Enhancing the Path Accuracy of an Industrial Robot with Visual Servoing and Reinforcement Learning

Tao Zhou

A Thesis

in

The Department

of

Mechanical, Industrial & Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

September 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Tao Zhou**

Entitled: **Enhancing the Path Accuracy of an Industrial Robot with Visual Servoing and Reinforcement Learning**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Youmin Zhang

_____ External Examiner
Dr. Jamal Bentahar

_____ Examiner
Dr. Youmin Zhang

_____ Supervisor
Dr. Wen-Fang Xie

Approved by _____
Martin D. Pugh, Chair
Department of Mechanical, Industrial & Aerospace Engineering

_____ 2022

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Enhancing the Path Accuracy of an Industrial Robot with Visual Servoing and Reinforcement Learning

Tao Zhou

Industrial robots generally exhibit poor path accuracy and thus cannot satisfy many manufacturing requirements. Improving robot path accuracy necessitates motion control with feedback from high-precision external sensors. Typically, feedback control strategies for an industrial robot are realized by adjusting the robot's intrinsic motion via an external controller.

In this research, the control scheme consists of position-based visual servoing (PBVS). An external pose correction controller is introduced to enhance a robot's path accuracy by reducing the path error.

A proportional-integral-derivative (PID) controller is utilized as the primary control method. Due to the repetitiveness of robotic tasks, the control performance can undergo iterative improvement by supplementing the baseline PID controller with a reinforcement learning (RL) based controller, trained via an actor-critic algorithm. The experimental platform comprises two commercial systems: the C-Track 780 dual camera sensor (Creaform) and the M-20iA robot (FANUC). The control performance is assessed using mean absolute error (MAE) and maximum error.

The effect of supplementing the baseline PID controller with the proposed RL-based controller is assessed with two experiments. Experiment 1 consists of position-only line following, while experiment 2 consists of full pose line following. Qualitatively, transient

performance is improved by overshoot attenuation. In experiment 1, the RL-based controller reduces MAE and maximum errors by 10% and 20%, respectively in the transient phase. The resulting *position* path accuracy is ± 0.09 mm. In the (full pose) experiment, the MAE is not significantly affected, where the maximum errors deteriorate by 4% in *position* but improve by 9% in *orientation*. The resulting path accuracy is ± 0.30 mm / ± 0.07 °.

Acknowledgments

First, I express my deepest gratitude to my supervisor, Professor Wen-Fang Xie. Her superb guidance was shown in her terrific insight into the research subject, empathy towards others, and eagerness and timeliness in responding to students. She helped consolidate my research path and provided invaluable feedback in refining this thesis. Needless to say, this research would not be possible without her expert input.

Much of the experimental work was done in collaboration with Jianyu Tang. I am indebted to Jianyu for his immense talent for cooperation, camaraderie, and enlightening conversations on research and life.

I thank Tingting Shu for her instrumental support to help kickstart my research and implement the experiments. I also thank Dr. Henghua Shen for providing feedback benefiting the quality of the thesis and for his generosity in involving me in one of his research projects. I am grateful to Ehsan Zakeri for his enthusiasm in sharing his vast knowledge, which helped me overcome specific research difficulties and improve my thesis. Grinding in the laboratory would not be as enjoyable without the company of Alireza Saboukhi and Tzvi Filler. I thank Marizeh Masoodi Nia for giving me an introduction to RoboDK.

I thank all my colleagues in Professor Xie's group: Bahar Ahmadi, Alex Bryce, Ibrahim Babiker, Tzvi Filler, Marizeh Masoodi Nia, Alireza Saboukhi, Henghua Shen, Tingting Shu, Jianyu Tang, Bingze Xia, Ehsan Zakeri, and Ningyu Zhu.

The author is supported by NSERC Canada Graduate Scholarship Master's Award.

To my beloved parents and brother

Liyan**, **Yaojin** and **Victor

Contents

List of Figures

List of Tables

Nomenclature

kNN k-Nearest Neighbour

ADRC Active Disturbance Rejection Control

AFP Automatic Fiber Placement

API Application Programming Interface

CMM Coordinate Measurement Machine

DPM Dynamic Path Modification

ECL End-Point Closed Loop

EOL End-Point Open Loop

GPI General Policy Iteration

IIC Initial Initialization Condition

ILC Iterative Learning Control

MAE Mean Absolute Error

MBPO Model-Based Policy Optimization

MDP Markov Decision Process

MPC Model Predictive Control

PBVS Position-Based Visual Servoing

PD Proportional-Derivative

PI Proportional-Integral

PID Proportional-Integral-Derivative

Q-Q Quantile-Quantile

RKF Robust Kalman Filter

RL Reinforcement Learning

RMSE Root Mean Square Error

TD3 Twin Delayed Deep Deterministic Policy Gradient

VS Visual Servoing

Chapter 1

Introduction

This research is motivated by industrial robots' lack of accuracy, which hinders their potential use in advanced industrial applications. The chosen approach to enhance path accuracy is real-time control with visual feedback, i.e., visual servoing. The rest of this chapter will elaborate on related works, background knowledge, contributions, and an outline of the thesis.

1.1 Motivation

Industrial robots offer the advantage of adaptability and versatility, eliminating the need to develop custom machines for specific manufacturing and automation processes. For instance, in aerospace, robots have a wide range of applications, as shown in Figure 1.1, such as welding, spray painting, palletizing, assembling, riveting, etc. In most use cases, the robot must be programmed online using the teach pendant, i.e., the programmer has to jog the robot to a sequence of desired poses corresponding to the motion of the required task, where each pose is *taught* to the robot controller. In this operation paradigm, only robot repeatability matters, while accuracy does not. Repeatability typically ranges from 0.005 to 0.1 mm [7]. Due to the open-loop nature of online programming, even little alterations

in the task or work environment can induce significant performance deterioration. Consequently, frequent reprogramming of the robot is often necessary, resulting in high time and capital expenses.

Moreover, online programming becomes infeasible for complicated paths due to the high amount of points needed to be taught. Ideally, robot tasks should be able to be programmed offline, i.e. the programmer defines the robot motion numerically in a virtual environment with the aid of robot simulation software. However, this offline approach often suffers from the inherent poor accuracy of industrial robots. Typically, there is an 8-15 mm offset between virtual and real robots [8]. Indeed, modern industrial robots usually exhibit poor accuracy relative to their repeatability. With an appropriate pose correction mechanism, industrial robots can theoretically reach a level of accuracy close to their repeatability. Accuracy is essential for applications requiring precise positioning or path following. For example, robots can be used to accurately position patients for particle beam disease treatment [9]. High path accuracy is necessary when robots are used for automatic fibre placement to lay fibre composites evenly, for spray painting to prevent the creation of scars or bubbles, or for aviation drilling to ensure the longevity of aircraft assembly [10, 11, 12]. Furthermore, aerospace manufacturing typically necessitates a robot accuracy of 0.20 mm [13]. Thus, one would like to increase the accuracy of industrial robots.

Much development has been made on robot calibration methods targeting positioning accuracy (*static accuracy*) as defined by the ISO 9283 norm [14]. Most robot calibration research focuses mainly on refining mathematical models to better represent the actual kinematics of the robots than the nominal models provided by the manufacturers. Robot inaccuracy stems from five major categories of error: environmental, parametric, dynamic, measurement, computational, and application [15, 16]. One representative study on robot calibration reported an improvement in the absolute accuracy of an ABB IRB 1600 industrial robot where the mean/max position errors were reduced from 0.968/2.158 mm to

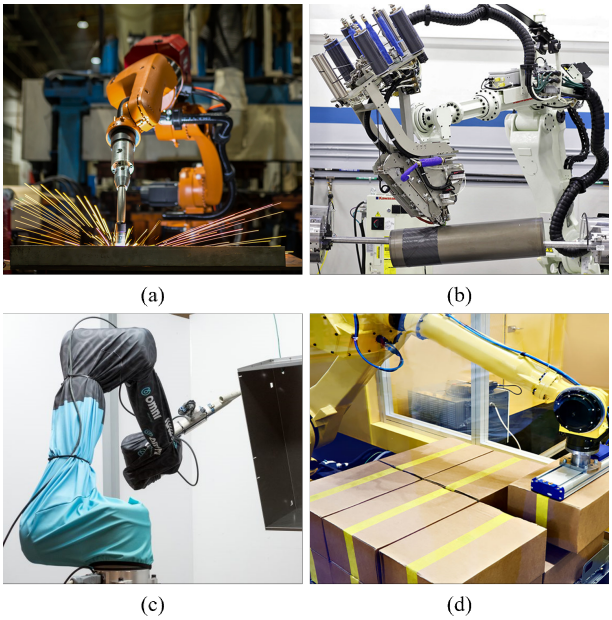


Figure 1.1: Applications of industrial Robots; (a) welding, (b) automatic fiber placement (AFP), (c) spray painting, (d) palletizing (adapted from [1], [2], [3], and [4]).

0.364/0.696 mm respectively [17]. This improvement is satisfactory for many applications that tolerate positioning errors of up to 0.50 mm. Further improvement, while theoretically possible, is hard to achieve practically due to the difficulty of modelling the dynamic and thermal effects [17]. Moreover, robots must also be calibrated frequently due to ongoing damage from their normal operation and changes in their operating environment.

Robot calibration only improves static positioning but has limited effect on the path accuracy [7]. Path accuracy refers to the accuracy of the robot's motion when following a predefined path. This shortcoming is not an issue for applications requiring only accurate static positioning. However, it represents a bottleneck for advanced tasks such as automatic fibre placement, where the robot has to move precisely around parts of complicated topology [10]. Thus, there are still considerable incentives to extend the viability of industrial robots in high precision path following applications. Moreover, being an open-loop approach, robot calibration is highly sensitive to any aberration to the robot's operation conditions and, therefore, must be performed frequently to maintain its effectiveness. One

solution to improve path accuracy is the real-time feedback control with an external sensor of superior accuracy to that of the internal sensor of the robot. Such a closed-loop approach can be made robust to handle uncertain operating conditions. Among possible sensors, cameras and laser trackers are widely considered since they do not physically interfere with the robot. Indeed, the usage of optical and laser sensors in the calibration process of industrial robots has been reported [18]. The usage of visual feedback signal in control is often referred to as *visual servoing*. Visual servoing has been adopted to enhance both the pose and path accuracy of industrial robots [19, 13].

Regarding control algorithms, the few studies on feedback-based enhancement of robot accuracy have mostly proposed variations of the proportional-integral-derivative (PID) controller [19, 13, 7]. Although the PID controller is a proven model-free method, it does not take advantage of the episodic nature of most robotic tasks. Industrial robots often perform many repetitions of the same task, allowing learning and improvement from previous experience. Therefore, there is an incentive to investigate learning-based control methods to enhance path accuracy. One such learning-based method is iterative learning control (ILC), which was first proposed to improve robotic control performance [20]. Alternatively, deep reinforcement learning is the emergent approach for robotic control optimization based on real-world experience [21].

Furthermore, the advent of industry 4.0 will shift mass production towards more personalized manufacturing. Instead of just repeating a singular task in a production line, industrial robots would need to perform various tasks in various work cell configurations while maintaining high accuracy. This new operation paradigm requires the motion controller of the industrial robots to be task-agnostic, i.e., it must have the capability to self-optimize in real-time regardless of the given task. A typical robot's controller is usually task-specific, i.e., slight alterations in the task or environment can lead to the costly recalibration of the robot or re-tuning of its controller. Although a perfect task-agnostic does not exist, this

research strives to develop a control algorithm that is *smart* enough to adapt to different robotic tasks of the same order of complexity.

As highlighted above, potential industrial applications reliant on offline programming are hindered by the general lack of accuracy of industrial robots. Additionally, robot calibration, the current go-to robot accuracy enhancement solution, is quite imperfect as it does not affect path accuracy, which is the limiting factor in many potential robotic applications. The aforementioned challenges motivate the exploration of various strategies to enhance the path accuracy of industrial robots. Since optical sensors are selected as the feedback source, the exploration space has been confined to visual servoing strategies. Furthermore, only learning-based control methods are considered to enable the iterative improvement of repetitive robotic tasks. Concretely, this research aims to *develop learning-based control algorithms, applied to a visual servoing robotic system, to improve path accuracy*.

1.2 Related Work

As discussed in Section 1.1, this research aims to enhance the path accuracy of industrial robots. In the scope of this research, the means to achieve such a goal are refined to visual servoing, and model-free control methods. Recently, these areas of research have been quite active. Below is a summary of the papers which are highly influential to the experimental setup and control algorithms used in this research.

The experimental setup of this research is inherited from two previous works from the same laboratory. [19] utilized the visual feedback of an optical sensor to perform real-time pose correction of industrial robots. The optical sensor in question is the C-Track 780 from Creaform, and the industrial robots are the FANUC LR Mate 200iC and M-20iA robots. With the above hardware, experiments have been able to achieve a *positioning* pose accuracy of $\pm$ 0.050 mm and $\pm$ 0.050° for position and orientation, respectively. The underlying

control algorithm consists of the PID controller. While the improvement in positioning accuracy is remarkable, the proposed approach does not deal with path accuracy, i.e., the accuracy of motion along a path. Nonetheless, the aforementioned research paper laid the groundwork for the experimental setup of this current research work. [13] addressed the *path accuracy* in a direct extension of the aforementioned work. A dynamic path tracking scheme was proposed to correct the motion of industrial robots in real time. The control algorithm consists of the proportional-integral (PI) controller implemented in discrete time. With the same previously mentioned hardware, experiments showed that the *path following* accuracy can reach $\pm$ 0.20 mm for position and $\pm$ 0.10° for both linear and circular paths. The above experimental result is the benchmark this current research strives to improve. Compared to the benchmark study, this research utilized the same experimental setup and control architecture but improved control algorithms.

Regarding control algorithms, they can be classified into model-based and model-free. This research focuses solely on model-free methods due to the difficulty of accurately modelling a system characterized by an industrial robot working in a dynamic and stochastic environment. Among model-free methods, learning-based methods are particularly suitable for optimizing the control of robotics tasks that are often repetitive. One well-known method is iterative learning control (ILC) [22]. ILC can significantly improve tracking performance by mitigating aberrations reoccurring at each iteration, and such is the case of work by [23]. However, ILC can not handle random disturbances and is subjected to the requirement of identical initialization conditions (IIC) in every iteration.

Another more versatile learning paradigm is reinforcement learning (RL) which has achieved unprecedented capabilities in virtual environments. However, RL has yet to see much use in real physical applications. One important case is the work of [24], which introduced two reinforcement learning-based compensation methods to optimize the control performance of industrial robots. The compensation signal is added to the existing

nominal control signal to optimize tracking accuracy. Experiments showed that the two reinforcement learning compensation methods yield favourable performance compared to proportional-derivative (PD) control, model predictive control (MPC) and ILC. This successful application of reinforcement learning in a real industrial robot provided confidence in this research to propose an RL-based method as the control algorithm. However, the above study implemented an RL-based compensator separately for each joint. This decoupling neglects possible significant dynamic or kinematic interactions between joints. Another weakness in the proposed method is that the feedback signal is provided in the joint space, whereas accuracy matters more in Cartesian space for most applications. Hence, inferring the Cartesian error requires the inverse kinematics, whose accuracy can be affected by aberrations or uncertainties in the kinematic model. Ideally, the feedback signal should be given in Cartesian space. Moreover, the policy (controller) is parameterized by a linear combination of radial basis functions. Such a function approximator might not perform better than a deep neural network, the current state-of-the-art.

Other than learning-based methods, there are also methods which are extensions of PID control, such as active disturbance rejection control (ADRC). [7] proposed the use of ADRC to correct industrial robots based on the distance feedback of a linear transducer. The experiment showed a maximum radial error of less than 0.100 mm for a circular path tracking task and a mean error of 0.015 mm. However, PID-based controllers often require careful tuning of their parameters to yield good performance. In a real physical system, tuning PID parameters is often done by procedural trial and error methods, which can be tedious. Moreover, the tuning process might need to be repeated periodically to compensate for the inevitable change in the system or environment. Ideally, the controller should need to be tuned once and be *smart* enough to self-optimize according to the operational data.

1.3 Background

1.3.1 Visual Servoing

Visual servoing usually refers to the closed-loop position control of the robot end-effector using the feedback of visual information. It comes with different varieties based on the following classification [25]:

- **Camera configuration:** Differentiation is often made on whether the camera is mounted on the robot end-effector (*eye-in-hand*) or fixed in workspace (*eye-to-hand*).
- **Error signal:** The feedback error signal can be expressed in image space in the form of image features or pixels and fed directly to the *image-based* controller. Alternatively, the feedback error signal can be given in terms of the target object's pose, estimated based on features extracted from the image, and then fed to the *position-based* controller.
- **Control hierarchy:** Most of the time, the control structure is hierarchical. At the low level, industrial robots have an internal join-level controller receiving feedback from joint encoder readings. At the high level, the visual servo controller provides the reference signal to the low-level controller. This hierarchical control architecture is called the *dynamic look and move* system. In the case of the robot's internal controller being replaced entirely by the visual servo controller, the control system relies entirely on vision, referred to as *direct visual servoing*.
- **Observation of end effector:** Systems, where only the target object is visually observed, are referred to as *end-point open loop* (EOL) systems. Systems where both the target object and the robot end effector are observed are referred to as *end-point closed loop* (ECL) systems.

Based on the above taxonomy, the visual servoing system, studied in this research, is considered an *eye-to-hand*, *position-based*, *dynamic look and move*, *end-point closed loop* visual servoing system. It should be referred to as *position-based visual servoing* (PBVS) for brevity's sake. In this research, PBVS is selected due to the availability of an optical sensor integrated with off-the-shelve pose estimation and a robot option to correct end effector pose based on Cartesian offsets. Such a system is illustrated by the block diagram shown in Figure 1.2. The robot comprises a path planner and a joint-level feedback controller. The path planner takes the reference signal in Cartesian space and converts it to the joint space signal using inverse kinematics. The camera extracts the image features on the robot end effector, which are processed to yield its estimated pose. Based on the robot's estimated pose, the Cartesian controller computes the control signal according to the desired control objective.

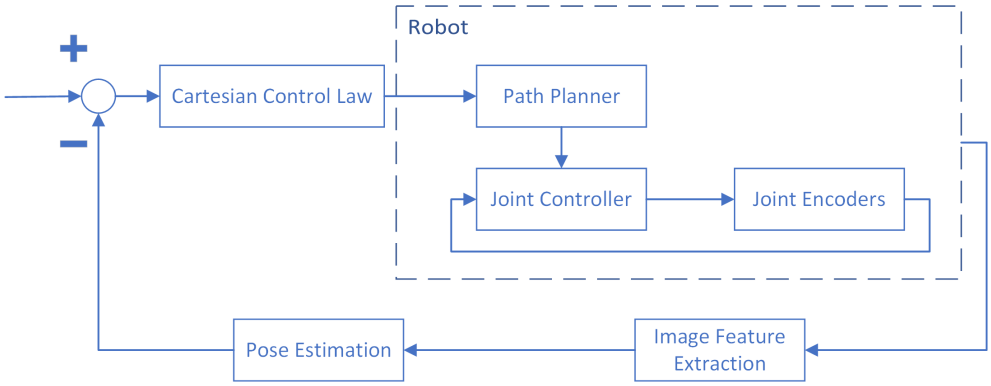


Figure 1.2: PBVS control architecture.

Position-based control neatly separates the problems of motion control and pose estimation from visual data. This research concerns motion control only, focusing on the *positioning task* as defined by [25].

Definition 1.3.1. Consider a kinematic error function $\mathbf{E}(\mathbf{x}_e, \mathcal{P}) : \mathcal{T} \rightarrow \mathbb{R}^m$ where $\mathcal{T}$ represents the robot workspace with d degrees of freedom, m the robot's degree of freedom, $\mathbf{x}_e$ the robot end effector pose, and $\mathcal{P}$ the reference path. The positioning task consists of

controlling $\mathbf{x}_e$ such that $\mathbf{E}(\mathbf{x}_e, \mathcal{P}) = \mathbf{0}$.

In general, the kinematic error function applies a constraint to the degree of freedom of the robot as $d \leq m$. The design of the kinematic error function depends on the given robot task. This research concerns mainly with the task of *point-to-line positioning*, i.e., some point $\mathbf{p}_e$ on the end effector is to be guided to the line defined by two fixed points $\mathbf{s}_1$ and $\mathbf{s}_2$. Here, the error function represents the shortest path to move point $\mathbf{p}_e$ to the line. This shortest path is perpendicular to the line. The resulting path error $\mathbf{E}_{pl}$ is expressed in $\mathbb{R}^3$, given by

$$\mathbf{E}_{pl}(\mathbf{p}_e, \mathbf{s}_1, \mathbf{s}_2) = (\mathbf{s}_1 - \mathbf{p}_e) - ((\mathbf{s}_1 - \mathbf{p}_e) \cdot \mathbf{n})\, \mathbf{n} \tag{1.1}$$

where $\mathbf{n} = (\mathbf{s_2} - \mathbf{s_1})/||\mathbf{s_2} - \mathbf{s_1}||$ and $\cdot$ denotes the dot product. If the orientation of the end effector, expressed in rotation matrix $\mathbf{R}_e$, is to be kept constant at $\mathbf{R}_{ref}$, an additional rotational error function $\mathbf{E}_r$ can be defined as

$$\mathbf{E}_r(\mathbf{R}_e, \mathbf{R}_{ref}) = \mathbf{R}_{ref} \times \mathbf{R}_e^{-1} \tag{1.2}$$

where $\times$ denotes matrix multiplication. The resulting error can be expressed as a rotational matrix, Euler angles, roll-pitch-yaw angle, or other orientation descriptions [26].

1.3.2 Path Following

To improve the dynamic accuracy of robots following a predefined path, the fundamental problem to solve is that of path following. In the context of this research, path following consists of guiding the robot following a geometric reference curve in the robot workspace. Contrary to the typical trajectory tracking, no temporal constraint is imposed, i.e., it does not matter when the end effector reaches a certain point on the path.

The path following problem is defined based on work by [27]. Consider a reference path given by $\mathcal{P}$ defined by the parameterization $p(\theta) \in \mathbb{R}^6$ where θ is the path parameter.

Usually, the specified path has start and end points defined by θ_0 and θ_g, respectively. Additionally, consider the robot end effector pose to be the output of a discrete-time system given by

$$x_{t+1} = f(x_t, u_t) \tag{1.3}$$

where x_t and u_t are the end effector pose and control input, respectively. Then, the path following task has to satisfy the following two constraints.

(1) **Path Convergence:** The kinematic error function $\mathbf{E}(x_t, p(\theta_t))$ as discussed in Subsection 1.3.1 has to converge such that

$$\lim_{t\to\infty} ||\mathbf{E}(x_t, p(\theta_t))|| \to 0$$

(2) **Convergence on Path:** The end effector can only move along path $\mathcal{P}$ in the forward direction, eventually converging to the end point, i.e.,

$$\dot{\theta}_t \geq 0 \quad \text{and} \quad \lim_{t\to\infty} ||\theta_t - \theta_g|| \to 0$$

In the above problem formulation, one can treat the path parameter θ as a virtual state and then solve the augmented system as a typical trajectory tracking problem [27]. However, for most industrial robots, the path following problem is already handled by their built-in internal controller. Indeed, most industrial robots only allow the real-time interaction of their path following motion at a positioning level since their position controller cannot be bypassed. As a result, the problem to be solved is the stabilization of the kinematic error function.

1.4 Contributions

Recall that the research goal is to develop control algorithms that can improve industrial robots' path accuracy. In terms of conceptualization and experimental setup, this thesis is built upon previous work from the benchmark study by [13]. The main novelty comes mainly from the experimentation of a reinforcement learning (RL) based control algorithm, which yields superior performance to the one proposed by the benchmark study. Note that much of the content of the thesis is to be published [28]. The contributions are as follows:

- The PBVS scheme and experimental setup consist of refined versions of the ones used in the benchmark study. The improvements are better consistency and various bug fixes in the external control software.

- The RL-based controller consists of a policy trained by a state-of-the-art actor-critic (TD3) method to optimize for path following performance. While reinforcement learning has already been used to improve the control performance of industrial robots with joint error feedback by [24], it is the first time that reinforcement learning has been used to control industrial robots based on Cartesian errors.

- Two experiments are carried out, characterized by their corresponding task path: line (position) and line (full pose). For each experiment, three experimental runs are carried out based on three control configurations to be compared: no external controller, baseline PID controller, and baseline PID controller supplemented with the RL-based controller. Note that the baseline PID controller is the same as the one proposed in the benchmark study. The performance of the three control configurations is compared based on two performance metrics: mean absolute error (MAE) and maximum error relative to the reference path. The path accuracy is characterized by the maximum value of the path errors across multiple experimental episodes.

- Regarding the first experiment, experimental results show that the supplementation

of RL-based controller improved positional path accuracy from 0.11 to 0.09 mm (-20%) in terms of 3D distance.

- Regarding the second experiment, experimental results show that the supplementation of RL-based controller worsened path accuracy from 0.29 to 0.30 mm (+4%) in terms of position but improved orientation path accuracy from 0.08 to 0.07 °(-9.2 %) in term of equivalent angle. Due to the relocation of the robot, this experiment cannot be properly completed to yield satisfactory results.

- Qualitatively speaking, the RL-based control algorithm is shown to affect the transient control performance favourably but not the steady state control performance. Practically, a better steady state convergence, i.e., a faster convergence to a steady state, is beneficial for processes requiring a quick response.

1.5 Thesis Outline

The thesis is structured as the followings. Chapter 2 elaborates on the visual servo control scheme and control problem under study. Chapter 3 presents the RL-based control algorithms designed to solve the previously defined problem. Chapter 4 presents the experimental setup to test and validate the proposed control algorithms, comprising the industrial robot and the optical sensor. Chapter 5 discusses the experimental results and compares the performances between the baseline and supplementary Rl-based solutions. Chapter 6 concludes the thesis and offers an outlook for future research.

Chapter 2

Visual Servoing Scheme

The control task involves guiding the robot precisely along a path where the geometric curve and speed of motion are predefined. Since only geometric constraints but not temporal constraints are involved, such a control task is often called *path following*. Industrial robots usually have their built-in path following solutions. However, the resulting path accuracy is abysmal due to the reliance on joint encoder feedback in conjunction with inaccurate kinematic models. To address this issue, many brands of industrial robots provide technologies that enable real-time high-level intervention of the robot's motion from an external device. However, most of these technologies are somewhat limited in that only the kinematics (position, velocity, acceleration) can be controlled while control of the dynamics (torque, motor voltage) remains inaccessible. Additionally, the implementation details of these software modules are unknown to the end users, adding another layer of unknown system dynamics. This research focuses on industrial robots, which provide the real-time capability of correcting the end effector Cartesian pose. In the scope of this research, the kinematic control is restricted to only position and orientation control. This restriction allows the original path following problem to be reformulated as a typical stabilization problem which is referred to as the path error stabilization problem. The plant to be stabilized consists of not only the robot but also any other subsystems affecting its intrinsic

motion, including embedded software, electronic systems, and even the specified task path. The feedback signal consists of the robot end effector pose extracted from measurements made by a visual sensor. Using a k-nearest neighbour algorithm, the Euclidean deviation of the end effector pose from the predefined path is calculated. This deviation, referred to as the *path error*, serves as the input to the controller. The pose correction controller itself includes two controllers structured in parallel. One is a PID controller, which is the primary stabilizer of the system. The other serves a supplementary role in optimizing control performance metrics related to path accuracy. The controller's output is directed to the robot's high-level real-time motion control system. Since the pose correction controller is run on an external device and receives a feedback signal from an external sensor, it is referred to as the *external* controller. The block diagram of the overall control system is shown in Figure 2.1. The focus of the research consists of the pose correction (external) controller block. In the rest of the chapter, each element of the aforementioned control system is elaborated further.

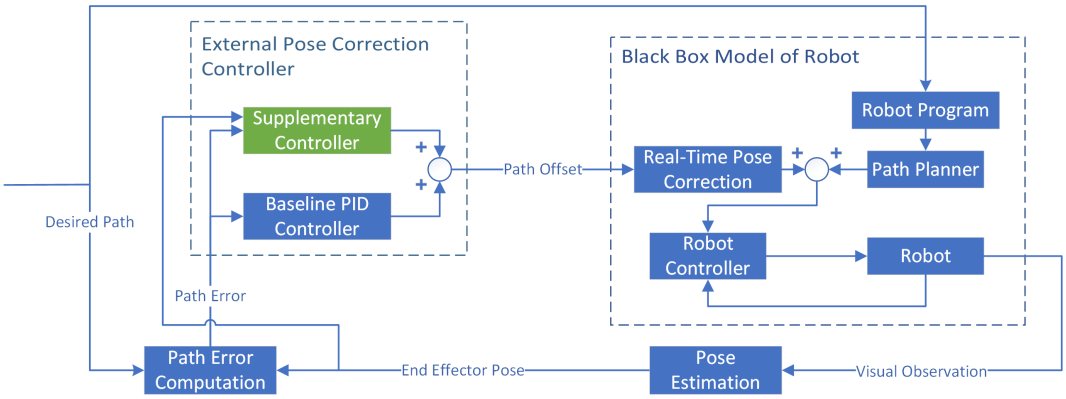


Figure 2.1: Block diagram of the closed-loop control system.

2.1 Robot Model

Based on the block diagram shown in Figure 2.1, the plant subjected to stabilization is an amalgamation of the various processes and systems related to industrial robots relevant to this research:

(1) **Robot**: The robot is a 6DOF serial robot comprising a series of actuators, links, and other electromechanical components.

(2) **Robot Program**: The robot program contains a sequence of Cartesian poses which define the path followed by the robot when performing the corresponding task.

(3) **Path Planner**: The path planner generates a trajectory in joint space based on the sequence of Cartesian poses specified in the robot program. The exact path planning algorithm used is generally proprietary and, therefore, unknown to the end user.

(4) **Robot Controller**: The robot controller, acting at the joint level, controls each actuator's torque based on the rotary encoders' feedback.

(5) **Real-Time Pose Correction Software**: The real-time pose correction software enables real-time offsetting of the robot end effector Cartesian pose on top of the robot's intrinsic motion. The implementation of such software is proprietary.

Note that each subsystem does not have its dynamical model and critical parameters disclosed to the end users. Although possible, it is tedious to identify each of them, not to mention their interactions with each other. Since this research focuses on model-free learning-based, system identification is not necessary. Therefore, it suffices to lump all the subsystems together in a black box model, which will represent the plant to be controlled.

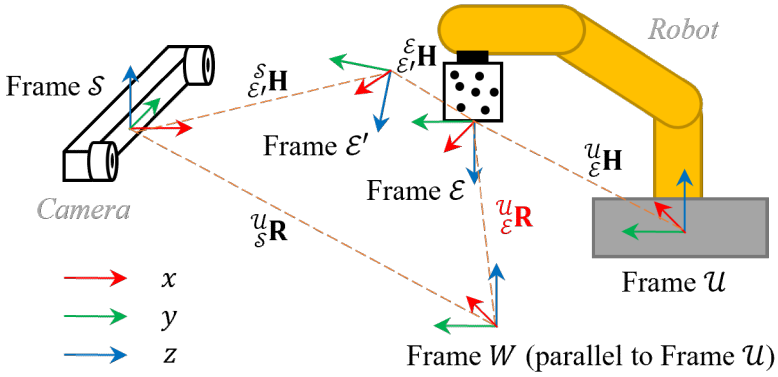


Figure 2.2: Schematics of frames used in the pose estimation and path error computation.

2.2 Pose Transformation

In this research, a commercial visual coordinate measurement machine (CMM) is used to measure robot end effector pose with respect to the sensor frame $\mathcal{S}$. Due to the effects of optical aberrations and vibrations, the measurement signal of visual sensors is inherently noisy, which is detrimental to feedback control. To smoothen out the feedback signal, a robust adaptive Kalman filter, similar to the one proposed by [13] and [29], is usually applied. The introduction of a filter inevitably would cause a phase shift which should be minimized via fine-tuning filter parameters.

Using real-time pose correction necessitates the expression of the end effector pose with respect to a frame $\mathcal{W}$ of which the axes are parallel to the robot frame $\mathcal{U}$. However, the optical CMM only provides the end effector pose with respect to the sensor frame $\mathcal{S}$. Consider $\mathcal{E}'$ to be the end effector frame as measured by the optical sensor, and the corresponding pose in the sensor frame be represented by the homogeneous transformation ${}^{\mathcal{S}}_{\mathcal{E}'}\mathbf{H}$ (transform from frame $\mathcal{E}'$ to frame $\mathcal{S}$). Additionally, consider $\mathcal{E}$ to be the end effector frame as measured by the robot's internal sensors, and the corresponding pose in the robot frame be given by ${}^{\mathcal{U}}_{\mathcal{E}}\mathbf{H}$. The relation between frames $\mathcal{E}$ and $\mathcal{E}'$ is that of a constant positional and rotational offset. The relation between all the frames is shown in Figure 2.2.

To obtain the pose in robot frame ${}^{\mathcal{U}}_{\mathcal{E}}\mathbf{H}$ from ${}^{\mathcal{S}}_{\mathcal{E}'}\mathbf{H}$, the following relation must be used.

$$ {}^{\mathcal{U}}_{\mathcal{E}}\mathbf{H} = {}^{\mathcal{U}}_{\mathcal{S}}\mathbf{H} \times {}^{\mathcal{S}}_{\mathcal{E}'}\mathbf{H} \times {}^{\mathcal{E}'}_{\mathcal{E}}\mathbf{H} \tag{2.1} $$

Here, $\times$ denotes matrix multiplication. Since the origin of the frame $\mathcal{W}$ does not need to coincide with frame $\mathcal{U}$, only rotation matrices $\mathbf{R}$ need to be considered.

$$ {}^{\mathcal{U}}_{\mathcal{E}}\mathbf{R} = {}^{\mathcal{U}}_{\mathcal{S}}\mathbf{R} \times {}^{\mathcal{S}}_{\mathcal{E}'}\mathbf{R} \times {}^{\mathcal{E}'}_{\mathcal{E}}\mathbf{R} \tag{2.2} $$

Using the above equation requires that both ${}^{\mathcal{U}}_{\mathcal{S}}\mathbf{R}$ and ${}^{\mathcal{E}'}_{\mathcal{E}}\mathbf{R}$ to be found empirically. The method used in this research is as follows:

(1) Pick any position in the robot workspace as the initial position ${}^{\mathcal{S}}\mathbf{P}_0$ to be measured in the sensor frame

(2) Move the robot in the x-axis of the robot frame to a point ${}^{\mathcal{S}}\mathbf{P}_1$ also measured in sensor frame.

(3) Move the robot in the y-axis of the robot frame to a point ${}^{\mathcal{S}}\mathbf{P}_2$ also measured in sensor frame.

(4) Calculate the unit vectors of the robot frame as follows:

$$ {}^{\mathcal{S}}\hat{\mathbf{x}} = {}^{\mathcal{S}}\mathbf{P}_1 - {}^{\mathcal{S}}\mathbf{P}_0 \;/\; \| {}^{\mathcal{S}}\mathbf{P}_1 - {}^{\mathcal{S}}\mathbf{P}_0 \| $$

$$ {}^{\mathcal{S}}\hat{\mathbf{y}} = {}^{\mathcal{S}}\mathbf{P}_2 - {}^{\mathcal{S}}\mathbf{P}_1 \;/\; \| {}^{\mathcal{S}}\mathbf{P}_2 - {}^{\mathcal{S}}\mathbf{P}_1 \| $$

$$ {}^{\mathcal{S}}\hat{\mathbf{z}} = {}^{\mathcal{S}}\hat{\mathbf{x}} \times {}^{\mathcal{S}}\hat{\mathbf{y}} \;/\; \| {}^{\mathcal{S}}\hat{\mathbf{x}} \times {}^{\mathcal{S}}\hat{\mathbf{y}} \| $$

where $\times$ denotes the cross product.

(5) Obtain the rotation from frame $\mathcal{S}$ to $\mathcal{U}$ as

$$ {}^{\mathcal{U}}_{\mathcal{S}}\mathbf{R} = \begin{bmatrix} {}^{\mathcal{S}}\hat{\mathbf{x}} & {}^{\mathcal{S}}\hat{\mathbf{y}} & {}^{\mathcal{S}}\hat{\mathbf{z}} \end{bmatrix}^T $$

.

(6) Take end effector pose measurements arbitrarily from both the optical sensor and robot sensor to obtain ${}^{\mathcal{S}}_{\mathcal{E}'}\mathbf{R}$ and ${}^{\mathcal{U}}_{\mathcal{E}}\mathbf{R}$.

(7) Calculate the rotation offset between $\mathcal{E}$ and $\mathcal{E}'$ as

$$ {}^{\mathcal{E}'}_{\mathcal{E}}\mathbf{R} = \left({}^{\mathcal{S}}_{\mathcal{E}'}\mathbf{R}\right)^{-1} \times \left({}^{\mathcal{U}}_{\mathcal{S}}\mathbf{R}\right)^{-1} \times {}^{\mathcal{U}}_{\mathcal{E}}\mathbf{R} $$

.

The aforementioned procedure assumes that the robot can move in the Cartesian coordinates of its internally defined frame.

2.3 Path Error Analysis

In the scope of this research, the real-time pose correction control depends on the deviation of the actual end effector pose from the desired end effector pose on the path, referred to as the *path error*. *The path error is the difference from the actual pose to the nearest pose on the desired path.* If the path is a simple line, the path error is just the Euclidean distance of a point to a line. However, for more complicated paths, the computation of path error is not so trivial. One challenge is that the nearest pose on the path may not necessarily lie in the neighbourhood of the actual path progression, potentially happening when sections of the path at a different progression stage get close to each other. Moreover, it needs to account for the fact that the motion along the path is always forward and never backward.

A path error calculation algorithm based on k-nearest neighbour (kNN) is proposed to address the above issues as shown in Algorithm 1. The kNN algorithm is a non-parametric supervised learning method suitable to classify nearest neighbours based on distance [30]. The kNN algorithm search the nearest neighbour via brute force.

Algorithm 1 k-nearest neighbour path error algorithm.

Initialize a path $\mathcal{P}$ defined by a sequence of positions $p_0, p_1, p_2, \ldots, p_N$ and Euler angles $\psi_0, \psi_1, \psi_2, \ldots, \psi_N$.
Initialize the path progression index $\eta \leftarrow 0$
Initialize algorithm parameters: τ
for each incoming measured position vector q_i and Euler angles ξ_i **do**
Find nearest position to q_i among p_η, $p_{\eta+1}$, $p_{\eta+2}$, $\ldots$, $p_{\eta+\tau}$.
$p_m \leftarrow$ k-NearestNeighbour$(\eta, \mathcal{P})$
$\eta \leftarrow m$ **only if** $m > \eta$
Find the line $\mathcal{L}$ defined by the position pair $\{p_m, p_{m+1}\}$.
Calculate the desired position q_{id} as the nearest position to q_i from line $\mathcal{L}$.
$q_{id} \leftarrow$ ClosestPointFromLine$(q_i, \mathcal{L})$
Calculate the fraction β of distance travelled between p_m and p_{m+1}.
$\beta \leftarrow \frac{||q_{id} - p_m||}{||p_{m+1} - p_m||}$
Calculate the desired orientation ξ_{id} with linear interpolation.
$\xi_{id} \leftarrow \psi_m + \beta(\psi_{m+1} - \psi_m)$
Calculate the position error e_p and orientation e_r error.
$e_p \leftarrow q_{id} - q_i$
$e_r \leftarrow \xi_{id} - \xi_i$
end for

Regarding the algorithm, the path is required to be defined as a sequence of positions and orientations represented by Cartesian vectors and ZYX-Euler angles, respectively. An index is introduced to keep track of the path progression and prevent backtracking during path error calculation. The algorithm loops for each time a new pose measurement is made during the robot's path following. With an off-the-shelf kNN algorithm, the nearest position to the measured position is found from the path section marked by the path progression index. Subsequently, a line is best fitted to the position pair consisting of the nearest position and its subsequent neighbour, as shown in Figure 2.3. The desired position is then defined as the closest position to the measured position from the line defined by the position pair. Concerning orientations, a simple linear relationship is established between the

position pair and their corresponding orientations. The desired orientation corresponding to the desired position can thus be calculated using this linear relationship. Once both the desired position and orientation are found, calculating the position and orientation errors is done by subtracting the desired value from the measured value.

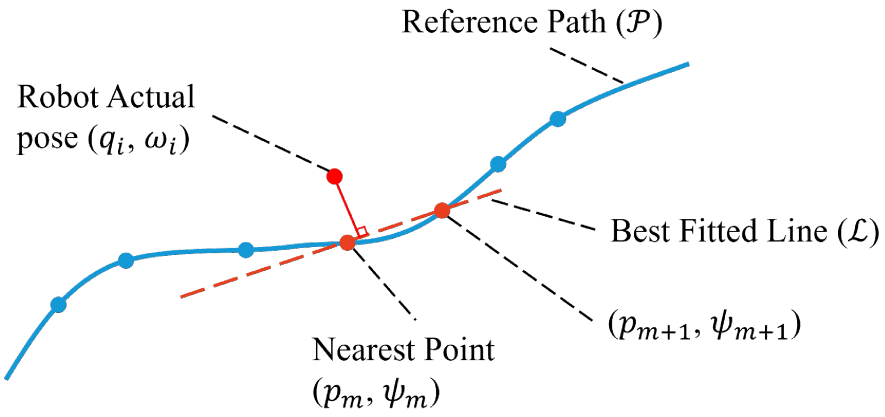


Figure 2.3: Path error computation schematics.

2.4 Path Stabilization Problem

The original path following problem, mentioned in Section 1.3, is to be reformulated as a path error stabilization problem. Indeed, since the robot already follows the programmed path intrinsically with its internal system, it only suffices for the pose correction controller to induce a slight motion adjustment to enhance path accuracy. Thus, the control goal is to stabilize the path error as defined in Section 2.3, to zero (ideally).

Let the black box model of the robot be treated as a discrete-time nonlinear system.

$$x_{t+1} = f(x_t, u_t) \tag{2.3}$$

where f is an unknown nonlinear function of the end effector pose x and the external control signal u. The end effector pose x is expressed with respect to the robot frame $\mathcal{U}$.

The system's output, denoted by y, is the end effector pose measured by the visual CMM and expressed with respect to the sensor frame $\mathcal{S}$, given by

$$y_t = \left({}^{\mathcal{U}}_{\mathcal{S}}H\right)^{-1} x_t \tag{2.4}$$

where ${}^{\mathcal{U}}_{\mathcal{S}}H$ is the homogeneous transformation matrix from frames $\mathcal{S}$ to $\mathcal{U}$ as defined in Section 2.2. The path to be followed $\mathcal{P}$ is given as a sequence of desired poses. And, the path error e is computed with the algorithm presented in Section 2.2 as

$$e_t = E(y_t, \mathcal{P}) \tag{2.5}$$

where E is the function representing algorithm 1. From the path error definition, the path stabilization task is defined as follows.

Definition 2.4.1. The path stabilization task consists of controlling x_t with an external controller $u_t = g(e_t, x_t)$ such that $e_t \to 0$.

2.5 Pose Correction Controller

The design of the pose correction controller needs to respect certain limitations imposed by the real-time pose correction mechanism provided by the industrial robots. These limitations are as follows:

(1) Real-time control is limited to only pose correction (no modification of velocity or acceleration available).

(2) Pose offset is to be given in Cartesian coordinates of the robot frame $\mathcal{U}$.

(3) Pose correction occurs in parallel with the robot's intrinsic motion. The intrinsic motion consists of the motion solely induced by the built-in robot controller and the

robot program.

Due to the model of the plant being unknown, the controller must be model-free. Previous work has already explored PID-like controllers to yield significant path accuracy improvement. PID control is still utilized in this research as the main feedback stabilization mechanism. A supplementary control is added to work in parallel with the PID controller to improve path accuracy. The combined control signal is therefore given as

$$u_t = g_{\text{pid}}(e_t) + g_{\text{sup}}(e_t, x_t) \tag{2.6}$$

where g_{pid} is the baseline PID controller and g_{sup} is the supplementary controller.

2.5.1 Baseline PID Controller

PID control has proven to be a highly effective model-free control method, as shown by its widespread application in industries. Hence, PID control serves as the baseline of comparison for any other proposed control solutions to assess their performance. The baseline PID is the main feedback path stabilization mechanism within the overall control framework. In the discrete-time domain, the PID controller is typically implemented in the following recursive form:

$$\Delta g_{\text{pid}}(e_t) = k_p(e_t - e_{t-1}) + k_i(e_t) + k_d(e_t - 2e_{t-1} + e_{t-2}) \tag{2.7}$$

where k_p, k_i, k_d are the proportional, integral, derivative control coefficients. The tuning of these coefficients is usually done via trial and error.

2.5.2 Supplementary Controller

The main objective of this research is to outperform the PID baseline controller. Instead of replacing the PID controller, it is kept as is, but a secondary controller is introduced to complement it. This supplementary controller is meant to induce fine motion adjustment parallel to the baseline PID control. The design of such a supplementary controller is the main challenge of this research. In general, the supplementary controller, denoted by $g_{\text{sup}}(e_t, x_t)$ is a nonlinear function of both the path error and the robot end effector pose. In the work by [13], it was proven that the baseline PID controller guarantees the robot's stability. Compared to the baseline PID controller, the supplementary controller is designed to contribute significantly less control effort. Thus, it is assumed not to affect the overall system stability.

2.6 Summary

Chapter 2 elaborates on the control scheme and discusses the control problem to be solved in detail. As shown in Figure 2.1, the control scheme is a feedback system composed of parts corresponding to the robot model, the pose estimation, the path error computation, and the pose correction controller. The feedback controller comprises a baseline PID controller acting in parallel with a supplementary controller. Moreover, it is identified that the control problem is that of a stabilization problem because industrial robots commonly have the feature of real-time pose correction based on an external signal. Based on the aforementioned control scheme, Chapter 3 will propose an RL-based controller serving as the supplementary controller.

Chapter 3

Reinforcement Learning Control

Chapter 3 proposes a RL-based controller which acts as the supplementary controller as discussed in Chapter 2. It consists of a controller parameterized as a neural network trained in an episodic manner via a state-of-the-art actor-critic method. Chapter 2. The proposed control algorithm will be discussed further in the rest of the chapter.

3.1 Preliminaries

Reinforcement learning is a machine learning paradigm in which an agent interacts with the environment to learn a policy which maximizes long-term reward. Most RL problem is modelled as a *Markov decision process* (MDP) which is a framework for sequential decision making as shown in Figure 3.1. At each time step t, the agent takes an action a, receives an reward r, and transitions from current state s to next state s'. Actions are selected based on a *policy* π which is a mapping from states to the sets of probabilities measures of action space $\mathcal{A}$, i.e, $\pi_\theta(a|s)$ is the conditional probability density at a given s. According to [31], an MDP is comprised of

(1) Action space: action $a \in \mathcal{A}$

(2) State space: state $s \in \mathcal{S}$

(3) Initial state distribution: $\rho(s_1)$

(4) Transition dynamics: $p(s'|s, a)$

(5) Reward function: reward $r = R(s, a)$

For an MDP, the transition dynamics have to satisfy the Markov property, such as

$$p(s_{t+1}|s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1}|s_t, a_t)$$

where the subscript t denotes the discrete time step corresponding to the state, action or reward. More precisely, the current time step information is sufficient to predict what happens in the next time step. The agent makes decisions over a sequence of discrete time steps in a stochastic environment.

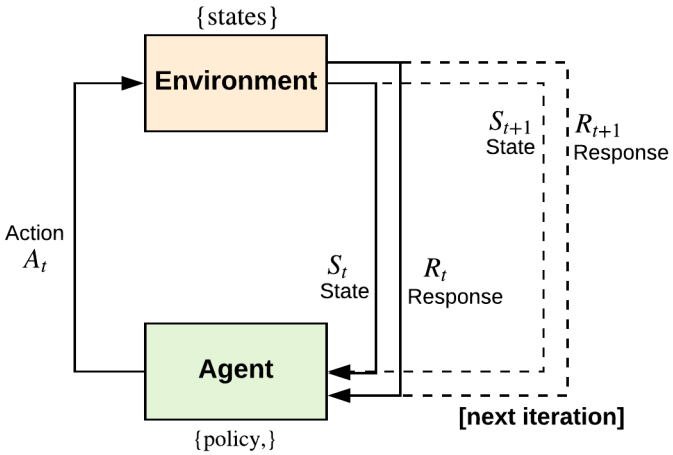


Figure 3.1: Markov decision process (adapted from [5]).

In this research, the task is considered episodic. For episodic task ending at time step T, the goal of the agent is to make a sequence of decisions to maximize the expected return

(discounted sum of immediate rewards) given by

$$g_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t} r_T \tag{3.1}$$

$$= \sum_{i=t}^{T} \gamma^{i-t} r_i \tag{3.2}$$

where the immediate reward is given as $r_i = R(s_i, a_i)$, and γ is the discount factor which puts more weight on immediate rewards. Practically, the expected return is computed using the recursive form given by

$$g_t = r_t + \gamma g_{t+1} \tag{3.3}$$

Another key concept of RL consists of value functions, which originated from dynamic programming and come in two types [32]. The state value function $V(s)$ is the expected return for a given state, which estimates how good it is for an agent to be in a given state, given by

$$V^{\pi}(s) = \mathbb{E}[g_t | s_t = s] \tag{3.4}$$

Here, $\mathbb{E}[\cdot]$ denotes the expected value of a variable. The action value function $Q(s, a)$ is the expected return for a given state-action pair, which estimates how good it is for an agent to perform a given action in a given state, given by

$$Q^{\pi}(s, a) = \mathbb{E}[g_t | s_t = s, a_t = a] \tag{3.5}$$

It is important to note that value functions are always defined with respect to a policy, annotated by a superscript. Due to the Markov property, most RL algorithms compute the

value functions in the recursive forms (Bellman equations), given by

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^{\pi}(s')] \tag{3.6}$$

$$Q^{\pi}(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s', a') \right] \tag{3.7}$$

Given the goal of maximizing return, an optimal policy π^* is defined such that the corresponding value functions have the highest possible values in all states, such as:

$$V^* = V^{\pi^*}(s) = \mathbb{E}_{\pi^*}[g_t|s_t = s] = \max_{\pi} V^{\pi}(s) \tag{3.8}$$

$$Q^* = Q^{\pi^*}(s, a) = \mathbb{E}_{\pi^*}[g_t|s_t = s, a_t = a] = \max_{\pi} Q^{\pi}(s, a) \tag{3.9}$$

If the MDP state transition dynamics $p(s', r|s, a)$ is known, the optimal policy can be found using the Bellman optimality equations given by

$$V^*(s) = \max_{a} \sum_{s',r} p(s', r|s, a) \left[r + \gamma V^*(s') \right] \tag{3.10}$$

$$Q^*(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} Q^*(s', a') \right] \tag{3.11}$$

Once the value functions are estimated, the optimal policy can be found either with the state value or action value function, such as

$$\pi^*(s) = \arg\max_{a} \sum_{s',r} p(s', r|s, a) \left[r + \gamma V^*(s') \right] \tag{3.12}$$

$$\pi^*(s) = \arg\max_{a} Q^*(s, a) \tag{3.13}$$

In the case where the MDP dynamics are unknown, most model-free RL algorithms still make use of the Bellman optimality principle.

In model-free reinforcement learning, there is often the dilemma of exploration-exploitation. On the one hand, the agent should explore the environment to find new actions that yield better rewards. On the other hand, it is also desirable for the agent to repeat past favourable actions to maximize the total cumulative rewards. The critical challenge in designing a reinforcement learning algorithm is to balance exploration and exploitation. A straightforward way to control exploration is by the ϵ-greedy approach, where the agent explores at a predefined probability of ϵ, and exploits otherwise.

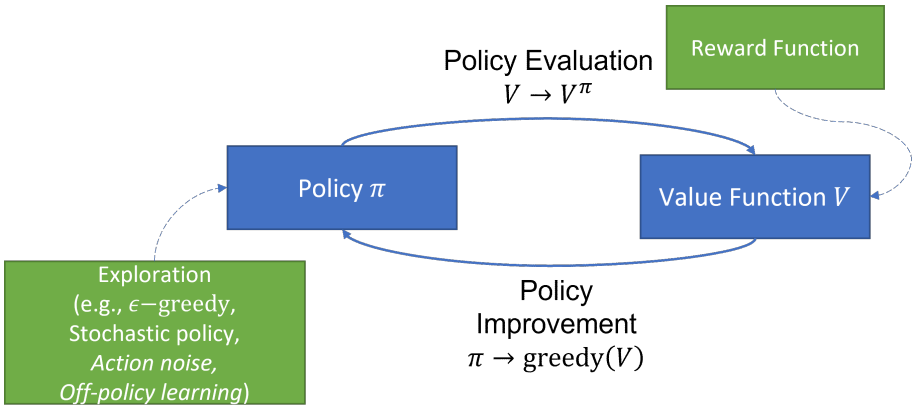


Figure 3.2: General policy iteration.

Finding an optimal policy often involves a process referred to as general policy iteration (GPI), which consists of two simultaneous, interacting processes as shown in Figure 3.2 [32].

(1) **Policy evaluation**: Make the value function more accurate with respect to the policy.

(2) **Policy improvement**: Make the policy *greedy* with respect to the value function, i.e., optimize the policy to maximize the value function for all states.

GPI implementation requires an exploration process (often embedded in the policy) and a reward function to complete the policy evaluation process. Almost all reinforcement learning algorithms are forms of GPI.

In the scope of this research, the action space $\mathcal{A}$ and state space $\mathcal{S}$ are considered to be continuous. The policy is constrained to be deterministic and is approximated by a nonlinear function $\pi_\phi(s) : \mathcal{S} \rightarrow \mathcal{A}$ which is a mapping from state space to action space, parameterized by vector ϕ. Similarly, the action value function is approximated by nonlinear function $Q^\pi_\theta(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ parameterized by vector θ. Concerning policy evaluation, a behaviour policy $\beta(s)$ is introduced to promote exploration. To improve the accuracy of the value function Q_θ, one has to minimize the temporal difference (TD) error given by [32]

$$L(\theta) = \mathbb{E}_\beta \left[\left(Q_\theta(s, a) - (r + \gamma Q_\theta(s', \pi(s'))) \right)^2 \right] \tag{3.14}$$

The standard approach to solve the optimization problem is the semi-gradient descent method which updates the function parameters as $\theta \leftarrow \theta + \alpha \nabla_\theta L(\theta)$ where α is the gradient step size.

Regarding policy improvement, the goal is to optimize $\pi_\phi(s)$ to maximize the performance measure defined as the expected initial return given by

$$J(\phi) = \mathbb{E}_\pi[g_1] \tag{3.15}$$

According to the deterministic policy gradient theorem, the performance gradient $\nabla_\phi J(\phi)$ with respect to the policy parameters is given by [31]

$$\nabla_\phi J(\phi) = \mathbb{E}_\beta \left[\nabla_a Q^\pi_\theta(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s) \right] \tag{3.16}$$

Then, it is straightforward to optimize the policy via gradient ascent by updating the function parameters as follows $\phi \leftarrow \phi + \alpha \nabla_\phi J(\phi)$.

Within the GPI framework, a widely used architecture is the *actor-critic* method consisting of the actor represented by policy approximation $\pi_\phi(s)$, and the critic represented by the action-value function approximation $Q^\pi_\theta(s, a)$, as shown in Figure 3.3. The actor is the

approximation of the optimal policy, while the critic estimates the true unknown optimal action-value function $Q^{\pi}(s, a)$. Typically, an actor-critic algorithm repeats the following process at each time step: the actor applies action a based on current state s, and then observes the next state s' and receives reward r, the action is evaluated using the temporal difference (TD) error, the critic is updated to minimize the TD error, and the actor is updated to maximize reward.

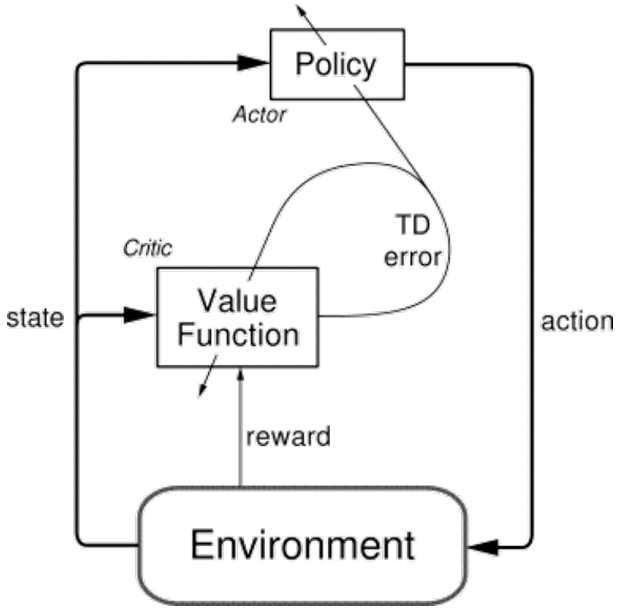


Figure 3.3: Actor-critic reinforcement learning scheme (adapted from [6]).

For continuous action space, one state-of-the-art actor-critic algorithms is the *twin delayed deep deterministic policy* (TD3) algorithm which is given by Algorithm 2 [33]. The TD3 contains the following important features:

(1) Deep neural networks approximate the actor and critic.

(2) The actor and critic are learned off-policy by introducing noise to the regular policy.

(3) The policy is updated by deterministic policy gradient ascent.

(4) Overestimation of value functions is reduced by double Q-Learning.

(5) Variance in the learning process is minimized via target networks and smoothing noise.

Algorithm 2 Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm.

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor-network π_ϕ, with random parameters θ_1, θ_2, ϕ
Initialize target networks $Q_{\theta'_1}$, $Q_{\theta'_2}$, $\pi_{\phi'}$, where $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for each episode **do**
  for $t = 1$ **to** T **do**
    Take action $a \sim \pi(s) + \epsilon$, where $\epsilon = \mathcal{N}(0, \sigma)$, and observe reward r, and next state s'
    Store state transition (s, a, r, s') in $\mathcal{B}$
    Sample a batch of N transitions from $\mathcal{B}$ and compute targets with target networks as follows
    $\tilde{a} \leftarrow \pi_{\phi'}(s) + \epsilon$ where $\epsilon = \text{clip}(\mathcal{N}(\prime, \tilde{\sigma}), -\rfloor, \rfloor)$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
    Update critics $\theta_i \leftarrow \min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
    if t **mod** d **then**
      update policy π_ϕ with deterministic policy gradient
      $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
      Update target networks
      $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
      $\phi'_i \leftarrow \tau\phi_i + (1 - \tau)\phi'_i$
    end if
  end for
end for

As shown in Algorithm 2, many hyperparameters must be fine-tuned to learn a high-performing policy. The parameter values can be quite different based on the reinforcement environment to be solved. Usually, the hyperparameters are tuned via trial and error, which is a tedious process. Preferably, one should be able to reuse a set of well-tuned hyperparameters for a particular RL environment in a different but similar RL environment. In practice, such an approach works well when the action and state spaces are always scaled from -1 to 1 [34]. A collection of well-tuned parameters for common RL environments can be found in open source RL training frameworks such as RL Baselines3 Zoo [35]. The list of hyperparameters for TD3 and the corresponding tuned values for a typical simple environment is given in Table 3.1. Note that this research does not involve high-dimensional states. Therefore, the neural network architecture of both actor and critic consists of the

straightforward multi-layer perceptron (MLP) with ReLU activation function. Both neural networks are optimized with the state-of-the-art gradient-based algorithm Adam [36]. Furthermore, this research deals with a relatively simple RL environment.

Hyperparameters	**Symbol**	**Common Value**
Actor Network	-	MLP[400,300]
Critic Network	-	MLP[400,300]
Activation Function	-	ReLU
Discount rate	γ	0.99
Policy delay	d	2
Actor learning rate	α_ϕ	$1e^{-3}$
Critic learning rate	α_θ	$1e^{-3}$
Batch size	-	100
Target update rate	τ	$5e^{-3}$
Exploration noise	ϵ	$\mathcal{N}(0, \sigma = 0.1)$
Smoothing noise	η	$\mathcal{N}(0, \tilde{\sigma} = 0.2)$
Smoothing noise clip	c	0.5
Optimizer	-	Adam

Table 3.1: Hyperparameters of TD3 algorithm for a simple environment.

3.2 RL-Based Supplementary Controller

As discussed in Chapter 2, the visual servo control scheme under study has limiting factors that affect the design of the control algorithm. Indeed, real-time motion control is restricted to position level only. Moreover, the dynamics of the robot and its subsystems are unknown and are therefore lumped together. Under such circumstances, a model-free optimal control method is required to enhance the path tracking accuracy of the system. In addition, as discussed in Chapter 2, the model-free controller should be modular to augment industrial robots' built-in control system seamlessly. Furthermore, the proposed controller should exploit the repetitive nature of robotic tasks to learn (self-optimize) from experience. Beyond the above requirements, the ideal supplementary controller should also strive to be task-agnostic, as discussed in Chapter 1. One learning-based solution candidate

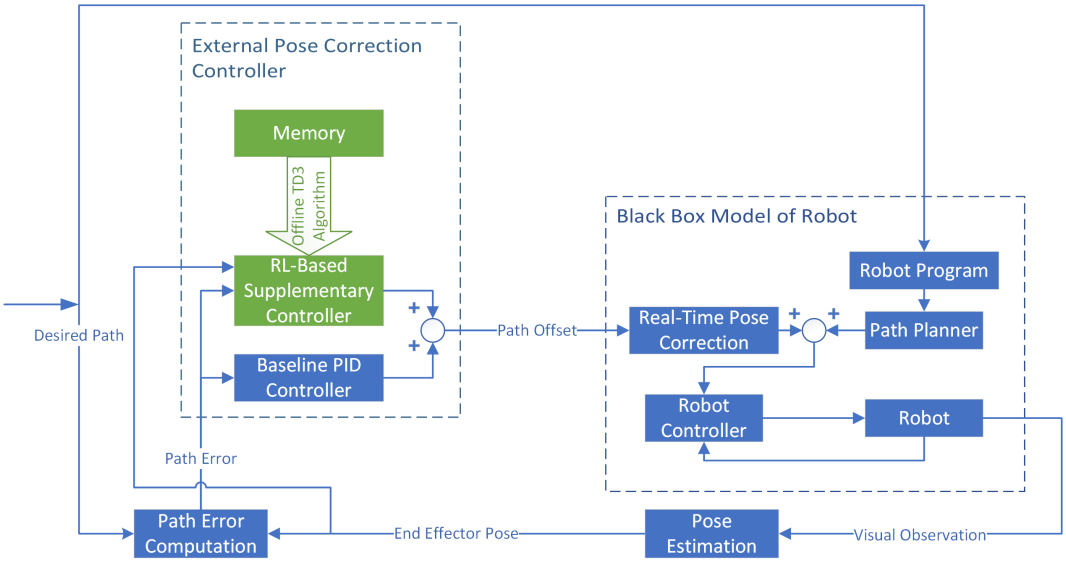


Figure 3.4: Block diagram of the control scheme with RL-based supplementary controller.

is iterative learning control (ILC) as introduced by [20]. The alternative solution consists of RL-based control.

Due to its significant drawbacks, ILC is not selected as the supplementary control method in this research. The learning convergence of ILC depends on the identical initialization condition (IIC) for each task trial. Indeed, the system must start at the same initial position for each trial, and the initial position, and velocity must be equal to the initial reference position and velocity [22]. This requirement severely limits the utilization of ILC in many manufacturing applications. Additionally, with the aforementioned shift to personalized manufacturing, industrial robots have to perform various tasks throughout their operational lifespan. Usually, each task requires a distinct set of ILC parameters to yield satisfactory performance. Therefore, the ILC controller is task-specific, necessitating the re-tuning of the robot's controller each time the task is switched. Due to the lengthy performance assessment comprising many learning trials, the tuning process of the ILC controller can become unacceptably tedious. The ideal learning-based controller should not have task-specific parameter tuning.

This research proposes an RL-based supplementary controller since RL can meet the

aforementioned requirements. RL is a versatile framework that solves problems modelled as Markov decision processes (MDP). Even highly complex tasks such as playing the game of Starcraft can be treated as MDP, where an agent trained by RL can reach parity with highly skilled human players [37]. Although RL needs to adhere to the Markov property, it is a much less strict constraint than the aforementioned IIC. Compared to ILC, RL can be applied to a broader class of problems. This research presumes that MDPs can adequately approximate the path following tasks of typical industrial robots. In terms of implementation, the modularity of reinforcement learning, as presented in Section 3.1, enables seamless integration with the existing control system of the industrial robot. Furthermore, most RL algorithms, such as TD3, do not rely on a model but rather on real-time data to train the policy (controller). RL can also be applied continuously over the operational lifetime of the robots, enabling the controller to self-adjust to compensate for dynamic changes in the environment. Like ILC, RL algorithms require tuning their hyperparameters to yield satisfactory learning performance. However, tasks of similar complexity levels can be learned well using the same hyperparameters. Nevertheless, the tuning of hyperparameters is generally task-specific and is out of the scope of this research.

Specific drawbacks of RL should also be noted. Due to its trial and error nature, RL is data-inefficient and computationally hungry. In addition, the Markov property does not necessarily hold in the real world. The probability distribution of specific processes might depend on the processes' history. Moreover, RL might not be suitable for high-precision applications, as discussed later in Chapter 5.

The control system described in Chapter 2 can easily be modelled as an MDP. The *task* consists of correcting the intrinsic motion of the robot, following a predefined task path, to minimize path errors as computed with Algorithm 1. The task in question is episodic, in accordance with the repetitive nature of most robotic tasks. The *environment* is the black

box model of the robot lumped with the baseline PID controller. Additionally, the environment has continuous action and state spaces. The *state* consists of the end effector pose and the path error. This state is given by vector $s = [x;\ e]$ where x is the end effector pose, and e is the path error. The *action* consists of the high level end effector pose correction signal as discussed in Chapter 2. The immediate *reward* consists of a function that punishes in proportion to the path error's magnitude. The design of the reward function is often based on experience and heuristics. Since the goal is to minimize the path errors, the reward function is set to be the negative L1-norm of the path errors. Consider the path error to be given by vector $\mathbf{e} = [e_1, e_2, \ldots, e_n]$, then the reward r is given by

$$r = -\sum_{i=1}^{n} |e_i| \tag{3.17}$$

Note that the design of the reward function is an open question subject to further investigation.

Once the MDP is properly defined, it comes down to training a policy with a state-of-the-art actor-critic method to maximize the total discounted cumulative rewards. The TD3 algorithm, as presented in Section 3.1, is the chosen RL algorithm due to its proven capability in solving complicated virtual environments. The policy $\pi_\phi(s)$ is approximated by a neural network parameterized by vector ϕ. Similarly, the twin value functions Q_{θ_1} and Q_{θ_2} are approximated by neural networks parameterized by vectors θ_1 and θ_2 respectively.

In terms of implementation, certain modifications need to be made to the TD3 method presented in Algorithm 2. Standard TD3 is an online learning algorithm which performs at least one training step at each time step. As such, it requires real-time computing, which can introduce latency and be a problem with insufficient computational power. Circumventing this issue requires the modification of the standard TD3 algorithm to perform offline training at the end of each episode, akin to iterative learning control (ILC). Another issue is the convergence to local minima caused by insufficient exploration. To promote

more exploration, *warm-up* episodes, where the policy samples randomly from a bounded continuous action space, are incorporated at the beginning of the RL training process [34]. Considering various practical considerations, a modified offline TD3 algorithm is presented in Algorithm 3.

Algorithm 3 Offline TD3 algorithm.

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor-network π_ϕ, with random parameters θ_1, θ_2, ϕ
Initialize target networks $Q_{\theta'_1}$, $Q_{\theta'_2}$, $\pi_{\phi'}$, where $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for each of E_w warm-up episodes **do**
  for $t = 1$ **to** T **do**
    Take action a sampled from a uniform distribution of the bounded action space $\mathcal{A}$, and observe reward r, and next state s'
    Store state transition (s, a, r, s') in $\mathcal{B}$
  end for
end for
for each of E_t training episodes **do**
  for $t = 1$ **to** T **do**
    Take action $a \sim \pi(s) + \epsilon$, where $\epsilon = \mathcal{N}(0, \sigma)$, and observe reward r, and next state s'
    Store state transition (s, a, r, s') in $\mathcal{B}$
  end for
  for each of G gradient steps **do**
    Sample a batch of N transitions from $\mathcal{B}$ and compute targets with target networks
    $\tilde{a} \leftarrow \pi_{\phi'}(s) + \eta$ where $\eta = \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
    Update critics $\theta_i \leftarrow \min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
    if t **mod** d **then**
      update policy π_ϕ with deterministic policy gradient
      $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
      Update target networks
      $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
      $\phi'_i \leftarrow \tau\phi_i + (1 - \tau)\phi'_i$
    end if
  end for
end for

Within the overall control scheme, the policy trained via reinforcement learning, corresponds directly to the supplementary controller $g_{\text{sup}}(x_t, e_t) = \pi_\phi(x_t, e_t)$. As shown in the block diagram from Figure 3.4, the trained policy contributes to the control signal in summation of the baseline PID control signal. At the end of each episode, the policy is

trained based on Algorithm 3. The typical hyperparameters of the training process can be found in Table 3.2. The amount of training is capped by the specified number of training episodes.

Hyperparameters	**Symbol**	**Common Value**
Actor Network	-	Mlp[60,50]
Critic Network	-	Mlp[120,100]
Activation Function	-	ReLU
Discount rate	γ	0.98
Policy delay	d	2
Actor learning rate	α_ϕ	$1e^{-3}$
Critic learning rate	α_θ	$1e^{-3}$
Batch size	-	100
Target update rate	τ	$5e^{-3}$
Exploration noise	ϵ	$\mathcal{N}(0, \sigma = 0.1)$
Smoothing noise	η	$\mathcal{N}(0, \tilde{\sigma} = 0.2)$
Smoothing noise clip	c	0.5
Optimizer	-	Adam
Gradient steps	G	200
Warm-up episodes	E_w	20
Training episodes	E_t	60

Table 3.2: Hyperparameters of the offline TD3 algorithm.

3.3 Summary

In Chapter 3, reinforcement learning is proposed to improve the path following accuracy as the episodic task is repeated. At the end of each episode, an offline TD3 algorithm is used to train the optimal policy approximated by a neural network. This approximation corresponds directly to the supplementary controller within the control framework presented in Chapter 2. The RL-based supplementary controller is to be trained and evaluated using the experimental setup discussed in Chapter 4.

Chapter 4

Experimental Setup

Chapter 4 presents an experimental platform designed to test and validate various solutions to enhance path accuracy. It is an implementation of a visual servoing system comprising three main components: the FANUC M-20iA robot (*plant*), the Creaform C-Track 780 optical CMM (*sensing element*), and the high-level external controller implemented on an external device (*controller*). The M-20iA robot is a 6DOF industrial serial robot, of which the most pertinent feature is Dynamic Path Modification (DPM), a software module that allows the end effector pose to be adjusted in real-time parallel to the robot's intrinsic motion. The C-Track is a dual camera optical sensor capable of continuously measuring the position and orientation of objects marked with optical reflectors. The high-precision visual feedback the C-Track provides is critical for path accuracy improvement. The external controller is a software application that hosts the proposed control algorithms and integrates the M-20iA robot with the C-Track via APIs provided by the respective manufacturers. The design of the external controller needs to be compatible with the proprietary systems already in place. Moreover, any proposed control algorithms must satisfy constraints specific to the control scheme defined in Chapter 2. The visual servoing experiment is designed around the various technicalities of the hardware. In the rest of the chapter, each main experimental component and the experiment itself are further discussed.

4.1 Creaform C-Track 780

The Creaform C-Track 780 is a dual camera optical sensor that can measure the position and orientation of object models marked by retroreflective stickers. The retroreflective stickers represent targets observed concurrently by the C-Track at each measurement instance. The measurement data are then processed to estimate the pose of the marked object via C-Track's embedded algorithm. The off-the-shelf pose measurement capability of the C-Track enables various applications such as 3D model tracking, 3D model probing, 3D model inspections, 3D scanning, and other functions. Various modules of the VXelements software manage the many functionalities of the C-Track. An API provides access to all the functionalities of VXelements in Microsoft's .NET framework, enabling the integration of C-Track with the test platform.

Figure 4.1: The C-Track 780 dual camera sensor (Creaform).

In this research, it is required to continuously track the position and orientation of the robot end effector in real-time. This task is handled by the dynamic tracking module VXtrack of VXelements. The VXtrack module enables measuring the 6DOF poses of object models, each defined by a rigid set of reflectors. Since the 6DOF poses of multiple rigid objects can be acquired simultaneously, the typical usage of C-Track involves the measurement of the relative pose of one object with respect to another, which is the case in this

research. The general procedure to carry out such a dynamic tracking task consists of the following steps:

(1) Perform calibration of the C-Track using a calibration bar with reflector targets of which the relative geometric relations are entirely known. The calibration process is done for the chosen measuring volume with the options of 3.8 to 14.8 m^3.

(2) Place reflector targets on the surfaces of the robot end effector and of the reference object according to manufacturer guidelines.

(3) Position the C-Track to maintain a good view of all reflector targets throughout the predefined motion of the robot.

(4) Select all the reflector targets placed on the robot end effector to define the *tracking model*. The VXtrack software continually measures the position and orientation of the frame associated with the tracking model. The origin of this frame is, by default, the centroid of the reflector targets, but a given offset can modify it.

(5) Select all the reflector targets placed on the reference object to define the *positioning targets*. The position and orientation of the tracked object model are always given with respect to the frame defined by the positioning targets.

(6) Start the tracking process.

The C-Track is a high-performance metrology instrument suitable for improving robot path accuracy. In terms of processing power, the C-Track can continually track object 6DOF pose at a sampling rate of up to 29 Hz, which more or less eliminates the typical time delay issues associated with the visual servoing system [25]. Regarding precision, the C-Track has tracking repeatability of 0.0025 mm, and a volumetric accuracy of 0.065 mm [38]. Note that the repeatability of the C-Track is superior to that of the M-20iA

robot, justifying the potential for accuracy improvement. The theoretical limit of accuracy enhancement should ideally be equal to the repeatability mentioned above.

In the scope of the overall control system, as shown in Figure 2.1, the C-Track serves as the pose estimation element. It is noted that the end effector poses estimated by the C-Track are not only given in the sensor frame but also deviated by a constant offset from the end effector pose as observed by the robot. Expressing the end effector pose in the robot frame requires the transformation procedure shown in Section 2.2.

4.2 FANUC M-20iA Robot

The FANUC M-20iA is a 6-DOF industrial serial robot, as shown in Figure 4.2. The robot system includes the R-30iB controller, the teach pendant, and the associated embedded software. The controller is a computer that controls the robot's operation, the robot's motion, and the robot's communication with external devices. The teach pendant is a hand-held device that serves as the operator interface of the embedded software in the controller. Its typical use is to control and program the robot. For safety, the robot work area is enclosed by safety fences and accessible through the safety gate. As a security measure, interlock devices were installed to prevent the robot from moving when the safety gate was open.

The M-20iA robot has a payload of 20 kg and a reach of 1811 mm [39]. Taking into account the topic of this research, one crucial specification of the robot is its repeatability which is defined as how precise a robot can return to the same position and orientation in its workspace. According to the ISO 9283 standard, [19] verified that the repeatability of the FANUC M-20iA robot is 0.08 mm. Although repeatability represents the upper limit of theoretically achievable accuracy for static positioning using open-loop approaches, its influence on the performance of closed-loop path following might not be as straightforward and remains an open problem.

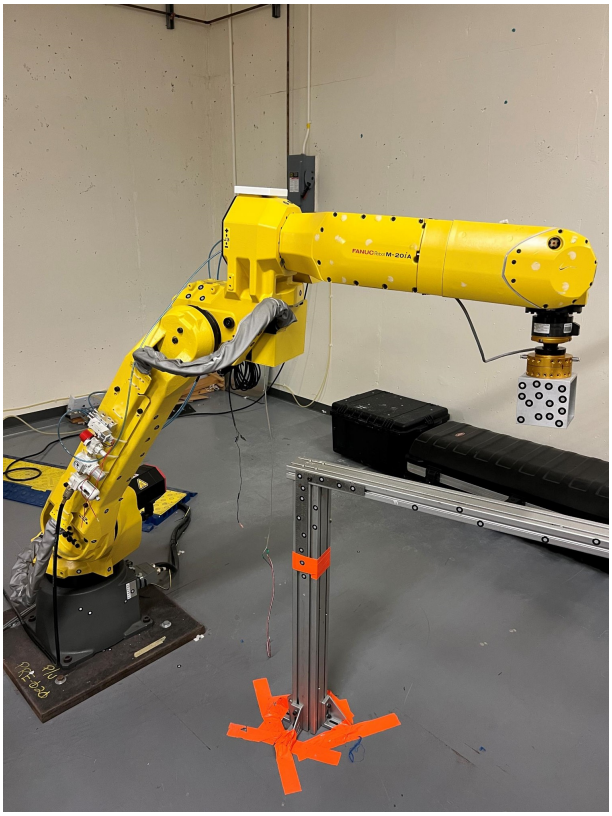


Figure 4.2: The M-20iA robot (FANUC).

Usually, the end user programs and operates the M-20iA robot solely with the teach pendant using the online programming paradigm. Online programming is the process of manually moving the robot to a sequence of desired poses to be recorded in the teach pendant. The resulting program is called a TP program. The robot's internal controller generates the task path via interpolation between the recorded poses from the TP program. For example, a line path only necessitates the start and end points to be taught. For paths defined by large amounts of points, the online programming process can be tedious, making programming a complex task path unfeasible. An alternative solution, offline programming, can be used to address this issue. In the offline programming paradigm, the task path is generated numerically in a virtual environment provided by robot simulation software such as RoboGuide (FANUC) or RoboDK. Such software can often generate the TP program corresponding to a robotic task. However, the drawback is that offline programming

exacerbates the inherent poor accuracy of the robot. In this research, online programming is sufficient since the robot task is limited to line path following.

Based on chapter 2, the robot, along with the R-30iB controller and the teach pendant, consists of a self-contained system that cannot be modified nor accessed by the users. Therefore, one cannot control the robot in real-time via torque or velocity. Fortunately, FANUC provides a real-time control option called the Dynamic Path Modification (DPM) software module. While the robot is moving based on the selected TP program, 6DOF Cartesian pose offset can be applied every 8 ms via DPM to modify the motion of the end effector [40]. While the robot's response to the DPM offset signal can be fine-tuned with the DPM parameters, the end users do not know the details of the underlying implementation of DPM. The DPM software module can be accessed from an external computer using the PC Developer's Kit (PCDK) provided by FANUC, a software that allows communication with the M-20iA via TCP/IP. An API of this software is available in the .NET framework.

Note that the DPM function carries a few peculiar caveats. First, DPM can only be applied when the robot performs fine linear or circular motions. Second, DPM offsets must be in Cartesian space with respect to a predefined reference frame in the robot controller. Third, the motion induced by the DPM offset and the intrinsic motion occur concurrently but not at the same speed. Indeed, the DPM offset motion is usually much faster than the intrinsic motion, indicating that the robot runs at near maximum speed during DPM motion. This discrepancy in the speed of response decouples the DPM offset from the intrinsic motion, which justifies the reduction of the path following problem to path stabilization as discussed in Chapter 2. The characteristics of the robot's response to DPM can be tuned by modifying the DPM parameters. Fourth, the DPM applies the offset not to the position at a particular time instance but to the entire pre-programmed path. Once a single DPM offset is applied, the remaining path shifts by the same offset. Finally, if a DPM offset motion

does not complete before the next DPM offset signal arrives, then the previous DPM offset is overridden by the new DPM offset.

In light of the proprietary nature and complexity of the FANUC M-20iA robot, it is challenging and unreasonable to try to model the former with reasonable accuracy. As discussed in Chapter 2, the M-20iA robot and all its associated embedded systems are treated together as a single black box model, which is the *plant* to be controlled.

4.3 External Kinematic Controller

Considering the various technicalities of the FANUC M-20iA robot and the C-Track, a software application running on an external device must be developed to realize real-time closed-loop motion control. This high-level control software serves three primary purposes: remote control of the M-20iA robot, remote control of the C-Track, and computation of the DPM offsets with the control algorithms proposed in Chapter 3. Note that all devices are connected with Ethernet cables managed by an Ethernet switch and communicate with the TCP/IP protocol. Regarding implementation, the software was developed in the .NET framework to ensure compatibility with the VXelements and PCDK APIs. Additionally, the software consists of a Windows Form app written in C#; an attribute inherited from previous work [13]. The app is divided into three modules based on functionality relating to the M-20iA robot, C-Track, and real-time control. This separation can be seen in the form design shown in Figure 4.3. The source code is found in the following GitHub URL `https://github.com/ztao-sd/dotnet-fanuc-controller.git`.

The C-Track module contains functions that connect to the VXelements API, extract the model tracking data periodically, process the data and save the data. A typical sequence of actions to initialized object model tracking are:

(1) Connect to VXelement API

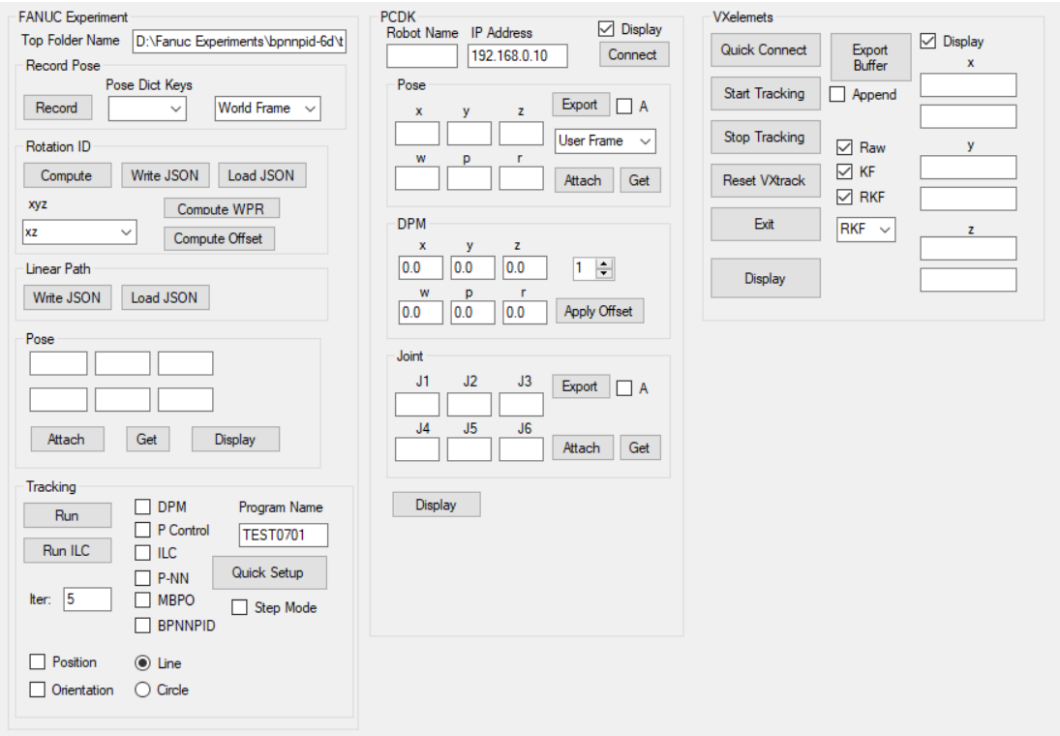


Figure 4.3: External software form design.

(2) Subscribe to events of the VXelement API

(3) Import positioning targets

(4) Import tracking model

(5) Start tracking

Once tracking starts, the VXelement API will trigger an event every time a new measurement instance of C-Track is completed, which happens according to the sampling rate of approximately 29 Hz. The corresponding event handlers would extract the incoming measurement data and store it in a memory buffer that other modules can access. Moreover, the event handlers may filter the data to enhance the quality of the measurement signal. In this research, a robust Kalman filter was used to attenuate the measurement noise, which is highly undesirable for path accuracy enhancement [41]. The events of VXelements are

triggered on a separate thread from the main application to prevent the app's slowdown and utilize parallel computing.

The FANUC module contains functions that connect to the M-20iA robot, configure the DPM parameters, send DPM to offset signal continuously at a fixed time interval, read and write digital I/O ports, run a given TP program, and extract robot 6DOF pose data as measured by the robot's built-in encoders. Contrary to the VXelements API, the PCDK API is not event-based. Reading data periodically from the M-20iA robot requires periodic event triggering with the aid of a timer implemented in the .NET framework. Like the VXelements API, the event handlers read the pose data and store it in a memory buffer that other modules can access. The pose data can be expressed in predefined Cartesian spaces or the joint space. The robot controller sends DPM offsets periodically by writing their values to certain system variables of the robot controller.

The real-time control module contains functions that calculate the transformation between sensor frame and robot frame, calculate path error, calculate DPM offset with proposed control algorithms, and send DPM offset periodically to the robot controller at a fixed time interval. The initialization sequence of this module involves two crucial actions:

(1) Calculate the rotation matrix between the sensor frame and the robot frame as well as the pose offset as discussed in Chapter 2 Section 2.2.

(2) Load the sequence of poses defining the reference path to be followed.

All the information regarding the rotation matrix, pose offset and sequence of poses can be easily exported to a JSON file and subsequently imported back to initialize the same control configuration. During real-time control, the module has a timer running on a separate thread which triggers events to calculate and send one instance of the DPM offset. The time interval between subsequent events is 80 ms, which is larger than the sampling interval of C-Track of 34.5 ms (29 Hz). The justification for this choice is that the control agent should

always be able to observe between subsequent actions such that it only performs at most one action per observation.

Concerning the RL-based control algorithm, the real-time control module trains the policy network with PyTorch within a Python environment. It deploys the policy in real-time via ONNX Runtime [42, 43]. Based on Algorithm 3, a Python script is executed at the end of each episode to train the policy network, which is then exported in the ONNX format. At the start of the next episode, the policy network is imported from the ONNX file and loaded for real-time inference.

4.4 Visual Servoing Experiment

The experimental layout is as shown in Figure 4.4. Note that both the end effector and reference object are covered with reflector targets observed by the C-Track to estimate their 6DOF poses. The relative placement of the end effector, reference object, and C-Track is such that the C-Track has a full view of all relevant reflector targets while the robot task is undergoing. On the computational side, the M-20iA robot and C-Track are driven by their corresponding controllers, while the external controller is run on a portable PC. In the scope of this research, there are two task paths: a straight line motion where the orientation of the end effector relative to the C-Track remains constant and a straight line motion where the orientation changes at a constant rate from start to end.

The overall experiment was structured into two separate experiments corresponding to the two task paths as previously mentioned. Each experiment contained many experimental runs, each characterized by a set of parameters and the number of episodes. The parameters included those which affect the intrinsic motion of the robot, the real-time position correction (DPM), and the control algorithms as proposed in Chapter 3. Each episode was a simultaneous execution of the predefined TP program (from the robot) and the high-level control software (from the external device). Indeed, the control algorithms proposed in

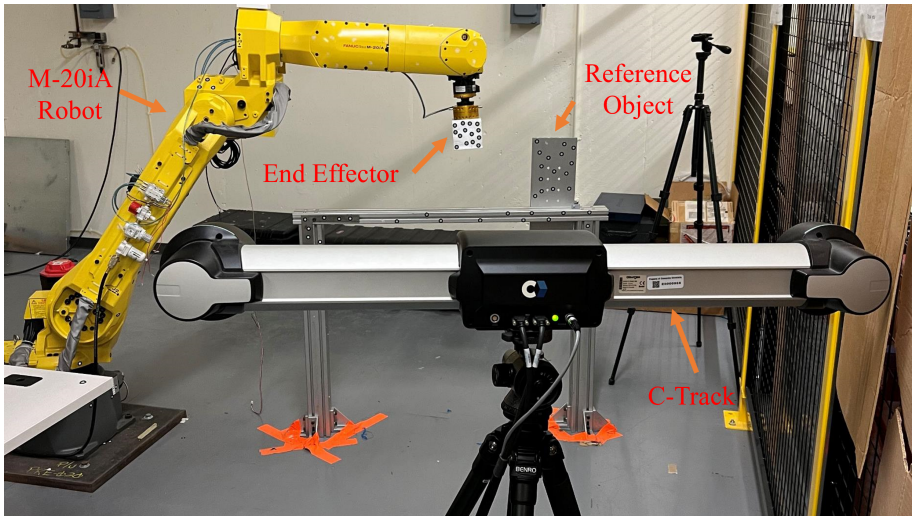


Figure 4.4: Experimental layout comprising of the M-20iA robot, the C-Track 780, and the end effector.

Chapter 3 are based on iterative optimization, which necessitates various training episodes. Due to the measurement stochasticity induced by the visual sensor C-Track and the robot, statistically meaningful result requires data collection across many episodes. Any control performance assessment should be based on performance metrics averaged over multiple episodes of data.

Before starting an experimental run, the following preparation steps were carried out.

(1) Calibrate C-Track and configure VXtrack according to Section 4.1.

(2) Generate a TP program from the desired task path by either programming online with the teach pendant or programming offline with robot simulation software.

(3) Modified the parameters of the TP program or the DPM parameters if needed.

(4) Choose one of the proposed control algorithms and a corresponding set of parameters.

(5) Prepare a file, readable by the external controller, that stores the desired task path data.

Since the experiment concerns mainly software, it can be well described by a programming flowchart as shown in Figure 4.5. The blocks in blue represent the external controller, the blocks in green specifically refer to the control algorithm proposed in Chapter 3, and the shapes in orange represent the TP program. The procedure can be summarized in the following points.

(1) **Initialization**: The initialization includes loading the desired path and the parameters of the experimental trial. Furthermore, a timer as discussed in Section 4.3 is enabled.

(2) **Start of TP program**: The TP program is started remotely from the external device. The first instruction of the TP program is to move the robot to the starting pose, after which DPM is enabled. Then, the robot moves according to its internally general path based on subsequently defined poses.

(3) **DPM synchronization**: Two digital outputs (DO) serve as flags to trigger to synchronize precisely the start of the control loop with the start of the task path. Both the external controller and the TP program can switch these DOs.

(4) **Control loop**: The control loop consists of a routine which is subscribed to the *tick* event triggered by the timer at a fixed interval as described in Section 4.3. This routine is composed of the following subroutines:

   (a) Check if the TP program is aborted.

   (b) Get pose measurement from VXelements.

   (c) Compute the path error.

   (d) Compute the DPM offset according to the chosen control algorithm.

   (e) Check if the path error and DPM offset are within the safety range.

   (f) Send DPM offset signal to the robot controller.

(5) **Termination**: Termination occurs when the TP program is aborted or the system is not considered safe. Then, most of the relevant variables are reset. Any iterative optimization routine is then executed. If it is the last iteration, the experiment is terminated, or a new iteration is initiated.

4.5 Summary

In Chapter 4, the discussion on the experimental setup is mainly about the experimental hardware structure and procedure. Regarding the hardware, the three main constituents are the FANUC M-20iA robot, the C-Track, and the external kinematic controller. Regarding the structure, the experiments are organized hierarchically based on the task path, the control configurations, and the experimental episodes. The procedure consisted of the following main processes: initialization, trigger of the TP program, DPM synchronization, control loop, and termination. Based on the experimental setup described above, many experiments are conducted to assess the control algorithm proposed in 3, and their results are discussed in Chapter 5.

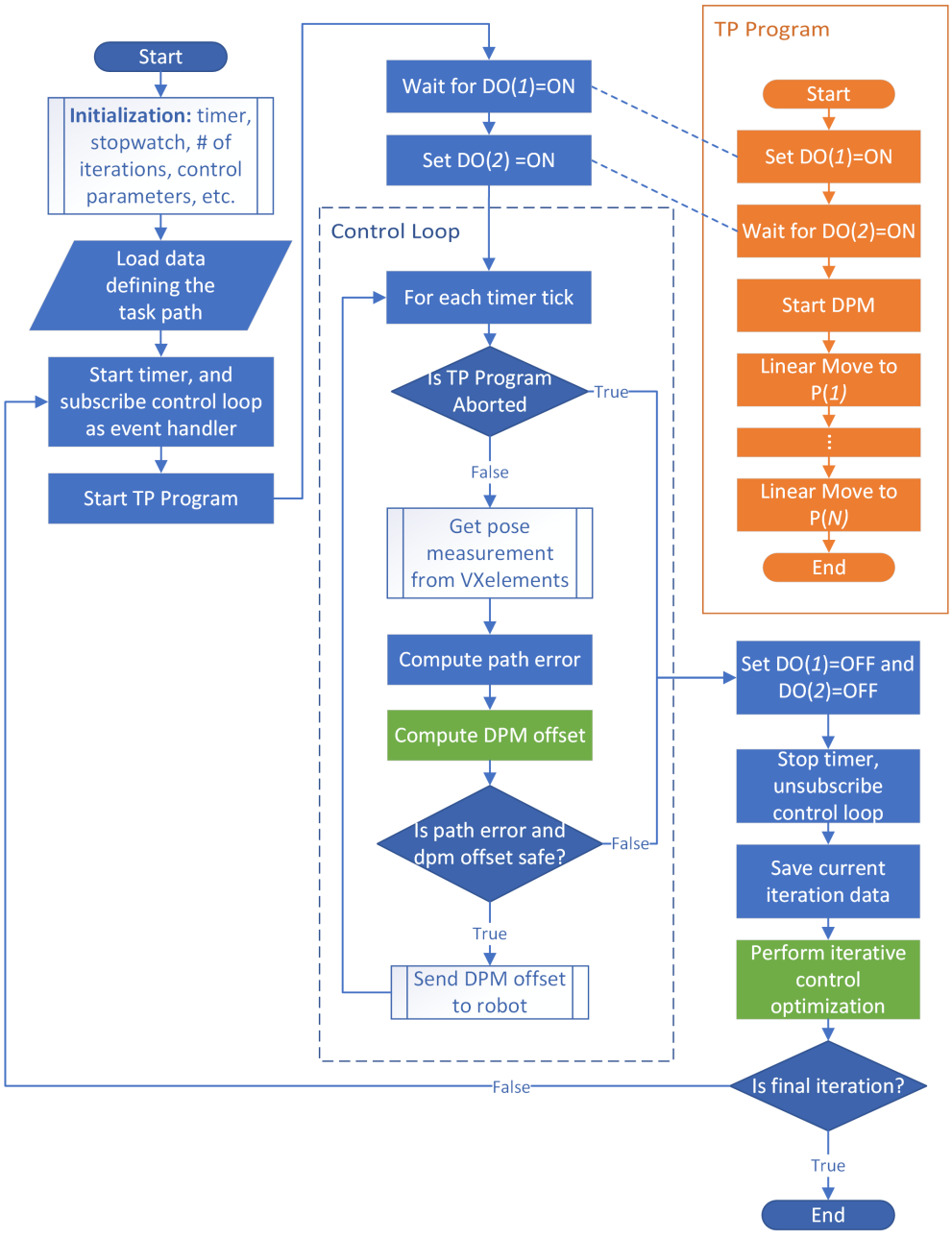


Figure 4.5: Flowchart of the experimental software.

Chapter 5

Experimental Results

Chapter 5 presents the experiments conducted in this study, the analysis of experimental data, and the comparison of performance metrics between the baseline PID and RL-based control algorithms. Two experiments are carried out: line following (position) and line following (full pose). The rest of the chapter elaborates further on the experiments and their results.

5.1 Experiment 1: Line Following (Position)

The position line following experiment consists of correcting the end effector of the industrial robot moving in a straight line at a fixed orientation. The straight line is completely defined by two points in space. In this experiment, the end-effector orientation is not subject to extrinsic intervention. Therefore, the orientation portion of the pose correction is set to zero at all times. As discussed in Chapter 4, the experiment is entirely characterized by two separate software applications running concurrently: the TP program and the external kinematic controller.

The TP program is generated via online programming with the teach pendant. The procedure to define a linear motion consists of teaching a start point and an end point

in Cartesian space, between which the robot will move at a given constant speed. The principal parameters of the TP program are shown in Table 5.1. Graphically, the line path is shown in the 3D plot of Figure 5.1.

TP Parameter	**Value**
Start point	$[-371 \text{ mm}, \ 836 \text{ mm}, \ 715 \text{ mm}]$
End point	$[-476 \text{ mm}, \ 1386 \text{ mm}, \ 724 \text{ mm}]$
Speed	50 mm/s

Table 5.1: Parameters of TP program for line following (position) experiment.

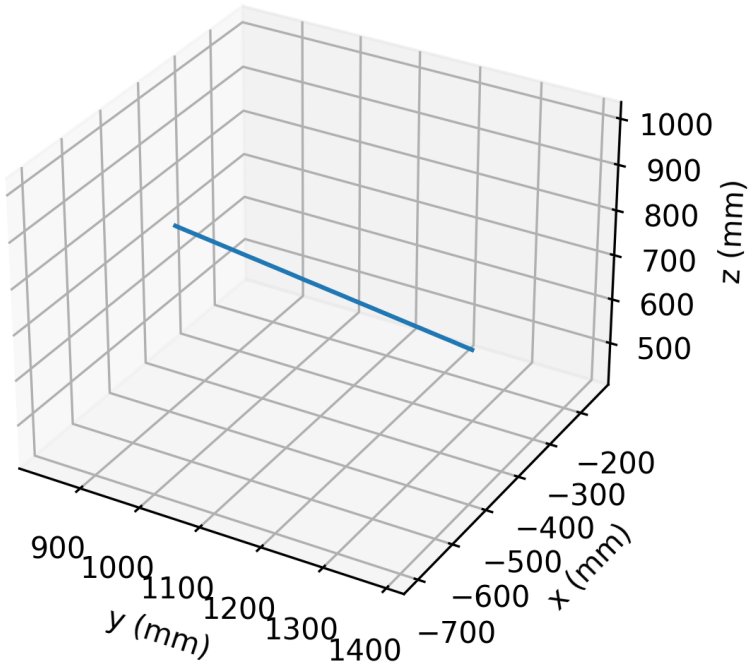
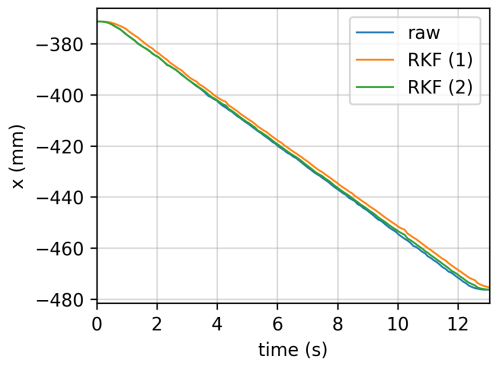


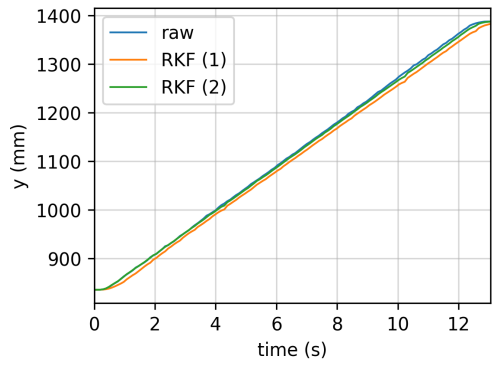
Figure 5.1: Reference path (line).

Regarding the external controller, the baseline PID controller and the RL-based supplementary controller require the specification of their respective sets of parameters. The PID control parameters consist of the control gains k_p, k_i, k_d and the time interval at which the pose correction signal is sent periodically. Through trial and errors, these parameters are tuned to the following values: $kp = \text{diag}(0.2, 0.2, 0.1)$, $k_i = \text{diag}(0.005, 0.005, 0.005)$,

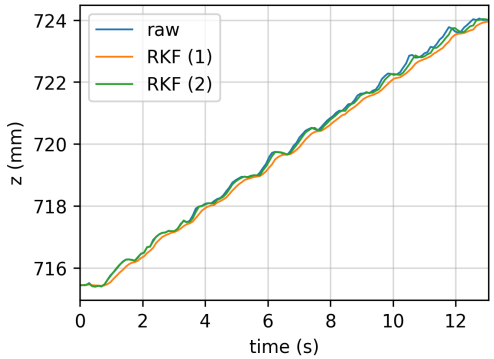
$k_d = \text{diag}(0.05, 0.05, 0.05)$. Moreover, the control interval is set to 80 ms based on the discussion provided in Section 4.3. The RL-based control parameters consist of the hyper-parameters of the TD3 algorithm, and the bounds of the state space and action space that scale the action and state to $[-1, 1]$ during RL training and then rescale back during policy rollout. All of the aforementioned parameters are listed in Table 5.2.



(a) x-direction



(b) y-direction



(c) z-direction

Figure 5.2: Comparison of raw and filtered estimated pose.

As mentioned in Chapter 2, the real-time estimated pose is filtered by a robust Kalman filter to attenuate noise. The plots from Figure 5.2 compare the raw and filtered signals. RKF (1) and RKF (2), denoted in Figure 5.2, are two versions of the same filter with different sets of parameters. The line following (position) experiment, discussed in Section 5.1, utilizes the RKF (1); whereas the (full pose) experiment, discussed later in Section 5.2, utilizes the RKF (2). RKF (1) has better noise attenuation capability but introduces a

noticeable phase delay. The phase delay causes a transient phase in the system response, as discussed later in Section 5.1.4. In contrast, RKF (2) prioritizes minimizing phase delay over noise attenuation, not causing a transient phase in the system response.

In the line following (position) experiment, three experimental runs are carried out. The first run serves to assess the performance of the baseline PID controller. For this purpose, ten experimental episodes are executed without the RL-based supplementary controller. The execution of multiple episodes accounts for the variance of data across episodes. The second run serves to train the supplementary controller via reinforcement learning, as discussed in Chapter 3, which requires up to 100 episodes. The third and final experimental run evaluates the performance of the supplementary reinforcement learning-based pose correction method. Ten experimental episodes are carried out like the first run, this time with the baseline PID controller supplemented by the RL-based controller. Note that the choice in the number of episodes for each experimental run is based on intuition, trial and error and learning performance.

5.1.1 RL Policy Training Assessment

Recall that the policy network is trained via the TD3 algorithm as described in Algorithm 3. The training process consists of 20 exploratory episodes and 100 training episodes, each containing approximately 140 time steps. The effectiveness of the RL training is assessed via the learning curve as shown in Figure 5.3. The learning curve is a line plot of the episodic return as a function of the number of training episodes completed. The training process is quite oscillatory, i.e., the return signal swings up and down at a high frequency. However, clear trends could still be identified by smoothing out the curve with a moving average filter. From episodes 0 to 60, the episodic returns display an upward trend, peaking approximately at episode 60. From episodes 60 to 100, the episodic returns show a downward trend that flattens near the end. The level-off of the learning curve indicates that

Common Parameters		
Control interval	T_c	80 ms
PID Parameters		
Proportional gain	k_p	diag$(0.2, 0.2, 0.1)$
Integral gain	k_i	diag$(0.005, 0.005, 0.005)$
Derivative gain	k_d	diag$(0.05, 0.05, 0.05)$
TD3 Hyperparameters		
State Max	-	[-300, 1500, 500]
State Min	-	[-600, 700, 800]
Action Max	-	[0.50, 0.50, 0.50]
Action Min	-	[-0.50, -0.50, -0.50]
Actor Network	-	MLP[*6*, 60, 60, *3*]
Critic Network	-	MLP[*9*, 120, 120, *1*]
Activation Function	-	ReLU
Discount rate	γ	0.98
Policy delay	d	2
Actor learning rate	α_ϕ	$1e^{-3}$
Critic learning rate	α_θ	$1e^{-3}$
Batch size	-	100
Target update rate	τ	$5e^{-3}$
Exploration noise	ϵ	$\mathcal{N}(0, \sigma = 0.1)$
Smoothing noise	η	$\mathcal{N}(0, \tilde{\sigma} = 0.2)$
Smoothing noise clip	c	0.5
Optimizer	-	Adam
Buffer size	-	100000
Gradient steps	G	800
Warmup noise	-	$\mathcal{N}(0, \bar{\sigma} = 0.05)$
Warmup episodes	-	20
Training episodes	-	100

Table 5.2: Hyperparameters of TD3 algorithm for line following (position) experiment.

the policy optimization reached a local minimum. Throughout the training progression, a snapshot of the policy is saved at the end of every episode. To find the best policy, it suffices to get the snapshot policy at the peak of the learning curve.

The pattern above indicates that the TD3 algorithm has effectively improved the policy throughout the first half of the training process. For the second half, the gradual decrease of episodic returns can be explained by the phenomenon of overtraining. Indeed, the inherent instability of TD3, a temporal difference learning algorithm, can steer the policy to perform better or worse arbitrarily [32]. Such instability entails that different instances of the same training process can yield policies of divergent performance levels. Therefore, the effectiveness assessment of the training algorithm usually requires many trials of the training process to account for the variance. In this research, only one training trial is carried out because each trial takes a long time to complete, making multiple trials unfeasible considering the available time frame.

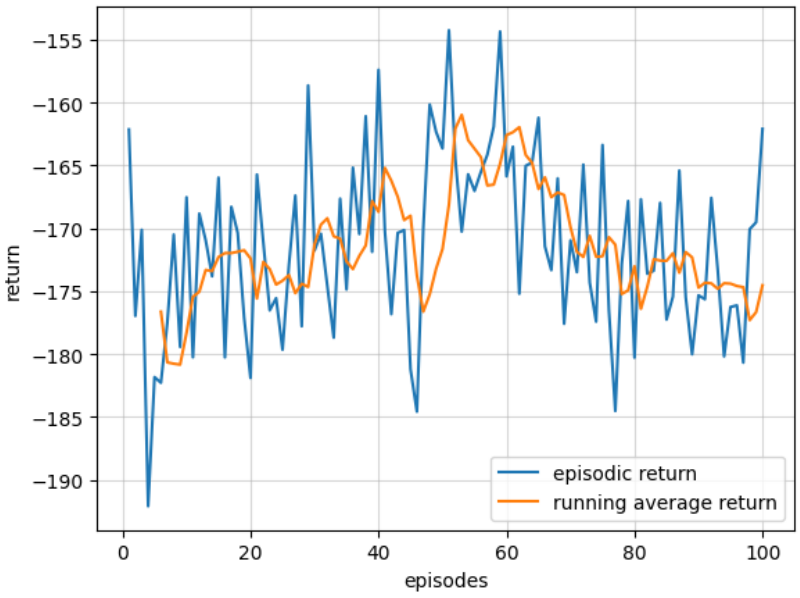


Figure 5.3: Learning curve for the line following (position) task over 100 episodes.

5.1.2 Path Error Data Distribution

Deriving any conclusion from the real-time estimates path errors necessitates finding the confidence interval of the estimated path errors. Therefore, the distribution of the deviations from the mean path error at each time step must be modelled. The target data consists of the path errors acquired throughout the ten experimental episodes with the rollout of the RL-based supplementary controller. Based on the histogram shown on the left of Figure 5.4, the deviations of the path errors in all Cartesian directions seem to follow a symmetric bell-shaped distribution. While a bell shape does not necessarily imply a normal distribution, normality tests can offer more insight into the target data distribution. One such test involves the quantile-quantile (Q-Q) plot, which plots the quantiles of the target distribution against those of a reference distribution, the normal distribution. If the target data is normally distributed, the dots of the Q-Q plot should lie in a straight line. Based on the plots shown on the right of Figure 5.4, the quantiles mostly lie on a straight line except at the extremities. Indeed, the extreme quantiles of the target distribution contain more extreme data than the normal distribution. This observed aberration might compensate for the fact that an actual data distribution cannot stretch to infinity like a normal distribution. Alternatively, the aberrations might simply be an inherent characteristic of the target distribution. Nevertheless, since the aberrations are relatively small and local to only one extremity, a normal distribution is still a good approximation of the target data distribution.

As a standard practice, the confidence interval of 95% probability is a conservative estimate of the range of path errors. Based on the normal distribution, 95% probability corresponds to the 97.5th percentile point, roughly 1.96 times the standard deviation σ of the mean. The target data's confidence intervals of the path errors are listed in Table 5.3. These confidence intervals are to be considered when calculating the path accuracy.

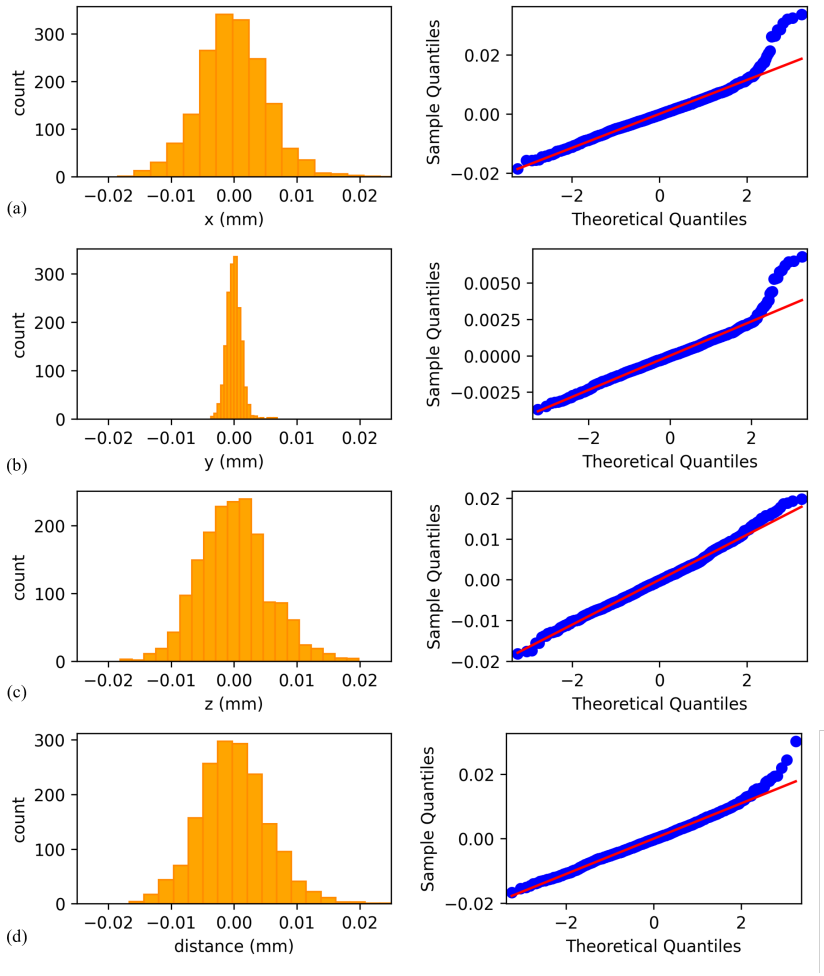


Figure 5.4: *Left*: Histogram of the mean path error deviations under control configuration (3) (baseline PID controller supplemented with RL-based controller), across ten performance assessment episodes; *Right*: Q-Q plot comparing the sampled data to a normal distribution; (a) X-direction, (b) Y-direction, (c) Z-direction, (d) 3D distance.

5.1.3 Control Performance Comparison

Recall, from Chapter 1, that the design goal of the supplementary controller is to act in parallel with the baseline controller to outperform the latter in terms of accuracy enhancement. Intuitively speaking, better path accuracy corresponds to smaller path errors. Since path errors are given as time series, the mean absolute error (MAE) is used to assess the mean path accuracy across experimental episodes. MAE is calculated as the arithmetic

95% Confidence Intervals	
x	$\pm 1.131 \times 10^{-2}$ mm
y	$\pm 2.317 \times 10^{-3}$ mm
z	$\pm 1.087 \times 10^{-2}$ mm
distance	$\pm 1.078 \times 10^{-2}$ mm

Table 5.3: Confidence intervals of path errors corresponding to the performance assessment run of the RL-based supplementary controller.

average of the absolute errors, given by

$$\text{MAE} = \frac{\sum_i^N |e_i|}{N} \tag{5.1}$$

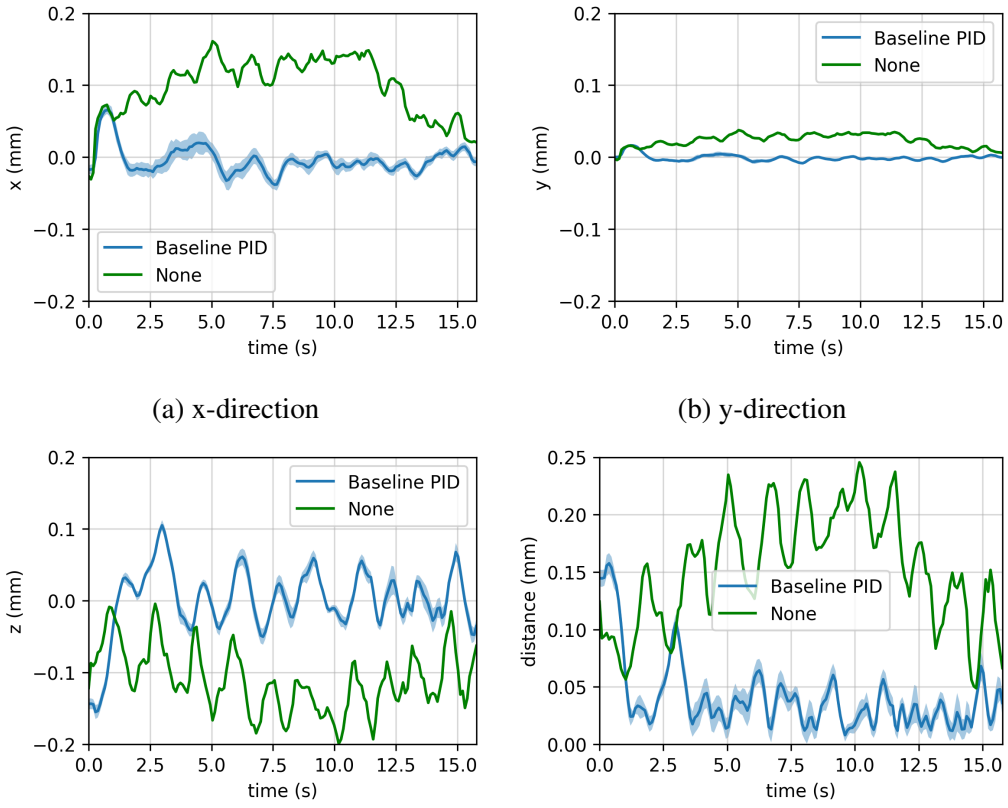
where e_i is the path error at a particular time instance, and N is the number of data points per episode. In addition, because the limiting case often defines performance, the maximum error is used to quantify the empirical path accuracy of an experimental episode. Since multiple episodes of performance assessment are conducted, the performance metrics are the average MAE and the maximum of the maximum error across all episodes. Conservatively, the path accuracy would be calculated by summing the maximum error, the volumetric accuracy of the C-Track optical sensor of ± 0.0025 mm, and the confidence intervals shown in Table 5.3. In the scope of this research, for the sake of simplicity, *the path accuracy is computed simply as the maximum error*.

Based on those above three experimental runs, three corresponding control configurations are evaluated:

(1) **No external controller**: This configuration has no external intervention of the robot's motion. Therefore, the robot follows its intrinsic motion.

(2) **Baseline PID controller**: In this configuration, an external controller corrects the robot's intrinsic motion in real-time. Its control algorithm consists simply of the baseline PID controller.

(3) **Baseline PID controller supplemented with RL-based controller**: Acting in parallel with the baseline PID controller, the RL-based controller is added in this configuration.



(a) x-direction

(b) y-direction

(c) z-direction

(d) 3D distance

Figure 5.5: Comparison of path error between control configurations (1) and (2), averaged across ten assessment trials; Shaded area represents a conservative estimate of the confidence interval.

The impact of the baseline PID is highlighted by comparing the path errors from experiments of control configurations (1) and (2), shown in the plots of Figure 5.5. While the robot is stationary at the start and end, the path errors are insubstantial due to the high repeatability of the M-20iA robot. From the beginning of the intrinsic motion, the robot deviates steadily from the linear path, reaching its peak at around the middle of the path, and then, the path error decreases gradually until the end point. The maximum path error

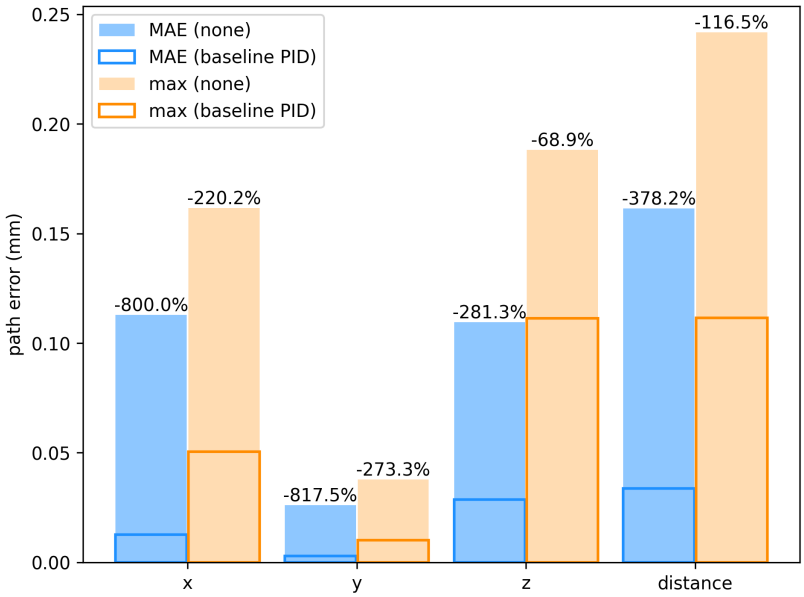
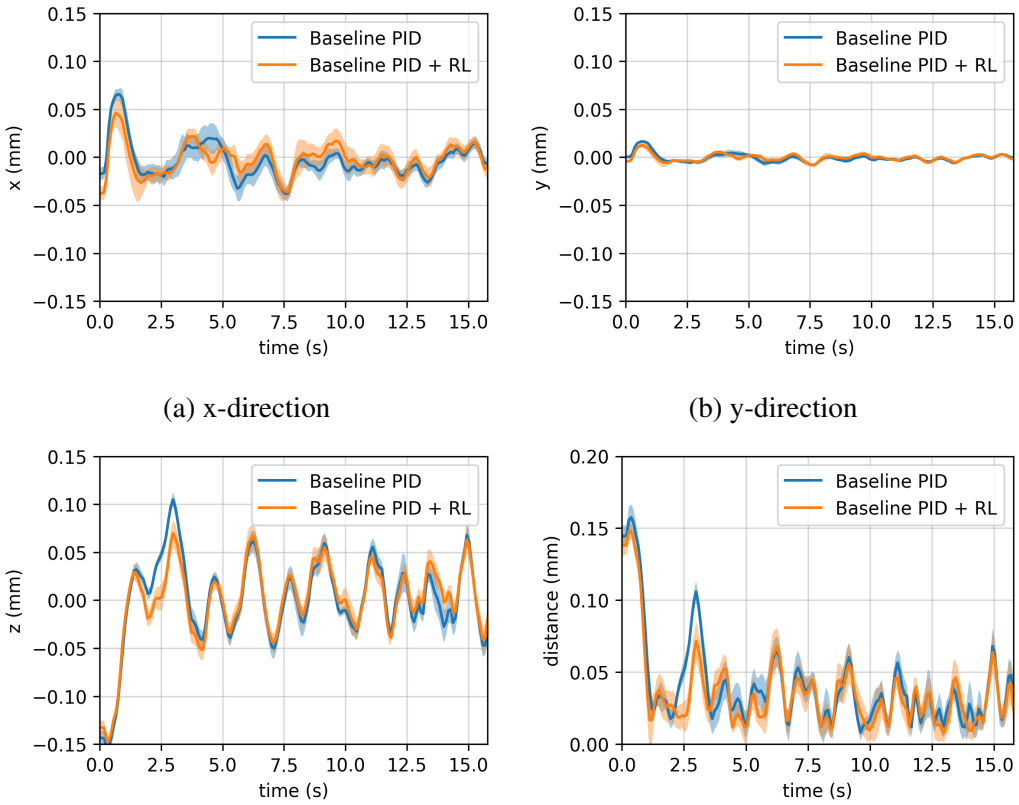


Figure 5.6: Comparison of average MAE and maximum path error between control configurations (1) and (2) across ten assessment trials (*starting from the 1.5 second mark*).

can reach up to 0.25 mm in terms of 3D distance. Across all directions in Cartesian space, the baseline PID controller significantly reduces path errors at all times.

The two configurations' performance metrics are compared in Figure 5.6. The performance metrics are calculated starting approximately from the 1.5 seconds mark, when the path error crosses the target (zero) for the first time, to ignore any effect from the initial error. It is observed that MAE and maximum error are reduced from 0.161 / 0.242 mm to 0.034 / 0.112 mm respectively in terms of 3D distance, corresponding to reductions of 378.2%/116.5%. While ignoring other sources of error, the path accuracy achieved by the baseline PID controller is determined as the maximum error, which is ± 0.112 mm. This result agrees with the benchmark study by [13], which achieved an accuracy of ± 0.20 mm.

The effect of the supplementation of the RL-based controller on top of the baseline PID is studied by comparing the average path error between the second and third configurations as shown in Figure 5.7. Note that the path error is averaged across the ten episodes of performance assessment. The RL-based controller appears to have no noticeable effect



(a) x-direction

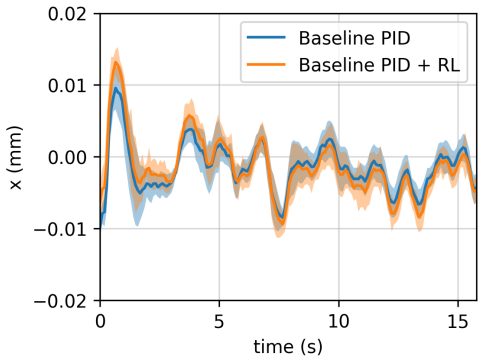
(b) y-direction

(c) z-direction

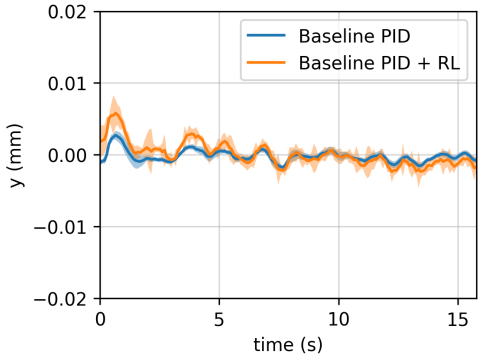
(d) 3D distance

Figure 5.7: Comparison of the path error between control configurations (2) and (3), averaged across ten assessment trials; Shaded area represents a conservative estimate of the confidence interval.

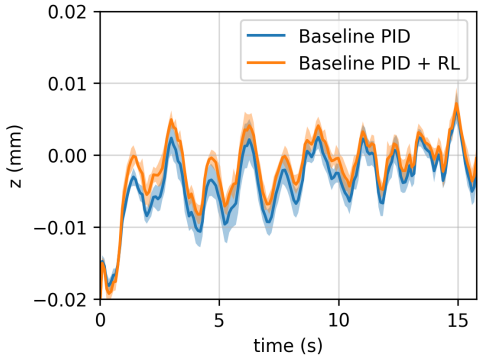
throughout most of the path motion. However, one could notice that the overshoots of path error, occurring before the 5 seconds mark, are significantly reduced across all Cartesian directions. The average control signal, shown in Figure 5.8, also reflects a more significant control effort contribution from the RL-based supplementary controller in the first 5 seconds of the path following task, matching the reduction of overshoots as mentioned above. Past the 5 seconds mark, the RL-based supplementary controller offsets the baseline control signal by a relatively constant minor amount, also matching its minor effect for the rest of the path.



(a) x-direction



(b) y-direction



(c) z-direction

Figure 5.8: Comparison of baseline and supplemented control signals under control configuration (3), across ten assessment trials; Shaded area represents a conservative estimate of the confidence interval

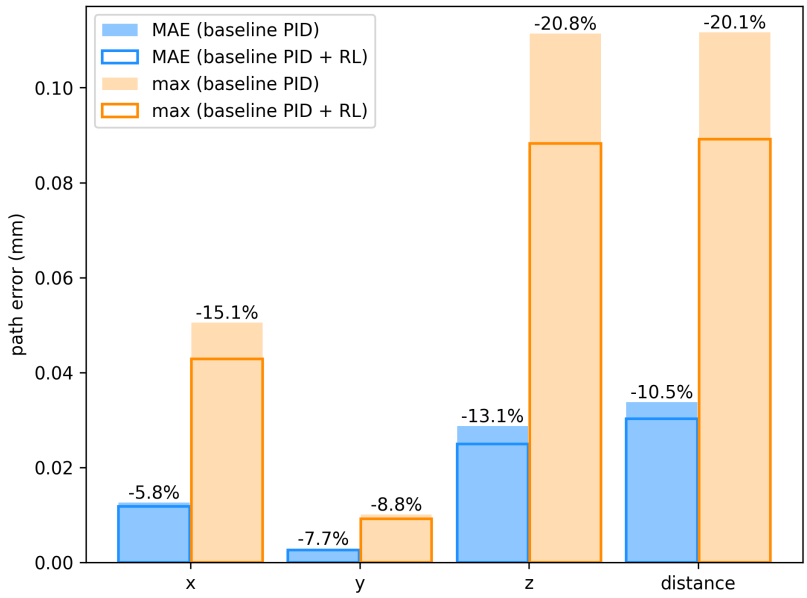


Figure 5.9: Comparison of average MAE and maximum path errors between control configurations (2) and (3), across ten assessment trials (*starting from the 1.5 second mark*).

A comparison of the performance metrics is illustrated in two column charts to quantify the reduction of overshoots. The chart shown in Figure 5.9 accounts for the effect of overshoots by showing the performance metrics computed beginning from the 1.5 second mark. In comparison, the chart in Figure 5.10 ignores the former by showing the metrics computed beginning from the 5.5 second mark. The first chart shows that both MAE and maximum error reduces from 0.034 / 0.112 mm to 0.030 / 0.089 mm, respectively, in terms of 3D distance, corresponding to reductions of 10.5%/20.1%. The second chart shows that the MAE reduces from 0.031 mm to 0.029 mm in the 3D distance, corresponding to a 5.7%. However, the maximum error increases from 0.077 mm to 0.078 mm, corresponding to an increase of 1.1%. While accounting for overshooting, the path accuracy achieved by supplementation of the RL-based control is given as the maximum error of ± 0.089 mm, improving from that of the baseline PID control by 20.1%.

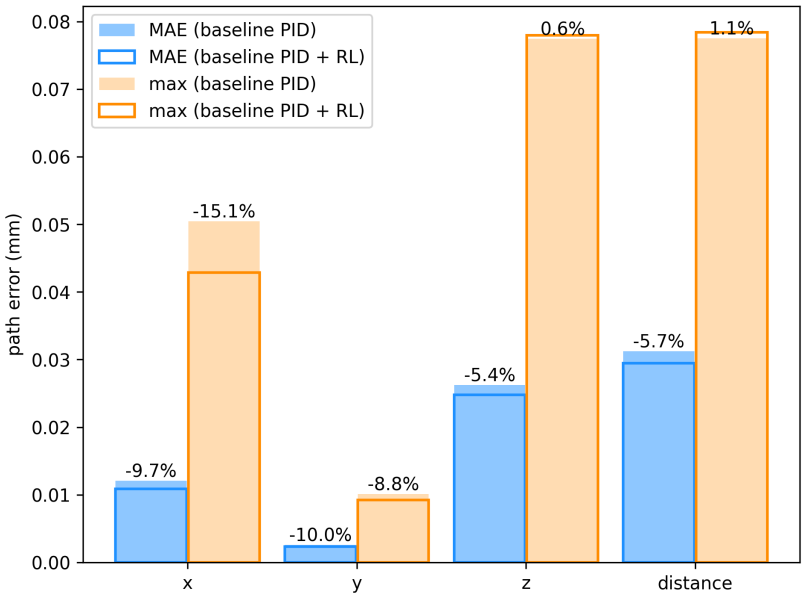


Figure 5.10: Comparison of average MAE and maximum path errors between control configurations (2) and (3), across ten assessment trials (*starting from the 5.5 second mark*)

5.1.4 Discussion

Note that the RL-based supplementary controller is trained specifically for the line following task characterized by the TP program parameters listed in Table 5.1. It likely would not yield the same performance if the task is altered. The amount of generalization of the RL-based controller remains to be investigated. Thus, as an assumption, any result derived from the line following experiment (position) cannot be generalized for other task paths.

As expected, the rollout of the baseline PID controller yields tremendous path accuracy improvement from the intrinsic motion without external control. From the summary Table 5.4, MAE and max error improve across all the Cartesian directions, amounting to an overall 116.5% max error improvement in 3D distance.

On the other hand, the effect of supplementation with RL-based controller is studied under two time frames. The first time frame, starting from the 1.5 second mark, includes the transient period where overshoot occurs. In the first time frame, path accuracy improvements are seen across all Cartesian directions. The summary table 5.4 shows that

Comparison between control configurations 1 & 2						
-	MAE			Max Error		
-	without PID	with PID	% change	without PID	with PID	% change
x (mm)	0.1129	0.0125	-800.0	0.1616	0.0505	-220.2
y (mm)	0.0262	0.0029	-817.5	0.0377	0.0101	-273.3
z (mm)	0.1095	0.0287	-281.3	0.1881	0.1114	-68.9
distance (mm)	0.1615	0.0338	-378.2	0.1615	0.1116	-116.5
Comparison between control configurations 2 & 3						
-	MAE			Max Error		
-	without RL	with RL	% change	without RL	with RL	% change
x (mm)	0.0125	0.0118	-5.8	0.0505	0.0429	-15.1
y (mm)	0.0029	0.0026	-7.7	0.0101	0.0092	-8.8
z (mm)	0.0287	0.0250	-13.1	0.1114	0.0882	-20.8
distance (mm)	0.0338	0.0302	-10.5	0.1116	0.0892	-20.1

Table 5.4: Comparison of the performance metrics between, from the 1.5 second mark, control configurations for the line following experiment.

the supplementation of the RL-based controller improves accuracy by 20.1% in the first time frame. In comparison, the second time frame, beginning from the 5.5 second mark, consists of the period where the path deviation more or less reaches a steady state. In the second time frame, path accuracy not only does not improve but even sees minor (1%) deterioration, which can be neglected.

From the observations above, the RL-based controller drastically improved transient performance but had a negligible effect on the steady-state performance. The transient phase may stem from the phase delay induced by the robust Kalman filter. The poor steady-state performance of the proposed RL-based controller stems possibly from the exploration-exploitation dilemma of the TD3 algorithm. Regarding TD3, exploration is done via the injection of Gaussian noise into the control signal. It is suspected that this exploration noise is not suitable for very fine path stabilization since noisy control signal often leads to chattering. Indeed, the exploration policy worsens the accuracy such that it seldomly stumble upon a more rewarding state transition, which leads to failure to improve the policy,

i.e., the policy is stuck in a local minimum. If one decreases the amplitude of the noise or even eliminates it, then the lack of exploration also leads to failure to improve the policy.

In contrast, TD3 deals with transient performance better since it consists of more coarse motion control, i.e. the magnitude of the path errors is more significant, resulting in less noise sensitivity. Potential solutions to improve steady-state performance include fine-tuning the exploration noise magnitude, proposing a more suitable alternative exploration policy, scheduling the exploration noise throughout the training process and so forth. However, these potential solutions are open problems outside this research's scope.

Another possible explanation is that the signal from the C-Track sensor is too noisy. Indeed, one drawback of optical sensors is their inherent noisiness [19]. The noisy signal might cause the approximate action-value function (critic) to be inaccurate. Since the policy (actor) is optimized based on the critic, the former's accuracy is limited by the latter. Alternatively, episode-specific aberrations can also drastically hinder the effectiveness of TD3. Any episodic RL algorithm hinges on the fact that the dynamics of the environment do not alter from episode to episode. In the case of industrial robots, the same Cartesian pose does not correspond to a unique position for the joint bearings and gears due to slippage and gear ratio. Therefore, irregularities of the joint bearings and gears constitute an episode-specific aberration. The above hardware-related issue cannot be solved via modifying the control algorithms.

5.2 Experiment 2: Line Following (Full Pose)

The line following experiment (full pose) is the same experiment presented in Section 5.1 except that the orientation changes at a constant rate from the start point to the end point. More precisely, the ZYX Euler angles vary linearly as a function of the positional path progression. The determination of the desired Euler angles based on the current path

progression is discussed in Section 2.2. The addition of orientation in the pose correction signal requires the orientation to be different from the start point to the end point, as reflected in the TP parameters shown in Table 5.5. For the sake of comparison with the position experiment, the start/end points and the speed remain the same.

TP Parameter	Value
Start point	$[-371\text{ mm},\ 836\text{ mm},\ 715\text{ mm},\ 179°,\ -3.93°,\ 89.2°]$
End point	$[-476\text{ mm},\ 1386\text{ mm},\ 724\text{ mm},\ 171°,\ 12.8°,\ 87.4°]$
Speed	50 mm/s

Table 5.5: Parameters of TP program for the line following (full pose) experiment.

Concerning the external controller, the parameters are shown in Table 5.6. The parameters of the Baseline PID control are changed to account for the coupling effect between position and orientation pose correction. The control gains for orientation are rather large due to the conversion from radian to degree. The control interval remains at 80 ms. A single policy network handles the position and orientation control. Aside from the bounds of state and action spaces, the TD3 parameters stay the same.

Even though the line following (full pose) is a follow-up to the position experiment, there is one key difference besides the addition of orientation control. This experiment utilizes a robust Kalman filter (RKF (2) shown in Figure 5.2) that is re-tuned for reduced phase lag but less noise suppression, resulting in a drastic change in more variance and oscillation in the path error. Thus, the comparison between experiments also requires redoing of the position experiment under the new Kalman filter, but unfortunate circumstances prevent further experimentation. Due to the relocation of the FANUC robot, there is not enough time to complete this experiment. The control parameters are not well-tuned enough to yield satisfactory control performance. The experimental results demonstrate a non-ideal effect on path accuracy, but there are included nonetheless for completeness.

Common Parameters		
Control interval	T_c	80 ms
PID Parameters		
Proportional gain	k_p	diag$(0.4, 0.3, 0.4, 17.19, 8.59, 25.78)$
Integral gain	k_i	diag$(0.005, 0.005, 0.05, 1.15, 1.15, 1.15)$
Derivative gain	k_d	diag$(0.0005, 0.0005, 0.0005, 0.2865, 0.2865, 0.2865)$
TD3 Hyperparameters		
State Max	-	[-300, 1500, 500, 3.20, 0.30, 1.60]
State Min	-	[-600, 700, 800, 2.90, -0.15, 1.40]
Action Max	-	[0.50, 0.50, 0.50, 0.002, 0.002, 0.002]
Action Min	-	[-0.50, -0.50, -0.50, -0.002, -0.002, -0.002]
Actor Network	-	MLP[*12*, 100, 100, *6*]
Critic Network	-	MLP[*18*, 150, 150, *1*]
Activation Function	-	ReLU
Discount rate	γ	0.98
Policy delay	d	2
Actor learning rate	α_ϕ	$1e^{-3}$
Critic learning rate	α_θ	$1e^{-3}$
Batch size	-	100
Target update rate	τ	$5e^{-3}$
Exploration noise	ϵ	$\mathcal{N}(0, \sigma = 0.1)$
Smoothing noise	η	$\mathcal{N}(0, \tilde{\sigma} = 0.2)$
Smoothing noise clip	c	0.5
Optimizer	-	Adam
Buffer size	-	100000
Gradient steps	G	800
Warmup noise	-	$\mathcal{N}(0, \bar{\sigma} = 0.05)$
Warmup episodes	-	20
Training episodes	-	32

Table 5.6: Hyperparameters of TD3 algorithm for line following (full pose) experiment.

5.2.1 RL Policy Training Assessment

Like the previous experiment, the training process consists of 20 exploratory episodes, but only 32 training due to the disruptions of the experiment. From the learning curve shown in Figure 5.11, the training process remains oscillatory, requiring noise filtering to notice the trend. The trend is that the episodic return increases with the number of training episodes, indicating the effectiveness of the RL process. However, the observation of the learning convergence will require more learning episodes. As before, the best policy is the policy with the highest episodic return, which happens in the last training episode for this experiment. Overall, the line following experiment's (full pose) policy training does not differ significantly from the position experiment. Hence, any further analysis follows what is discussed in Section 5.1.1.

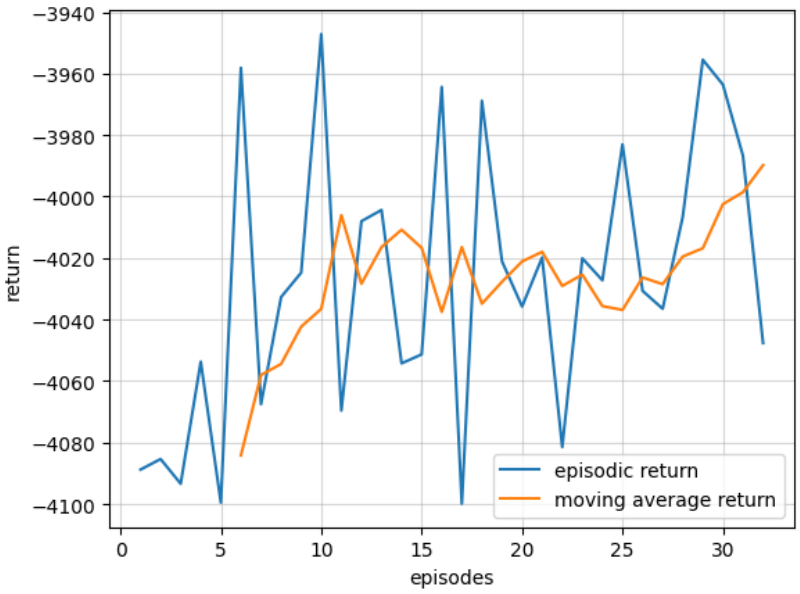


Figure 5.11: Learning curve for the line following (full pose) task over 32 episodes.

5.2.2 Path Error Data Analysis

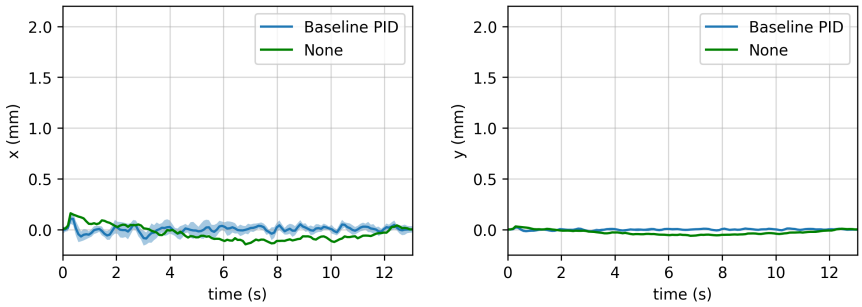
Based on the same analysis done in Subsection 5.1.2, the deviation from the mean path error at each time instance is assumed to follow the normal distribution. Thus, the confidence interval of the path error, in the line following (full pose) experiment, remains to be the 97.5th percentile point as appeared in table 5.7. Note that the confidence intervals are larger in this experiment compared to the previous one. This discrepancy can be attributed to the accuracy deterioration with the inclusion of orientation correction. Indeed, an added complexity in the dimension of control targets intuitively leads to the accumulation of errors induced by interactions between dimensions. Moreover, the higher variance could also be caused by the re-tuning of the Kalman filter (for C-Track pose estimation), mentioned previously, which prioritizes phase shift reduction instead of noise attenuation.

95% Confidence Intervals	
x	$\pm 4.831 \times 10^{-2}$ mm
y	$\pm 9.078 \times 10^{-3}$ mm
z	$\pm 5.595 \times 10^{-2}$ mm
3D distance	$\pm 4.743 \times 10^{-2}$ mm
w	$\pm 2.171 \times 10^{-2}$ °
p	$\pm 9.817 \times 10^{-3}$ °
r	$\pm 2.400 \times 10^{-2}$ °
eq.-angle	$\pm 1.746 \times 10^{-2}$ °

Table 5.7: Confidence intervals of path errors corresponding to the RL-Based supplementary controller assessment run for full pose experiment.

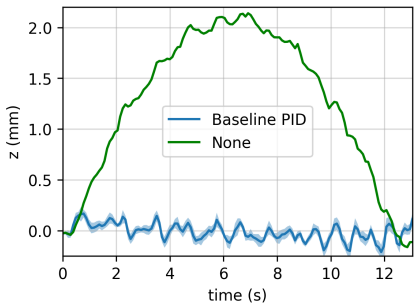
5.2.3 Control Performance Comparison

As discussed in Subsection 5.1.3, the RL-based supplementary control performance metrics consist of the average MAE and maximum path error across ten episodes of performance assessment. The maximum path error quantifies the path accuracy, accounting for the worse scenario. The same three control configurations are being compared: (1) no external control, (2) baseline PID controller, and (3) baseline PID controller supplemented

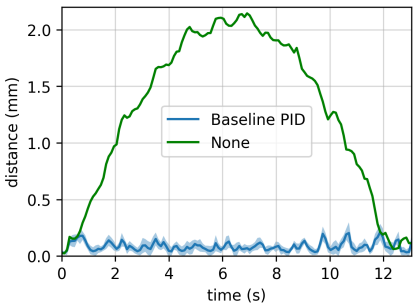


(a) x-direction

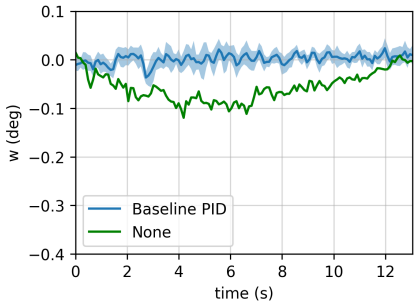
(b) y-direction



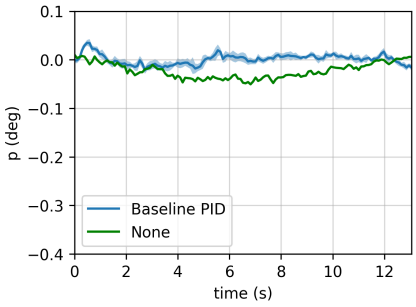
(c) z-direction



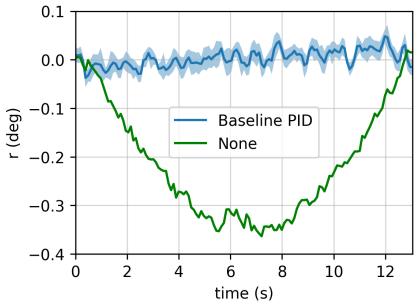
(d) 3D distance



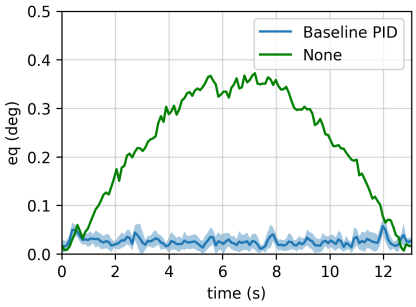
(e) x-rotation



(f) y-rotation



(g) z-rotation



(h) equivalent angle

Figure 5.12: Comparison of the position and orientation errors between control configurations (1) and (2), across ten assessment trials; (w, p, r corresponds to the ZYX Euler angles); Shaded area represents a conservative estimate of the confidence interval.

with RL-based controller.

The first comparison is made between configurations (1) and (2) to highlight the effect of the baseline PID controller. In Figure 5.12, the time-series plots compare the position and orientation errors, respectively, between control configurations (1) and (2). The plots show position and orientation errors averaged across ten performance assessment episodes. In all Cartesian directions, the position errors are negligible when the robot is stationary at the beginning and end of the path, owing to the remarkable positioning repeatability of the M-20iA robot. While in motion without an external controller, the robot intrinsically deviates from the reference path, reaching maximum path error around the middle of the path. Likewise, the orientation errors peak midway through the path but vanish at the extremities. The position error, in terms of distance, is mainly contributed by the error in the z-direction. A more significant margin of error, induced by the minimal change in the z-position throughout the reference path, could explain the prominence of the deviation in the z-direction. Similarly, the orientation error, in terms of equivalent angle, is mainly contributed by the error around the z-axis (ZYX Euler angles). The equivalent angle is calculated based on the equivalent angle-axis representation [44]. With the baseline PID controller's rollout, the robot no longer deviates as before but rather oscillates around the reference path.

The column charts, shown in Figures 5.13 and 5.14, compare the performance metrics of the two control configurations. The deployment of the baseline PID control reduces the MAE (distance) by -1562%, from 1.40 mm to 0.084 mm. Meanwhile, the maximum error (distance) is reduced by 650%, from 2.15 mm to 0.286 mm. Compared to the position experiment, the achieved accuracy of ± 0.286 mm is much lower than the previous value of ± 0.112 mm. Since the inclusion of orientation control, the accuracy has worsened, as discussed previously in Section 5.2. Regarding orientation, the MAE (equivalent

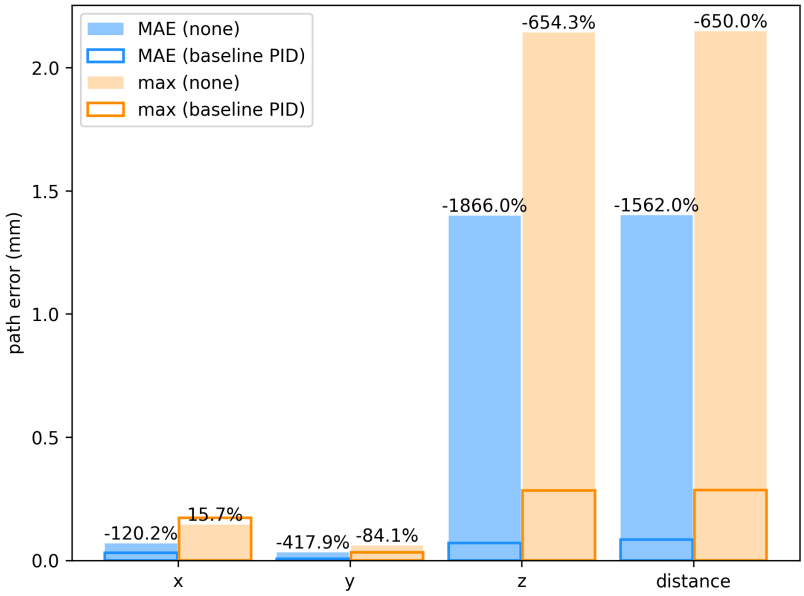


Figure 5.13: Comparison of average MAE and maximum position path errors between control configurations (1) and (2), across 10 performance assessment episodes (starting from the 1.5 second mark).

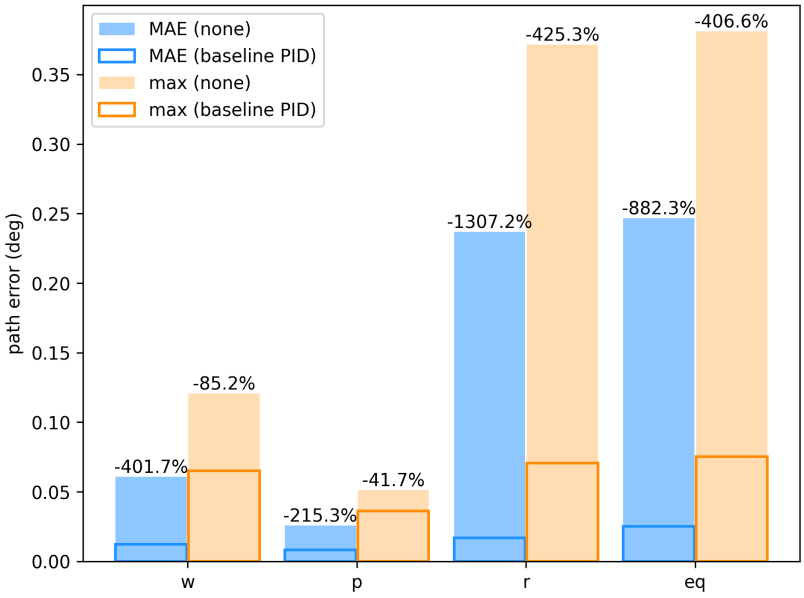
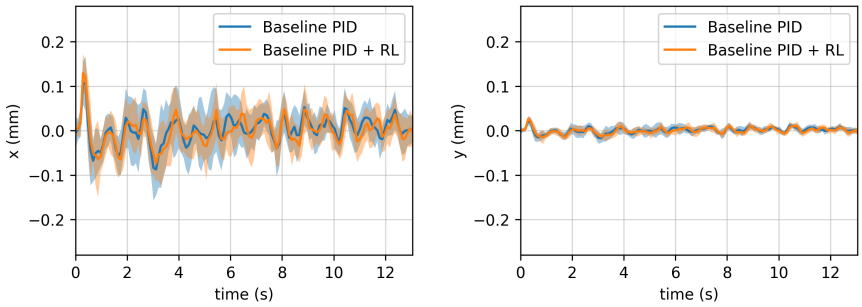
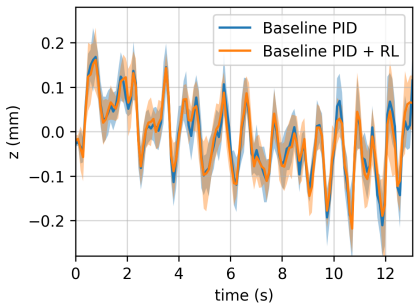


Figure 5.14: Comparison of average MAE and maximum orientation path errors between control configurations (1) and (2), across 10 performance assessment episodes (starting from the 1.5 second mark).

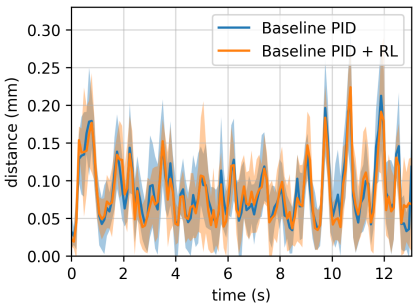


(a) x-direction

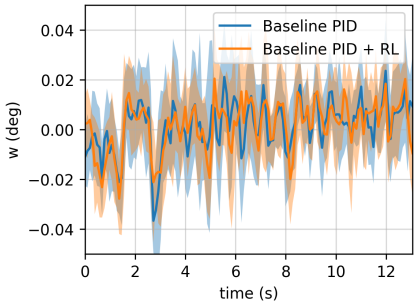
(b) y-direction



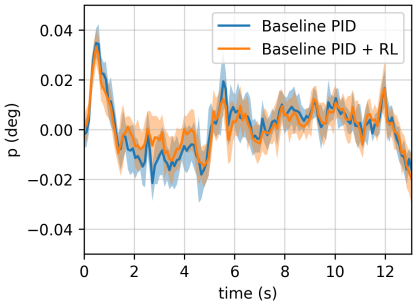
(c) z-direction



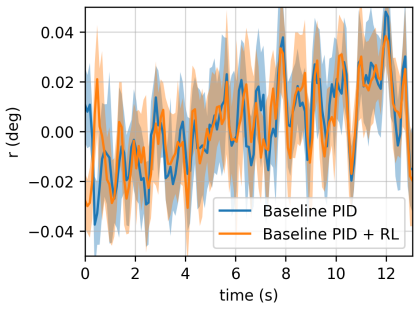
(d) 3D distance



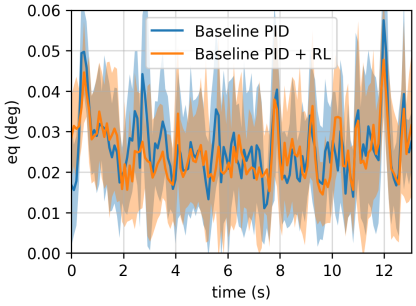
(e) x-rotation



(f) y-rotation

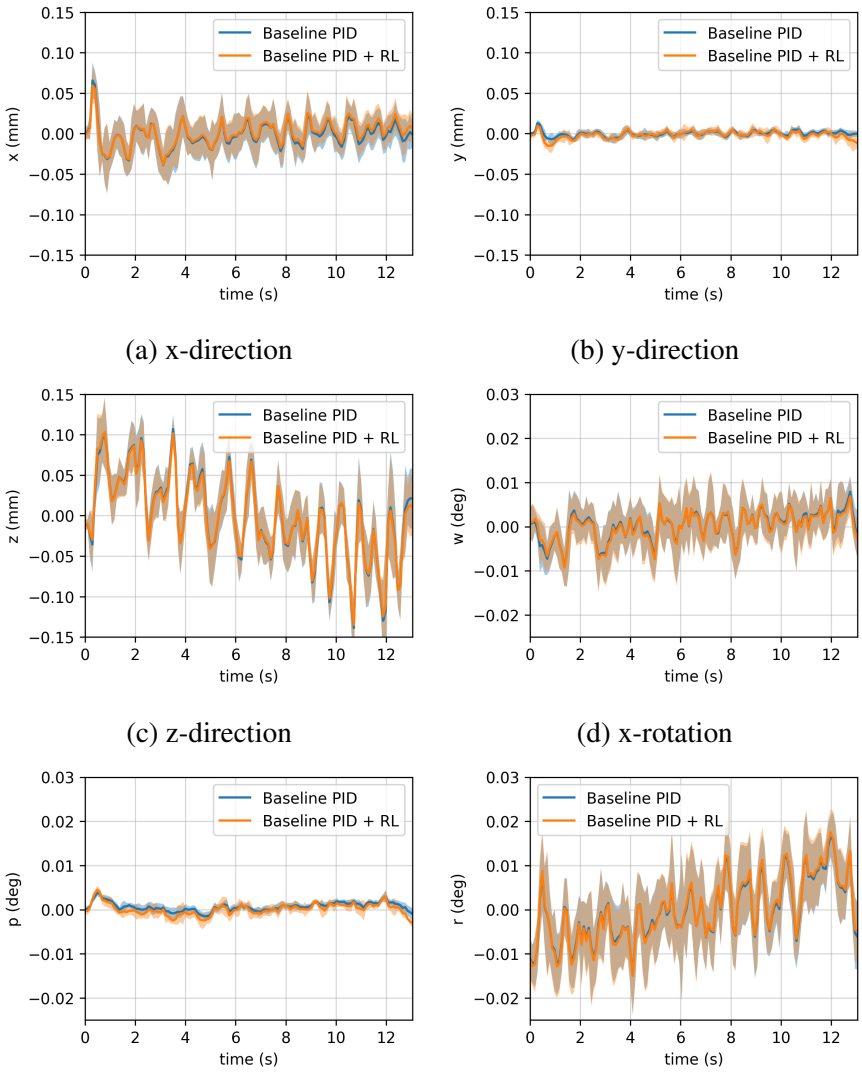


(g) z-rotation



(h) equivalent angle

Figure 5.15: Comparison of the position and orientation errors between control configurations (2) and (3), across ten assessment trials; (w, p, r corresponds to the ZYX Euler angles); Shaded area represents a conservative estimate of the confidence interval.



(a) x-direction

(b) y-direction

(c) z-direction

(d) x-rotation

(e) y-rotation

(f) z-rotation

Figure 5.16: Comparison of baseline and supplemented control signal under control configuration (3), across ten assessment trials; (w, p, r corresponds to the ZYX Euler angles); Shaded area represents a conservative estimate of the confidence interval.

angle) is reduced by 882.3%, from 0.247 to 0.0251°. The maximum error (equivalent angle) is reduced by 406.6%, from 0.381 to 0.0752°. As expected, the orientation accuracy improvement due to the deployment of the baseline PID control is substantial.

The second comparison is made between configurations (2) and (3) to show the effect of RL supplementation on the baseline PID controller. In Figure 5.15, the time series plots compare the position and orientation errors between the control configurations, showing that the RL-based supplementary control does not noticeably affect path stabilization. Indeed, the oscillations, with or without the supplementary controller, mostly coincide with minor differences in amplitude. Furthermore, as shown in Figure 5.16, the baseline and supplemented control signals overlap nearly perfectly with each other. The negligible control effort exerted by the RL-base supplementary controller corresponds to its minor effect on the path error.

The comparison of performance metrics is shown in the column charts from Figures 5.17 and 5.18. In terms of translation distance, the RL-based controller's supplementation reduces the MAE by 2.2%, from 0.0843 to 0.0824 mm, a negligible improvement justifiable by variance. However, the maximum error is increased by 4.2%, from 0.2 to 0.298 mm, a noticeable worsening. In terms of equivalent angle, the MAE is reduced by 4.9%, from 0.0251 to 0.0239°, a minor improvement. The maximum error is reduced by 9.2%, from 0.0752 to 0.0683°, also a minor improvement. The disparity of the supplementary control's effect on position and orientation accuracy can be attributed to either the incomplete experimentation mentioned previously in Section 5.2 or the inherent drawbacks of RL elaborated in Subsection 5.1.3.

5.2.4 Discussion

The deployment of the baseline PID controller does improve path accuracy substantially. The summary Table 5.8 shows that both position and orientation accuracy improve

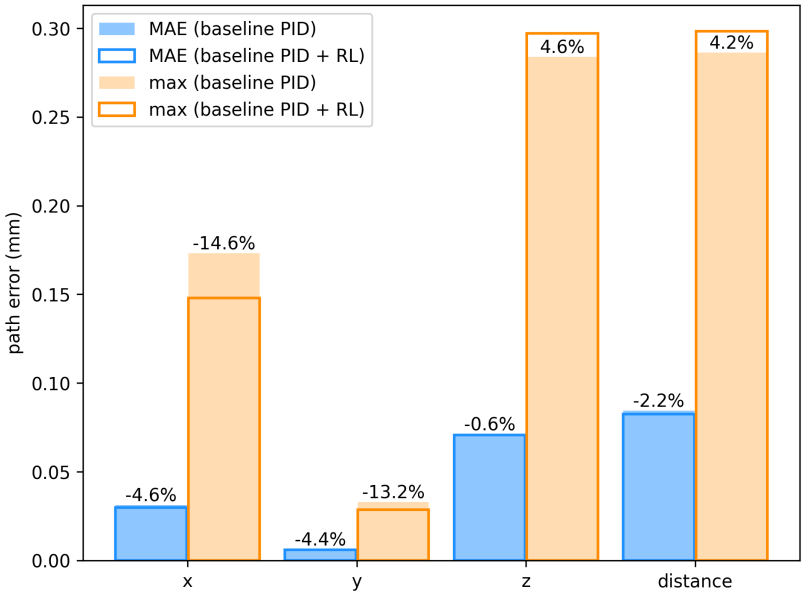


Figure 5.17: Comparison of average MAE and maximum position errors, with and without the RL-based supplementary control, across 10 performance assessment episodes (starting from the 1.5 second mark).

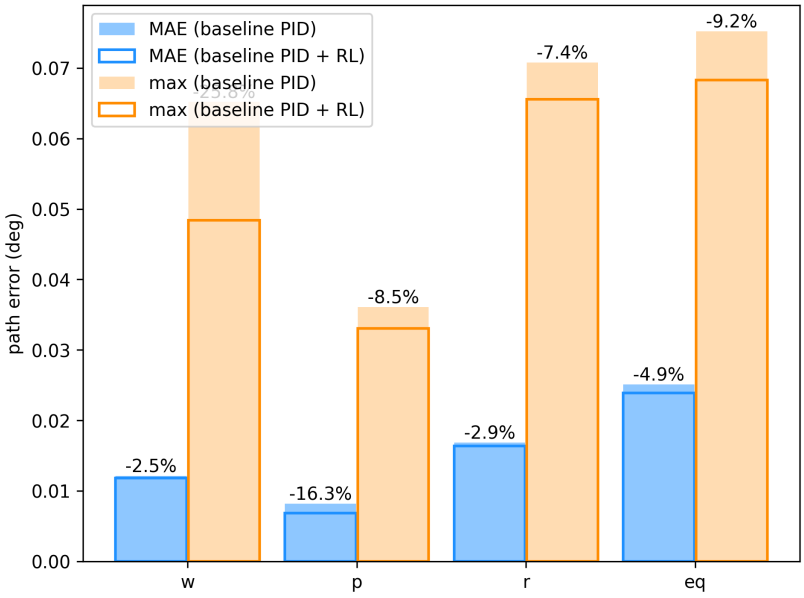


Figure 5.18: Comparison of average MAE and maximum orientation errors, with and without the RL-based supplementary control, across 10 performance assessment episodes (starting from the 1.5 second mark).

by 650% and 407% respectively. Accuracy improvement is observed across all directions and rotation axes, except for the x-direction, where the position accuracy decreases by 15%. An outlier in the data might explain this unexpected result.

Recall that the data of the position experiment demonstrate that the RL-based supplementary controller improves the transient performance but not the steady-state performance. In contrast, such a result is not found in the full pose experiment. The plots from Figure 5.17 and 5.18 do not display a transient phase of the path errors, i.e., the amplitudes of the oscillations stay constant at all time without the initial overshoot. The absence of a transient phase can be explained by the drastic reduction of phase delay of the re-tuned Kalman Filter discussed previously in Section 5.2. Therefore, only the steady-state performance can be evaluated. Like in the position experiment, the RL-based supplementary controller has a minor effect on the steady-state path accuracy. The summary table 5.8 shows a 4.2% worsening in position accuracy, and 9.2% improvement in orientation accuracy. An all-encompassing path accuracy improvement should be possible if a better set of control parameters is provided, as shown in Table 5.6.

Comparing the results from position and full pose experiments, one can observe that the accuracy achieved in the former is much better than that achieved in the latter. Differences between the experiments explain such a discrepancy, listed as follows:

- The inclusion of orientation adds more dimensions as control targets, leading to more errors due to interdimensional interaction.
- The robust Kalman filter is re-tuned for the full pose experiment to prioritize phase delay reduction over noise attenuation. Hence, the noisier reference signal might leads to a noisier control signal, resulting in more variance in the control performance.
- Regarding the pose estimated by the C-Track, the orientation signal is much noisier than the position signal. Within the policy network, the noise from the orientation

Comparison between control configurations 1 & 2						
-	MAE			Max Error		
-	without PID	with PID	% change	without PID	with PID	% change
x (mm)	0.0685	0.0311	-120.2	0.1460	0.1733	+15.7
y (mm)	0.0324	0.0063	-417.9	0.0609	0.0330	-84.1
z (mm)	1.3988	0.0711	-1866.0	2.1414	0.2839	-654.3
distance (mm)	1.4015	0.0843	-1562.0	2.1467	0.2862	-650.0
w (°)	0.0608	0.0121	-401.7	0.1207	0.0652	-85.2
p (°)	0.0258	0.0082	-215.3	0.0511	0.0361	-41.7
r (°)	0.2368	0.0168	-1307.2	0.3716	0.0707	425.3
eq.-angle (°)	0.2467	0.0251	-882.3	0.3808	0.0752	-406.6
Comparison between control configurations 2 & 3						
-	MAE			Max Error		
-	without RL	with RL	% change	without RL	with RL	% change
x (mm)	0.0311	0.0297	-4.6	0.1733	0.1481	+14.6
y (mm)	0.0063	0.0060	-4.4	0.0330	0.0287	-13.2
z (mm)	0.0711	0.0708	-0.6	0.2839	0.2971	+4.6
distance (mm)	0.0843	0.0824	-2.2	0.2862	0.2982	+4.2
w (°)	0.0121	0.0118	-2.5	0.0652	0.0484	-25.8
p (°)	0.0082	0.0069	-16.3	0.0361	0.0330	-8.5
r (°)	0.0168	0.0163	-2.9	0.0707	0.0655	-7.4
eq.-angle (°)	0.0251	0.0239	-4.9	0.0752	0.0683	-9.2

Table 5.8: Comparison of the performance metrics between, from the 1.5 second mark, control configurations for the line following experiment (full pose).

reference signal can spill over to affect both the position and orientation control signal.

- Based on the learning plot shown in Figure 5.11, the policy, in the full pose experiment, has yet to converge to a local minimum due to being trained for only 32 episodes. In the case of the position experiment, the policy converged at around 60 episodes.

5.3 Summary

In Chapter 5, the results of the two experiments are discussed. The two experiments are structured based on the task path: line (position) and line (full pose). For each experiment, three experimental runs contribute separately to assess the baseline controller, train the RL supplementary controller, and assess the Rl-based supplementary controller. From the experimental data gathered across experimental sets, the general qualitative observations are:

- The normal distribution well approximates the mean path error deviation.
- The RL-based controller improves transient performance by attenuating the magnitude of overshoots.
- The RL-based controller has a negligible effect on steady-state performance.

Note that the control performance is measured by the MAE and the maximum error of the path. These two performance metrics are chosen to gauge the amount of accuracy improvement. Regarding the line following (position) experiment, the path accuracy (defined as the max error) is improved by 20.1% under the supplementation of the RL-based controller compared to the rollout of only the baseline controller. Concerning the line following (full pose) experiment, the position accuracy is worsened by 4.2%, while the orientation accuracy is improved by 9.2%. The overall performance of the RL-based controller is discussed from a heuristic perspective. Chapter 6 will proposed potential improvement of the RL-based controller.

Chapter 6

Conclusion

Chapter 6 summarizes the research motivation, the problem statement, and the main results of this thesis. Furthermore, some perspectives on future works are discussed.

6.1 Research Summary

The motivation of the research comes from the inherent lack of accuracy in industrial robots, which hinders their use in manufacturing applications where accuracy is essential. Many robot calibration methods have been developed to improve the static accuracy of the robot significantly. However, robot calibration has little effect on path accuracy, which defines how precisely the robot can follow a predefined path. A promising approach to tackle path accuracy consists of real-time feedback control using an external visual sensor (visual servoing), which is the chosen approach in this thesis. Usually, the control problem to be solved is that of path following. However, the control scheme must be constrained by the robot's technical restrictions. Indeed, the robot's dynamics are often unknown, and the built-in control system is also not permitted to be modified by the end user. Therefore, any external intervention of the robot's motion needs to interface with software modules provided by the manufacturer of a given robot. Such software layer generally can take in

signals from an external device to offer real-time kinematic adjustment of the robot end effector in Cartesian space.

Based on the above technicalities stereotypical of most industrial robots, the control problem is reduced to a path error stabilization task. The robot in large is treated as a black box model to be stabilized by model-free control methods. The control framework is structured as two model-free controllers acting in parallel: a baseline PID controller and a supplementary controller. The baseline PID controller carries out the bulk of the error stabilization function, which has been proven to perform well in previous studies. In contrast, the supplementary controller supports the former to improve path accuracy further. The proposed supplementary controller is an RL-based policy (controller) trained with the TD3 algorithm.

Path stabilization experiments are carried out using a FANUC M-20iA robot and the C-Track dual-camera optical. Two experiments are carried out: line following (position) and line following (full pose). Custom control parameters are fine-tuned for each experiment to yield favourable results. Additionally, each experiment comprises three experimental runs used to assess the baseline controller, train the RL policy, and assess the RL-based supplementary controller. From the gathered experimental data, the following results are derived:

- **RL training assessment**: For all experiments, the offline TD3 algorithm proved to be effective at improving the supplementary controller (policy), which is shown via learning curves.
- **Path error deviation distribution modelling**: For all experimental sets, the data distributions are shown to be well approximated by the normal distribution. Hence, the confidence interval of the estimated path error can be properly quantified.
- **Performance metrics comparison**: The performance metrics consist of the MAE and max error of the path. Two comparisons are made: no external controller vs

baseline PID controller and baseline controller vs baseline PID controller supplemented with Rl-based controller. Across all experiments, the baseline controller, compared to no controller, improves path accuracy drastically. In contrast, the effect of the supplementation of RL-based controller is summarized as follows:

(1) Line following (position): The general observation is that transient performance is improved significantly, while steady-state performance is not affected. Quantitatively, the MAE and max error saw reductions of 10.5% and 20.1%, respectively, in terms of 3D distance. The resulting path accuracy is ± 0.0892 mm.

(2) Line following (full pose): Due to different experimental conditions, no transient phase of the path errors is observed. Therefore, only steady-state performance can be gauged, which shows no noticeable improvement in position and orientation accuracy. Note that this experiment has not been completed adequately due to the M-20iA robot's relocation.

The difference in transient and steady-state performances may be explained by deficiencies in the TD3 algorithm, which causes learning instability and local minima. Alternatively, hardware limitations may also play a role by causing episode-specific aberrations or noisy measurement signals.

6.2 Future Works

The poor steady-state performance of the RL-based supplementary controller motivates potential improvements to the RL algorithm:

- Scheduling of the magnitude of the exploration noise to smartly adapt to the need for exploration or exploitation.

- Using an exploration policy that is procedural instead of stochastic to eliminate the detriment of noisy control signals.
- Designing a better reward function that better reflects the path accuracy enhancement objective.
- Optimizing the hyperparameters of TD3 to improve learning stability and find a better policy.
- Training the policy with a different actor-critic algorithm to find a better policy.
- Switching to a model-based RL algorithm such as MBPO [45], or DayDreamer [46] to make learning more data-efficient. Learning a world model to predict the outcome of potential actions to minimize real-world trial and error.
- Proposing a control method utilizing discrete action space rather than continuous action space. RL algorithms for discrete action space typically does away with the exploration noise that is detrimental to fine motion control.
- (Full pose experiment) Implementing position and orientation control as separate neural networks. The primary reason for this decoupling is to prevent the relatively noisier orientation signal from spilling over to the position control signal.

Other than control algorithms, improvements in data acquisition, such as shorter sampling time and better noise filtering, can also be the subject of further research. Another possible extension is the experimentation with complex path geometries that are found in real manufacturing applications. Additionally, the path accuracy determination should be cross-validated by a third-party external sensor other than the C-Track. Moreover, the proposed control scheme and algorithm can also be tested in other types or brands of robots.

Bibliography

[1] "7 Common Types of Robotic Welding Processes and When They're Used." [Online]. Available: https://www.automate.org/blogs/7-common-types-of-robotic-welding-processes-and-when-they-re-used

[2] P. Heney, "Trelleborg showcasing Automated Fiber Placement technology at SPE Offshore Europe," Jul. 2019. [Online]. Available: https://www.sealingandcontaminationtips.com/trelleborg-showcasing-automated-fiber-placement-technology-at-spe-offshore-europe/

[3] "Batch Painting," May 2022. [Online]. Available: https://omnirobotic.com/applications/autonomous-robotic-painting/

[4] "When to Invest in Palletizing Robots | Genesis Blog," Feb. 2019. [Online]. Available: https://www.genesis-systems.com/blog/when-palletizing-robots-make-sense-for-your-facility

[5] E. Bisong, "The Exploration-Exploitation Trade-off." [Online]. Available: https://ekababisong.org//the-exploration-exploitation-trade-off/

[6] R. S. Sutton, "6.6 Actor-Critic Methods." [Online]. Available: http://incompleteideas.net/book/ebook/node66.html

[7] T. A. Khaled, O. Akhrif, and I. A. Bonev, "Dynamic Path Correction of an Industrial Robot Using a Distance Sensor and an ADRC Controller," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 3, pp. 1646–1656, Jun. 2021, conference Name: IEEE/ASME Transactions on Mechatronics.

[8] "Absolute accuracy industrial robot option," 2011. [Online]. Available: https://library.e.abb.com/public/0f879113235a0e1dc1257b130056d133/Absolute%20Accuracy%20EN_R4%20US%2002_05.pdf

[9] C. Mavroidis, J. Flanz, S. Dubowsky, P. Drouet, and M. Goitein, "High performance medical robot requirements and accuracy analysis," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 5, pp. 329–338, Oct. 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584598000222

[10] P. Li, "Visual Calibration, Identification and Control of 6-RSS Parallel Robots," phd, Concordia University, Apr. 2020. [Online]. Available: https://spectrum.library.concordia.ca/986930/

[11] B. Zhang, J. Wu, L. Wang, and Z. Yu, "Accurate dynamic modeling and control parameters design of an industrial hybrid spray-painting robot," *Robotics and Computer-Integrated Manufacturing*, vol. 63, p. 101923, Jun. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584519300043

[12] D. Chen, P. Yuan, T. Wang, Y. Cai, and L. Xue, "A Compensation Method for Enhancing Aviation Drilling Robot Accuracy Based on Co-Kriging," *International Journal of Precision Engineering and Manufacturing*, vol. 19, no. 8, pp. 1133–1142, Aug. 2018. [Online]. Available: https://doi.org/10.1007/s12541-018-0134-8

[13] T. Shu, S. Gharaaty, W. Xie, A. Joubair, and I. A. Bonev, "Dynamic Path Tracking of Industrial Robots With High Accuracy Using Photogrammetry Sensor," *IEEE/ASME*

Transactions on Mechatronics, vol. 23, no. 3, pp. 1159–1170, Jun. 2018, conference Name: IEEE/ASME Transactions on Mechatronics.

[14] "Manipulating industrial robots – Performance criteria and related test methods," International Organization for Standardization, Geneva, CH, Standard ISO 9283, 1998.

[15] Y. H. Andrew Liou, P. P. Lin, R. R. Lindeke, and H. D. Chiang, "Tolerance specification of robot kinematic parameters using an experimental design technique—the Taguchi method," *Robotics and Computer-Integrated Manufacturing*, vol. 10, no. 3, pp. 199–207, Jun. 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/073658459390055O

[16] B. Karan and M. Vukobratović, "Calibration and accuracy of manipulation robot models—An overview," *Mechanism and Machine Theory*, vol. 29, no. 3, pp. 479–500, Apr. 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0094114X94901309

[17] A. Nubiola and I. A. Bonev, "Absolute calibration of an ABB IRB 1600 robot using a laser tracker," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 1, pp. 236–245, Feb. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584512000816

[18] A. Nubiola, M. Slamani, A. Joubair, and I. A. Bonev, "Comparison of two calibration methods for a small industrial robot based on an optical CMM and a laser tracker," *Robotica*, vol. 32, no. 3, pp. 447–466, May 2014, publisher: Cambridge University Press. [Online]. Available: https://www.cambridge.org/core/journals/robotica/article/comparison-of-two-calibration-methods-for-a-small-industrial-robot-based-on-an-optical-cmm-and-4EADF266E900B2A5766C3F7B5FB5D0BB

[19] S. Gharaaty, T. Shu, A. Joubair, W. F. Xie, and I. A. Bonev, "Online pose

correction of an industrial robot using an optical coordinate measure machine system," *International Journal of Advanced Robotic Systems*, vol. 15, no. 4, p. 172988141878791, Jul. 2018. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1729881418787915

[20] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of Robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.4620010203. [Online]. Available: http://onlinelibrary.wiley.com/doi/abs/10.1002/rob.4620010203

[21] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, Apr. 2021, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364920987859

[22] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, Jun. 2006, conference Name: IEEE Control Systems Magazine.

[23] Y. M. Zhao, Y. Lin, F. Xi, and S. Guo, "Calibration-Based Iterative Learning Control for Path Tracking of Industrial Robots," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 5, pp. 2921–2929, May 2015. [Online]. Available: https://ieeexplore.ieee.org/document/6936298

[24] Y. P. Pane, S. P. Nageshrao, J. Kober, and R. Babuška, "Reinforcement learning based compensation methods for robot manipulators," *Engineering Applications of Artificial Intelligence*, vol. 78, pp. 236–247, Feb. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197618302446

[25] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996. [Online]. Available: http://ieeexplore.ieee.org/document/538972/

[26] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[27] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1505–1511, Jul. 2017, conference Name: IEEE Transactions on Control Systems Technology.

[28] T. Zhou, J. Tang, T. Shu, and W.-F. Xie, "Reinforcement learning based visual servoing of an industrial robot," 2022, [Manuscript submitted for publication].

[29] X. Wu, E. Zakeri, and W.-F. Xie, "Adaptive Robust Kalman Filter for Vision-based Pose Estimation of Industrial Robots," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, Dec. 2019, pp. 298–302.

[30] N. S. Altman, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992, publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/00031305.1992.10475879. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879

[31] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, second edition ed., ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.

[33] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *arXiv:1802.09477 [cs, stat]*, Oct. 2018, arXiv: 1802.09477. [Online]. Available: http://arxiv.org/abs/1802.09477

[34] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[35] A. Raffin, "Rl baselines3 zoo," https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

[36] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

[37] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[38] Creaform, "Technical Specifications: Dynamic Tracking | Creaform." [Online]. Available: https://www.creaform3d.com/en/ndt-solutions/dynamic-tracking/technical-specifications-dynamic-tracking

[39] "FANUC Robot M-10iA/M-20iA," 2015. [Online]. Available: https://www.fanuc.com/fvl/vn/product/catalog/RM-10iA(E)-07.pdf

[40] "FANUC ROBOT SERIES R-30iB CONTROLLER DYNAMIC PATH MODIFICATION USER'S GUIDE," 2014.

[41] X. Wu, "Robust Position-based Visual Servoing of Industrial Robots," Master's thesis, Concordia University, Aug. 2020. [Online]. Available: https://spectrum.library.concordia.ca/987524/

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[43] "Onnx runtime," https://onnxruntime.ai/, 2021, version: x.y.z.

[44] J. Craig, *Introduction to Robotics: Mechanics and Control*, ser. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005. [Online]. Available: https://books.google.ca/books?id=MqMeAQAAIAAJ

[45] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to Trust Your Model: Model-Based Policy Optimization," *arXiv:1906.08253 [cs, stat]*, Nov. 2021, arXiv: 1906.08253. [Online]. Available: http://arxiv.org/abs/1906.08253

[46] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "DayDreamer: World Models for Physical Robot Learning," Jun. 2022, arXiv:2206.14176 [cs]. [Online]. Available: http://arxiv.org/abs/2206.14176