0.2 \cdot e^{-\|\vec{v}_1 - \vec{v}_2\|} + 0.3 \cdot S_{obstacle} \tag{4.11}$$

where $S(s_1, s_2)$ represents the similarity between two states, with position similarity weighted at 50%, velocity similarity at 20%, and obstacle similarity at 30%. Experiences are classified as successful using a relative improvement approach - an

experience is successful if its error is below the average of recent experiences, with a fallback to an absolute threshold ($\epsilon < 0.5$). These successful experiences are stored in the trajectory memory for future reference. The system supports retrieval of similar experiences (k-nearest neighbors) and prioritizes obstacle experiences when obstacles are present. Memory influence on predictions scales with the success ratio of similar experiences, with a minimum influence of 10% when no successful experiences exist. The success rate is continuously updated as $r_{success} = n_{successful}/n_{total}$, providing a metric for learning progress.

4.6 Camera Processing System

The camera processing system consists of three main components: the Multi Object Detector for raw visual detection, the Camera Processor for real-time state processing, and the Camera View Predictor for future state prediction. Together, these components provide a comprehensive visual perception system that handles both current and predicted camera views.

4.6.1 Multi Object Detector

The Multi Object Detector serves as the foundation of the visual system, processing raw camera data from the RealSense D435. It uses HSV color segmentation to detect blue targets and green obstacles in real time. Detection is depth-aware through the use of camera intrinsics, and all coordinates are transformed from the camera's optical frame to the world frame. Contour-based object tracking with area thresholding ensures robustness, while debug visualizations overlay coordinate data for verification.

4.6.2 Camera Processor

The Camera Processor maintains the current visual state by processing data from the detector. It tracks both targets and dynamic obstacles, estimating their velocities and monitoring visibility status. TF2 is used for managing camera transforms, and occlusion is detected through depth comparisons.

To assess view quality, the processor computes several metrics: target visibility confidence, obstacle interference, safe viewing angles, and view ratios. These metrics help determine the suitability of the current view for control purposes. Obstacle histories are stored and updated using a sliding window approach for velocity estimation. A cleanup mechanism ensures stale data does not affect tracking accuracy.

4.6.3 Camera View Predictor

The Camera View Predictor forecasts future camera views based on motion and scene dynamics. Future transforms are estimated using:

$$T_{\text{predicted}} = T_{\text{current}} + \vec{v} \cdot \Delta t \tag{4.12}$$

It then predicts object positions and evaluates future view quality by estimating visibility, potential occlusions, interference levels, and viewing angles. The system anticipates dynamic obstacle movement and adjusts predictions accordingly.

This predictor uses a hierarchical model that accounts for camera motion, target dynamics, obstacle trajectories, field-of-view constraints, and depth-based occlusion. Consistency between current and predicted views is maintained using shared utility functions for coordinate transformations, view ratio calculations, occlusion detection, interference computation, and visibility quality assessment.

This integrated approach ensures that all visual data, current and forecasted are processed uniformly, providing reliable visual feedback to the control system.

4.7 Prediction Loop

The system implements a continuous prediction → control → correction loop inspired by predictive coding principles. In the prediction phase, the current robot state (position, velocity, target position, camera view) is used to generate predictions for the next robot state and camera view across multiple time horizons. These predictions are generated by the Base Predictor and Hierarchical Predictor, ensuring consistency between short-term and long-term predictions. In the control phase, the predicted next state (and its confidence) is used to generate control commands for the robot, guiding it toward the predicted state while maintaining target visibility and avoiding obstacles when required. In the correction phase, the actual observed robot state and camera view are compared with the predictions, and Error Detector computes error metrics (position error, velocity error, camera view error) and also obstacles avoidance vectors. These errors are used to update correction factors in Base Predictor, which influence future predictions to reduce prediction error. The Hierarchical Predictor ensures consistency between predictions at different time horizons, propagating errors and adjusting predictions accordingly. This loop creates a robust, error driven system that continuously refines its predictions and actions, aligning with the predictive coding principle of minimizing the difference between predicted and actual sensory input.

4.8 System Uncertainty

The system incorporates a robust uncertainty model to quantify and manage prediction errors across multiple time horizons. The uncertainty is primarily captured

through confidence scores and error metrics. In the prediction phase, each predicted state (position, velocity, camera view) is assigned a confidence score, which reflects the system's belief in the accuracy of the prediction. This confidence is calculated based on factors such as the time horizon (longer horizons yield lower confidence), the distance between predicted and current states, and the magnitude of predicted errors. The Error Detector computes detailed error metrics (position error, velocity error, camera view error) by comparing predicted states with actual observed states. These error metrics are used to update correction factors in Base Predictor, which influence future predictions to reduce uncertainty. Additionally, the Hierarchical Predictor ensures consistency between predictions at different time horizons, propagating uncertainty and adjusting predictions accordingly. While our system does not explicitly model uncertainty using a full probabilistic framework (e.g., precision weighting or free-energy minimization), it effectively uses confidence scores and error driven corrections to manage uncertainty in a practical, real-time manner. This approach allows the system to adapt dynamically to changing conditions, improving prediction accuracy and overall performance.

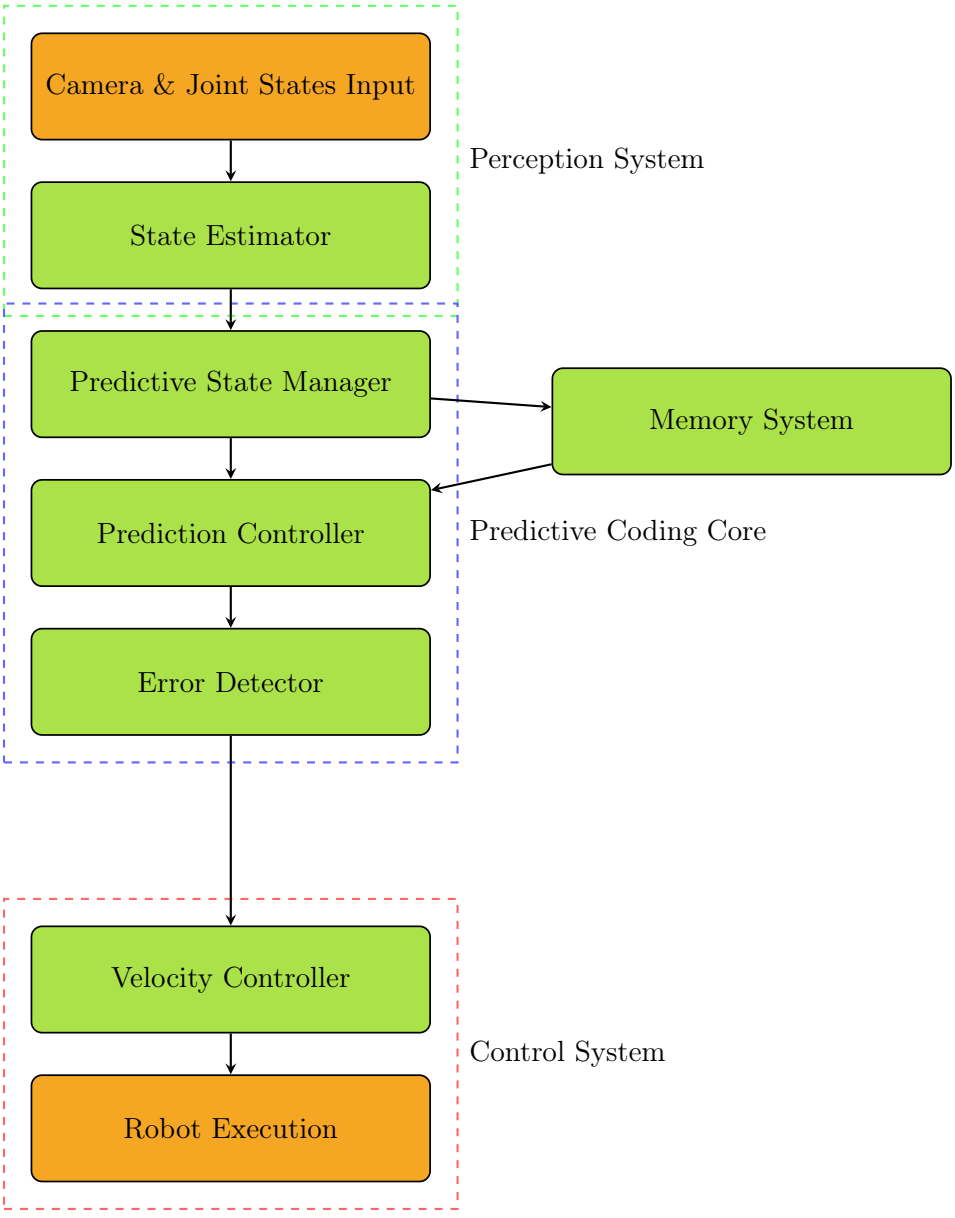
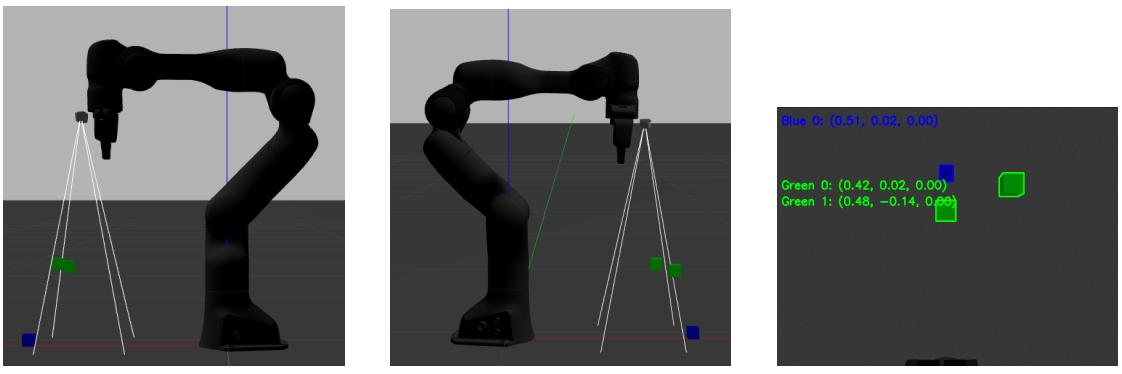


Figure 4.1: Vertical flowchart of the predictive coding system.

5 Results and Analysis

This chapter evaluates the predictive-coding controller in two simulated environments: *(i)* a free world without obstacles and *(ii)* an obstacle world with static impediments between the start pose and the goal. We systematically compare the controller's performance across several key metrics, providing insight into its adaptability and robustness in varying conditions.

This chapter as a whole addresses **Objective 1**: the design and implementation of a hierarchical predictive coding framework that enables multi-horizon prediction and integrates error detection, state estimation, and memory across temporal scales. The following sections evaluate the framework's performance with respect to the remaining objectives.



(a) Starting Position Left View (b) Starting Position Right View (c) Starting Position Camera View

Figure 5.1: The starting position of the arm with respect to the target and obstacles in Gazebo.

We focus on four main aspects:

1. Trajectory performance,
2. Error–minimisation dynamics,
3. Average prediction error at three horizons (0.2, 0.3, 0.5 s), and
4. Hierarchical influence & system confidence.

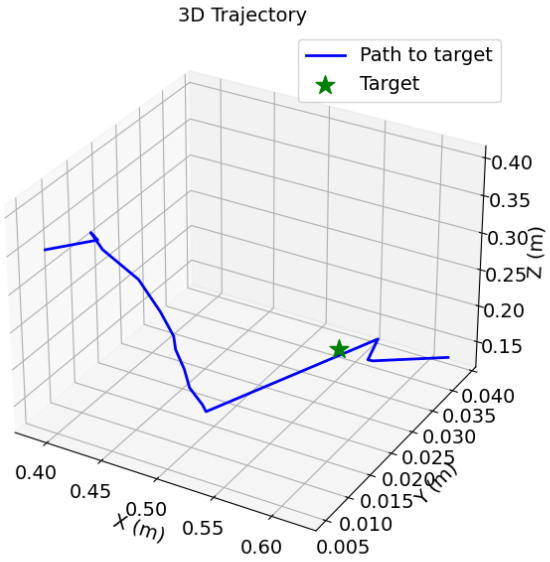
For every metric we juxtapose the two scenarios in paired plots and discuss the differences below each figure.

5.1 Trajectory Performance

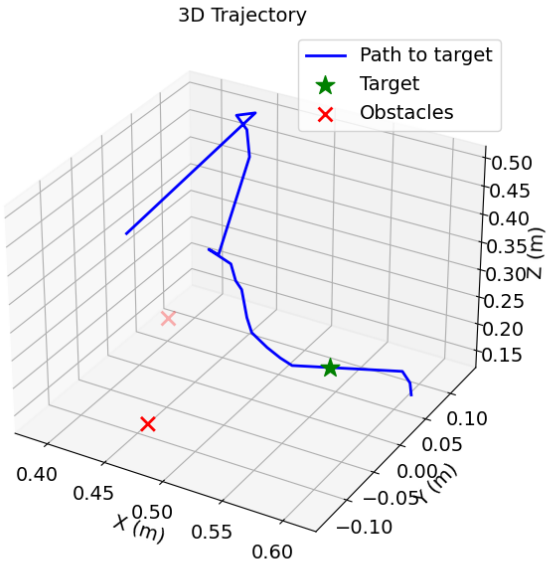
The Trajectory Performance section specifically demonstrates the framework's ability to generate robust, adaptive trajectories in simulated environments.

Trajectory adaptation is a primary indicator of the controller's robustness in dynamic simulated environments. Figure 5.2 compares the end-effector trajectories under both simulated environmental conditions.

In the free world (Fig. 5.2a) the robot arm moves directly towards the target without any obstructions. The path to target is smooth and follows a nearly straight line from the starting position to the target. The absence of obstacles allows the predictive coding controller to optimize for the shortest and most efficient path. With obstacles (Fig. 5.2b) the predictive coding system must adapt its trajectory to avoid collisions while still reaching the target. The trajectory exhibits clear deviations as the arm navigates around obstacles, demonstrating the system's ability to integrate sensory feedback and prediction to generate safe, adaptive movements. The red crosses mark the positions of obstacles, and the path shows smooth avoidance behavior



(a) Free world (no obstacles)



(b) World with obstacles

Figure 5.2: End-effector trajectories under the two environmental conditions. Blue lines denote trajectory paths; the green star marks the target; red crosses in (b) locate obstacles.

before converging on the target.

Key finding: The predictive coding controller enables efficient and adaptive target-reaching behavior in both scenarios, with smooth obstacle avoidance in complex environments.

5.2 Hierarchical Prediction Performance

This section addresses **Objective 4** by evaluating system performance using multi-horizon error analysis and confidence metrics, and **Objective 3** by analyzing confidence estimation mechanisms within the predictive coding framework.

5.2.1 Hierarchical Influence and System Confidence

System confidence, evaluated across multiple prediction horizons, provides insight into the controller's certainty under varying environmental complexity.

Figure 5.3 presents the evolution of system confidence across different prediction horizons for both the obstacle-free and obstacle-rich environments. As expected, the confidence for the obstacle-free scenario (dotted lines) is generally higher and more stable than for the scenario with obstacles (solid lines), reflecting the increased uncertainty and prediction challenges introduced by environmental complexity.

A closer inspection of the curves, however, reveals several important nuances:

Sharp Drop in No-Obstacle Case: In the obstacle-free environment, there is a pronounced drop in system confidence across all horizons at around 10–15 seconds. This drop coincides with a period during which the camera view ratio of the target is low, meaning the target is either partially occluded or outside the field of view as the arm moves. This perceptual limitation directly impacts the system's certainty,

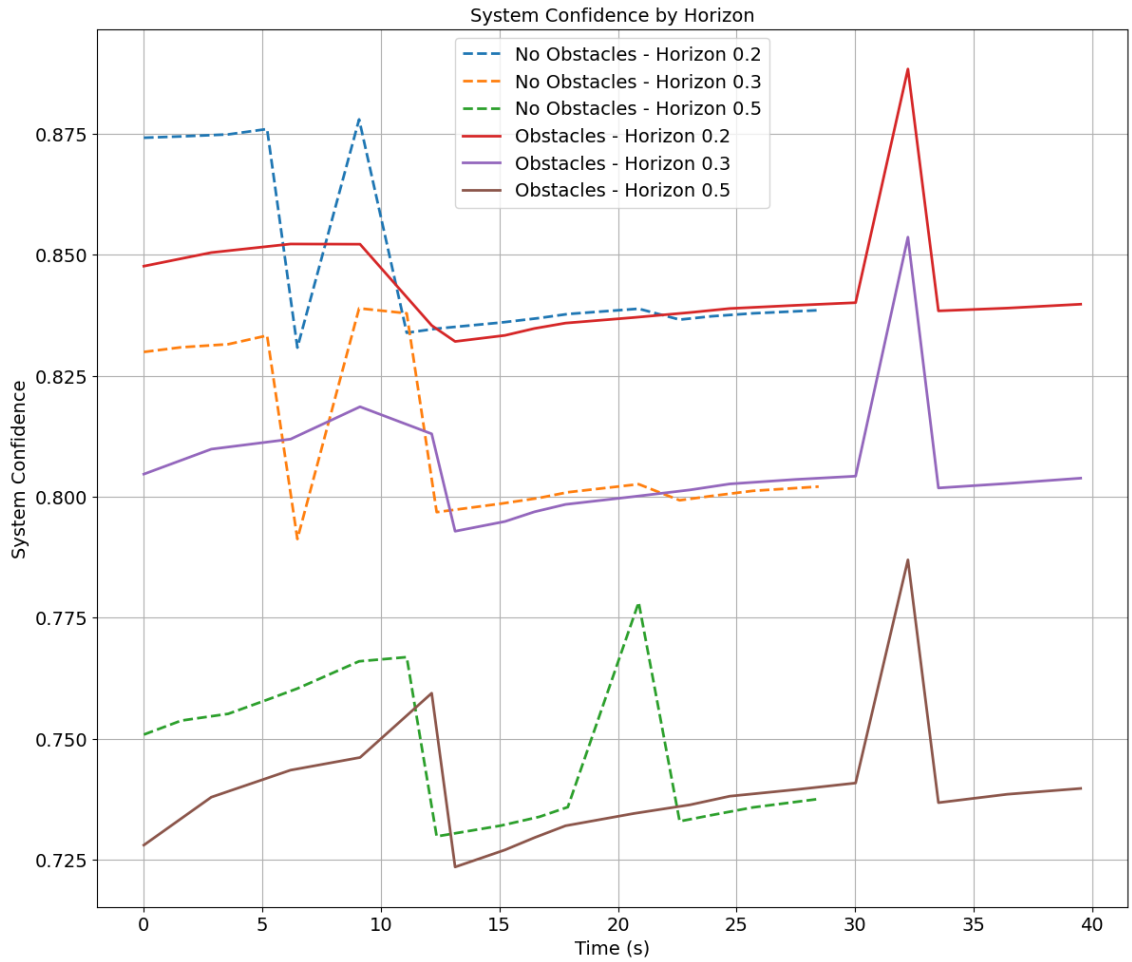


Figure 5.3: Comparison of system confidence by prediction horizon in a free world (no obstacles) and a world with obstacles. Solid lines represent scenarios with obstacles (from top to bottom: 0.2s, 0.3s, 0.5s horizons), while dotted lines represent scenarios with no obstacles (from top to bottom: 0.2s, 0.3s, 0.5s horizons).

demonstrating that confidence is not solely a function of environmental complexity, but is also highly sensitive to the visibility of the target in the camera view.

Sharp Increase in Obstacles Case: In the scenario with obstacles, a sharp increase in system confidence is observed at around 30 seconds. This spike can be attributed to two factors: (1) the camera view ratio of the target becomes high, indicating that the target is clearly visible to the system, and (2) the arm has successfully navigated past the obstacles, so obstacle influence on the system's confidence is minimal at this stage. The combination of clear target visibility and the absence of immediate obstacles allows the system to regain high certainty in its predictions.

General Trends: Overall, the system confidence in the obstacle-free environment remains higher and more stable throughout the episode, while the presence of obstacles introduces more variability and generally lower confidence, especially for longer prediction horizons. This is consistent with the expectation that environmental complexity and prediction horizon both contribute to uncertainty in the system's internal model.

Key finding: System confidence is influenced not only by the presence of obstacles but also by perceptual factors such as target visibility in the camera view. Periods of low confidence can result from occlusion or poor visibility, even in otherwise simple environments, while high confidence can be restored when the target is clearly visible and obstacles have been successfully avoided. This highlights the importance of both environmental and perceptual factors in determining the certainty of predictive robotic control systems.

5.2.2 Average Error By Prediction Horizon

This subsection further addresses **Objective 4**, providing a detailed analysis of predictive accuracy across different horizons.

The Average Error by Prediction Horizon bar chart effectively summarizes the system's predictive performance across different timescales. This combined analysis, shown in Figure 5.4, highlights how prediction accuracy varies with both horizon length and environmental complexity. The results reveal that, for both scenarios,

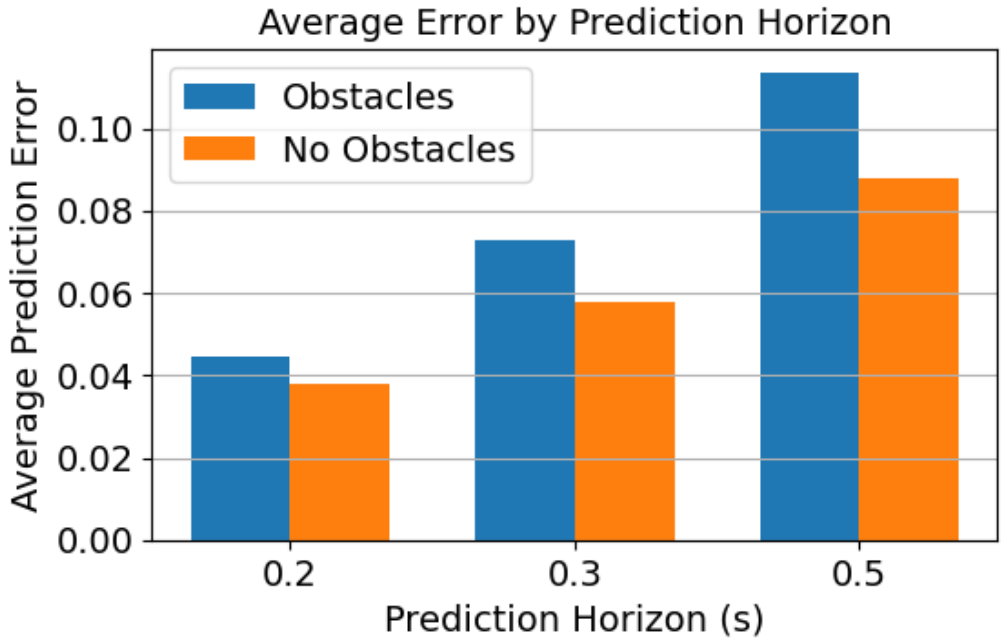


Figure 5.4: Comparison of average prediction error by horizon in a free world (no obstacles) and a world with obstacles. The plot shows the average error for each prediction horizon in both scenarios.

the average prediction error increases as the prediction horizon extends from 0.2s to 0.5s. This trend is expected, as longer horizons naturally introduce greater uncertainty and prediction becomes more challenging. However, the presence of obstacles consistently leads to higher average errors across all horizons, with the difference being most pronounced at the longest horizon (0.5s). This reflects the added difficulty of making accurate predictions in a dynamic, obstacle-rich environment, where un-

expected events and environmental complexity challenge the system's adaptability.

Key finding: The predictive coding system maintains low errors in predictable settings, but its accuracy is impacted by increased uncertainty and complexity, especially at longer horizons.

5.2.3 Memory Influence by Horizon

This subsection addresses **Objective 2**: the development and evaluation of an adaptive memory-augmented learning system and its influence on prediction accuracy across horizons.

The Memory Influence by Horizon graph illustrates how the predictive coding system dynamically leverages the memory module across different temporal horizons throughout the simulation. The results show that, memory influence increases most rapidly for the shortest horizons (0.2s and 0.3s), while remaining minimal for the longest horizon (0.5s). In the obstacle-free environment (dashed lines), the system quickly learns to rely on memory for short- and mid-term predictions, reflecting the abundance of successful experiences in a predictable setting. In contrast, the presence of obstacles (solid lines) leads to a more gradual increase and overall lower memory influence, especially for longer horizons. This reflects the increased challenge of learning from experience in a complex, dynamic environment, where successful outcomes are less frequent and harder to generalize.

Key finding: The predictive coding system is more effective at minimizing error in predictable environments, but remains resilient and adaptive even under challenging conditions with obstacles.

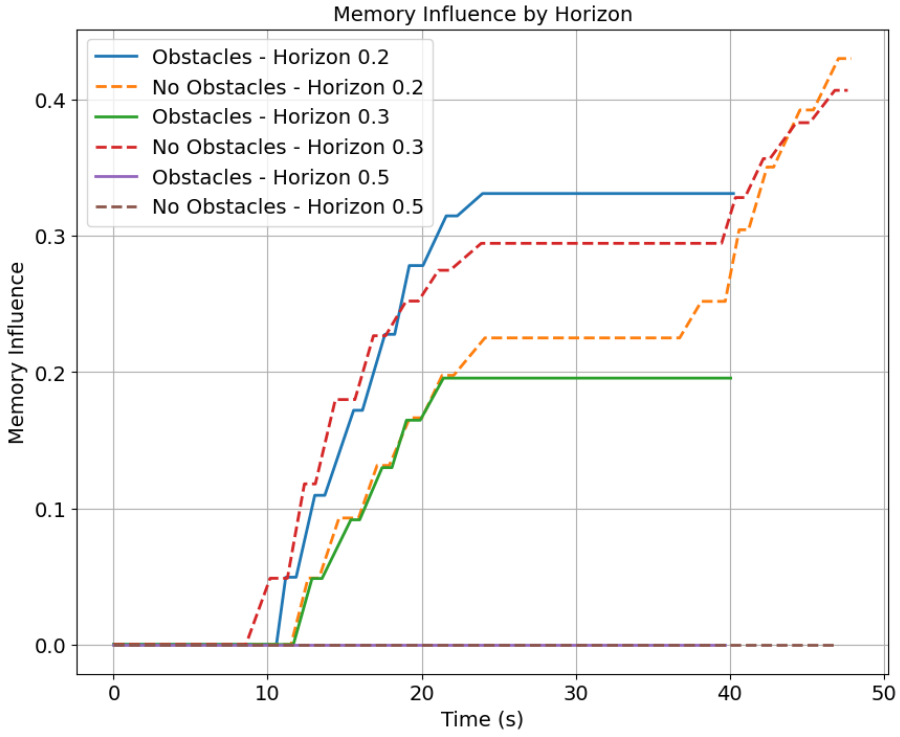


Figure 5.5: Comparison of memory influence by prediction horizon in a free world (no obstacles) and a world with obstacles. Solid lines represent scenarios with obstacles (Horizons 0.2s, 0.3s, 0.5s), while dashed lines represent scenarios with no obstacles (Horizons 0.2s, 0.3s, 0.5s).

5.3 Error Minimization Progress

This section addresses **Objective 3**, focusing on robust error minimization and the system's ability to adapt under uncertainty in simulated environments.

To further evaluate the predictive coding system, we present a direct comparison of its ability to minimize prediction error in the system. This combined analysis, illustrated in Figure 5.6, highlights how environmental uncertainty influences the system's learning and adaptation dynamics. In the obstacle-free scenario, the system demonstrates a rapid and consistent reduction in prediction error, reflecting efficient adaptation in a predictable environment. The error curve for this case shows a steady downward trend, indicating that the predictive coding system quickly learns and stabilizes its predictions when uncertainty is low. In contrast, the presence

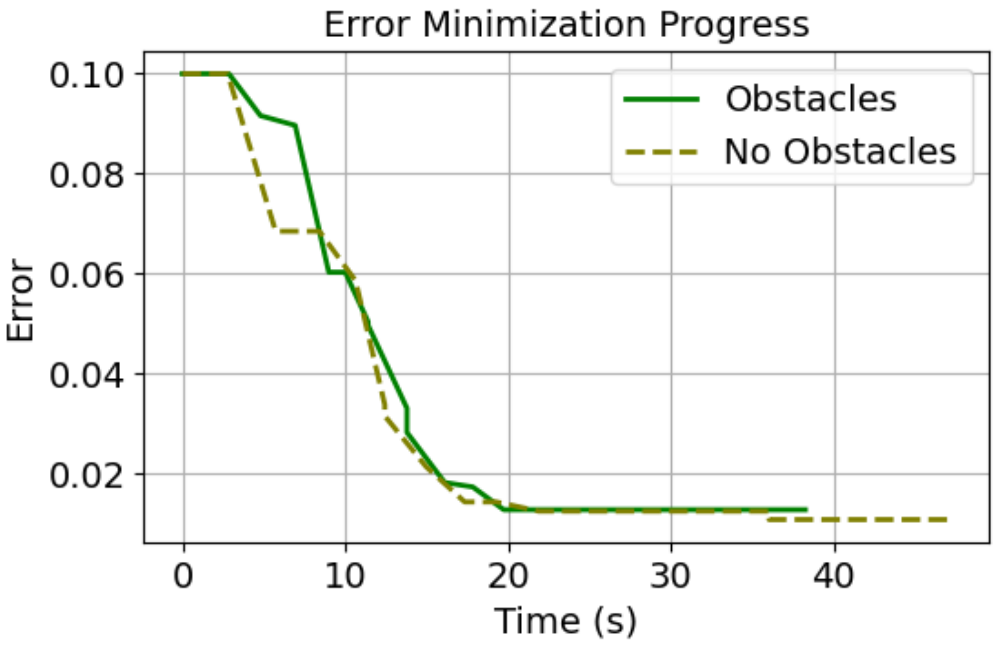


Figure 5.6: Comparison of error minimization progress in a free world (no obstacles) and a world with obstacles. The plot shows the smoothed prediction error over time for both scenarios.

of obstacles introduces higher uncertainty and more frequent prediction challenges. The error curve for the obstacle-rich environment exhibits more fluctuations and occasional plateaus, with a generally slower rate of error reduction. This behavior reflects the increased difficulty of making accurate predictions and adapting in a dynamic, complex setting.

Key finding: The predictive coding system is more effective at minimizing error in predictable environments, but remains resilient and adaptive even under challenging conditions with obstacles.

Summary of Key Findings

This chapter demonstrates that the predictive-coding controller achieves robust and adaptive performance across a range of metrics in simulation. In predictable, obstacle-free simulated environments, the system rapidly minimizes error, maintains

high confidence, and effectively leverages memory for accurate prediction and control. The introduction of obstacles increases uncertainty and prediction challenges, leading to higher errors, lower confidence, and reduced memory influence, particularly at longer horizons. Nevertheless, the controller remains resilient, adapting its trajectory and continuing to minimize error even in complex, dynamic simulated settings. These findings demonstrate the effectiveness of the predictive-coding controller in simulation. Future work will focus on transferring and validating these results in real-world robotic systems.

6 Conclusion and Future Work

6.1 Conclusion

This thesis presents a hierarchical predictive coding framework for the Franka Emika Panda robotic arm, demonstrating significant improvements in adaptability and learning within simulated environments. Motivated by the need for more adaptive and robust robotic control, the work addresses key challenges in multi-horizon prediction, memory integration, and confidence estimation. By integrating a forward predictor that generates multi-horizon predictions (at 0.2s, 0.3s, and 0.5s), applying hierarchical constraints for temporal consistency, and leveraging an adaptive memory module, the system achieves efficient and resilient control in both obstacle-free and obstacle-rich scenarios.

All results and validations were conducted in simulation, providing a controlled and repeatable environment to rigorously assess system performance. The controller demonstrated rapid error minimization, high confidence in predictions, and effective use of memory for adaptation, even as environmental complexity increased. These findings confirm the feasibility and effectiveness of predictive coding for advanced robotic control, and establish a strong foundation for future real-world deployment and research.

By systematically addressing these objectives, the thesis advances the state of the art in simulation-based robotic control and offers a practical framework for ongoing innovation in adaptive, learning-based robotic systems.

6.2 Limitations and Future Work

While the system demonstrates substantial capabilities, several limitations have been identified, each presenting a clear avenue for future research and development:

6.2.1 Testing Environment Limitations

- **Limitation**: Validation was conducted solely in simulation environments (ROS and Gazebo), without deployment on physical hardware.
- **Future Work**: Implement and validate the control system on the physical Franka Emika Panda robot to assess robustness under real-world uncertainties and noise.

6.2.2 Memory System Constraints

- **Limitation**: The memory system uses a fixed capacity (1000 experiences) and static retrieval mechanisms, potentially limiting scalability and adaptability.
- **Future Work**: Develop dynamic memory capacity management and implement advanced experience clustering algorithms to enhance long-term learning.

6.2.3 Temporal Horizon Restrictions

- **Limitation**: The system operates with fixed prediction horizons (0.2 s, 0.3 s, 0.5 s), with limited flexibility in adjusting based on context.

- **Future Work**: Enable adaptive horizon selection based on task phase, environmental complexity, and uncertainty levels; enhance inter-horizon coordination mechanisms.

6.2.4 Environmental Interaction

- **Limitation**: The current implementation is restricted to static obstacle scenarios with fixed collision thresholds (e.g., 0.1 m).

- **Future Work**: Extend the framework to handle dynamic obstacle avoidance and develop adaptive safety margins that respond to real-time environmental changes.

6.2.5 Performance Optimization

- **Limitation**: Hyperparameters such as learning rates and confidence thresholds were manually tuned, limiting adaptability across tasks.

- **Future Work**: Introduce automated hyperparameter tuning methods and adaptive learning strategies to improve performance across diverse scenarios.

These limitations provide a clear roadmap for extending the current framework. They not only highlight areas for refinement but also confirm the system's foundational effectiveness. This work lays the groundwork for further advancement of predictive coding in robotic control, contributing both a practical implementation and a research platform for ongoing innovation.

This work paves the way for more intelligent, adaptive robots capable of thriving in uncertain and dynamic environments. The proposed framework could be extended

to other robotic platforms or real-world tasks requiring adaptive, predictive control, contributing to the broader advancement of autonomous robotics.

References

[1] K. Friston, "The free-energy principle: A unified brain theory?", *Nature Reviews Neuroscience*, vol. 11, no. 2, pp. 127–138, 2010, ISSN: 1471-0048. DOI: 10.1038/nrn2787. [Online]. Available: https://doi.org/10.1038/nrn2787.

[2] L. Wang et al., "Predictive coding across the left fronto-temporal hierarchy during language comprehension", *Cerebral Cortex*, vol. 33, no. 8, pp. 4478–4497, Sep. 2022, ISSN: 1047-3211. DOI: 10.1093/cercor/bhac356. eprint: https://academic.oup.com/cercor/article-pdf/33/8/4478/49741508/bhac356.pdf. [Online]. Available: https://doi.org/10.1093/cercor/bhac356.

[3] M. W. Spratling, "A review of predictive coding algorithms", *Brain and cognition*, vol. 112, pp. 92–97, 2017.

[4] B. Millidge, A. Seth, and C. L. Buckley, *Predictive coding: A theoretical and experimental review*, 2022. arXiv: 2107.12979 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2107.12979.

[5] C. Meo, G. Franzese, C. Pezzato, M. Spahn, and P. Lanillos, "Adaptation through prediction: Multisensory active inference torque control", *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 1, pp. 32–41, 2023. DOI: 10.1109/TCDS.2022.3156664.

[6] A. Ciria, G. Schillaci, G. Pezzulo, V. V. Hafner, and B. Lara, *Predictive processing in cognitive robotics: A review*, 2021. arXiv: 2101.06611 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2101.06611.

[7] M. Baltieri and C. L. Buckley, "Pid control as a process of active inference with linear generative models", *Entropy*, vol. 21, no. 3, p. 257, 2019. DOI: 10.3390/e21030257. [Online]. Available: https://doi.org/10.3390/e21030257.

[8] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control", *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 2002.

[9] D. Kragic, M. Björkman, H. I. Christensen, and J.-O. Eklundh, "Vision for robotic object manipulation in domestic settings", *Robotics and autonomous Systems*, vol. 52, no. 1, pp. 85–100, 2005.

[10] J. Tani, *Exploring robotic minds: actions, symbols, and consciousness as self-organizing dynamic phenomena*. Oxford University Press, 2016.

[11] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges", *Robotics*, vol. 2, no. 3, pp. 122–148, 2013, ISSN: 2218-6581. DOI: 10.3390/robotics2030122. [Online]. Available: https://www.mdpi.com/2218-6581/2/3/122.

[12] G. Maffei, D. Santos-Pata, E. Marcos, M. Sánchez-Fibla, and P. F. Verschure, "An embodied biologically constrained model of foraging: From classical and operant conditioning to adaptive real-world behavior in dac-x", *Neural Networks*, vol. 72, pp. 88–108, 2015, Neurobiologically Inspired Robotics: Enhanced Autonomy through Neuromorphic Cognition, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2015.10.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608015002026.

[13] M. W. Spong, S. Hutchinson, M. Vidyasagar, et al., *Robot modeling and control*. Wiley New York, 2006, vol. 3.

[14] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", in *Proceedings. 1985 IEEE international conference on robotics and automation*, IEEE, vol. 2, 1985, pp. 500–505.

[15] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world", in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2017, pp. 23–30.

[16] A. Clark, "Whatever next? predictive brains, situated agents, and the future of cognitive science", *Behavioral and Brain Sciences*, vol. 36, no. 3, pp. 181–204, 2013. DOI: 10.1017/S0140525X12000477.

[17] R. P. N. Rao and D. H. Ballard, "Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects", *Nature Neuroscience*, vol. 2, no. 1, pp. 79–87, 1999, ISSN: 1546-1726. DOI: 10.1038/4580. [Online]. Available: https://doi.org/10.1038/4580.

[18] R. Bogacz, "A tutorial on the free-energy framework for modelling perception and learning", *Journal of Mathematical Psychology*, vol. 76, pp. 198–211, 2017.

[19] T. S. Lee and D. Mumford, "Hierarchical bayesian inference in the visual cortex", *Journal of the Optical Society of America A*, vol. 20, no. 7, pp. 1434–1448, 2003.

[20] J. Hwang, J. Kim, A. Ahmadi, M. Choi, and J. Tani, "Dealing with large-scale spatio-temporal patterns in imitative interaction between a robot and a human by using the predictive coding framework", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 5, pp. 1918–1931, 2020. DOI: 10.1109/TSMC.2018.2791984.

[21] J. Tani and S. Nolfi, "Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems", *Neural Networks*, vol. 12, no. 7, pp. 1131–1141, 1999, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(99)00060-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360809900060X.

[22] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment", *PLoS computational biology*, vol. 4, no. 11, e1000220, 2008.

[23] M. W. Spratling, "Reconciling predictive coding and biased competition models of cortical function", *Frontiers in computational neuroscience*, vol. 2, p. 300, 2008.

[24] D. W. Franklin and D. M. Wolpert, "Computational mechanisms of sensorimotor control", *Neuron*, vol. 72, no. 3, pp. 425–442, 2011.

[25] P. Lanillos et al., *Active inference in robotics and artificial agents: Survey and challenges*, 2021. arXiv: 2112.01871 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2112.01871.

[26] D. C. Knill and A. Pouget, "The bayesian brain: The role of uncertainty in neural coding and computation", *TRENDS in Neurosciences*, vol. 27, no. 12, pp. 712–719, 2004.

[27] S. Deneve, "Bayesian spiking neurons i: Inference", *Neural computation*, vol. 20, no. 1, pp. 91–117, 2008.

[28] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2024.

[29] A. A. Meera and M. Wisse, "Free energy principle based state and input observer design for linear systems with colored noise", pp. 5052–5058, 2020. DOI: 10.23919/ACC45564.2020.9147581.

[30] R. A. Adams, S. Shipp, and K. J. Friston, "Predictions not commands: Active inference in the motor system", *Brain Structure and Function*, vol. 218, no. 3, pp. 611–643, 2013.

[31] A. Jimoh, A. R. Zarei, J. Plosila, and H. Haghbayan, "Implementation of a pick-and-place robotic system using franka arm and vision-based object detection", Turku University, Technical Report, Feb. 2025. [Online]. Available: https://urn.fi/URN:ISBN:978-952-02-0065-7.

[32] G. Oliver, P. Lanillos, and G. Cheng, "An empirical study of active inference on a humanoid robot", *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 2, pp. 462–471, Jun. 2022, ISSN: 2379-8939. DOI: 10.1109/tcds.2021.3049907. [Online]. Available: http://dx.doi.org/10.1109/TCDS.2021.3049907.

[33] C. Pezzato, R. Ferrari, and C. H. Corbato, "A novel adaptive controller for robot manipulators based on active inference", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2973–2980, Apr. 2020, ISSN: 2377-3774. DOI: 10.1109/lra.2020.2974451. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.2974451.

[34] C. Sancaktar, M. A. J. van Gerven, and P. Lanillos, "End-to-end pixel-based deep active inference for body perception and action", in *2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, IEEE, Oct. 2020, pp. 1–8. DOI: 10.1109/icdl-epirob48136.2020.9278105. [Online]. Available: http://dx.doi.org/10.1109/ICDL-EpiRob48136.2020.9278105.

[35] B. Millidge, A. Tschantz, and C. L. Buckley, "Predictive coding approximates backprop along arbitrary computation graphs", *Neural Computation*, vol. 34, no. 6, pp. 1329–1368, May 2022, ISSN: 0899-7667. DOI: 10.1162/neco_a_

01497. eprint: https://direct.mit.edu/neco/article-pdf/34/6/1329/2023477/neco_a_01497.pdf. [Online]. Available: https://doi.org/10.1162/neco%5C_a%5C_01497.

[36] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[37] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking", in *European conference on computer vision*, Springer, 2020, pp. 107–122.

[38] F. Chaumette and S. Hutchinson, "Visual servoing and visual tracking", *Handbook of Robotics*, pp. 563–583, 2008.

[39] Franka Emika. "Franka emika robot arm - overview". Accessed: 2025. [Online]. Available: https://frankaemika.github.io/docs/overview.html.

[40] Intel RealSense. "D435 depth camera". Accessed: 2025.