POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Direct position control of an industrial robot based on an external tracking system

Tesi di Laurea Magistrale in
Mechanical Engineering - Ingegneria Meccanica

Authors: **Enrico Bruno Chiappini and Federico Pleuteri**

Student ID: 996962 - Enrico Bruno Chiappini and 988347 - Federico Pleuteri
Advisor: Prof. Andrea Maria Zanchettin
Co-advisors: Julian Markus Alexander Blumberg
Academic Year: 2022-23

Abstract

The use of industrial robots as machine tools offers great potential for increasing the flexibility of manufacturing companies. However, their low absolute pose accuracy in position and orientation, due to serial construction and flexibility of gearings, and the significant path errors, due to low structural stiffness, still prevent the development of new fields of application with high external loads and accuracy requirements.
To increase the absolute accuracy, the pose of the industrial robot can be directly detected and controlled in real-time by an external tracking system (e.g., camera- or laser-based). Still, the resulting accuracy strongly depends on the quality of the measurements as well as on the speed and robustness of the controller. Therefore, this thesis work deals with the development of a robust and highly-dynamic controller for the real-time compensation of a real industrial robot, on the basis of investigations of the control and tracking process.

Starting with an already existing communication protocol, experimental investigation on the robot were carried on to investigate the system characteristics and to identify the measurement noise. Thus, a Kalman filter was implemented, to improve tracking accuracy, together with a robust controller making use of PID controller parameters and fuzzy logic. Lastly, the method was validated both through simulation, exploiting the FANUC's software Roboguide, and through experiments, using the industrial robot FANUC M-900iB/700 available at the *Produktionstechnisches Zentrum (PTZ)* of TU Berlin.
The focus of the investigation has been on the achievable robot accuracies with and without the additional measurement technology and control strategies, driving a comparison of the different control algorithms and providing an outlook on the applicability of dynamic tracking and control using external cameras in industrial robots' processes.

Keywords: industrial robot, FANUC, tracking-system, Kalman filter, dynamic control

Abstract in lingua italiana

L'utilizzo di robot industriali come macchine utensili ha un grande potenziale per la flessibilità delle aziende manifatturiere. La loro scarsa accuratezza di posa in posizione e orientamento, però, dovuta alla fabbricazione in serie e alla resilienza degli ingranaggi, oltre che ai significativi errori di traiettoria per via della scarsa rigidezza strutturale, previene lo sviluppo di nuovi campi di applicazione, in particolare quelli ad alti carichi esterni e stringenti requisiti di precisione.
Per aumentare l'accuratezza assoluta, si può identificare e controllare la posa del robot industriale direttamente e in tempo reale, grazie a un sistema di tracciamento esterno (ad esempio basato su camere a infrarossi o laser). Nonostante ciò, l'accuratezza è fortemente legata alla qualità delle misurazioni e alla reattività e robustezza del sistema di controllo. Lo scopo di questo lavoro di tesi è, quindi, sviluppare un controllo robusto e altamente dinamico per la compensazione in tempo reale di un robot industriale, sulla base di investigazioni svolte sul processo di controllo e tracciamento.

Partendo da un sistema di comunicazione già sviluppato in precedenza, sono state svolte analisi sperimentali per investigare le caratteristiche strutturali del sistema e i disturbi sulle misure. Dopodiché, è stato implementato un sistema di controllo, costituito da un filtro di Kalman insieme a un controllo PID e una logica fuzzy, per poter migliorare l'accuratezza del tracciamento. Infine, il metodo è stato validato attraverso simulazioni svolte sul software Roboguide di FANUC, e su un robot industriale FANUC M-900iB/700 presente al *Produktionstechnisches Zentrum (PTZ)* di TU Berlin.
Il focus degli esperimenti è stato sull'accuratezza raggiungibile con e senza le tecnologie di tracciamento e il sistema di controllo, realizzando così una comparazione tra i diversi algoritmi utilizzati e fornendo una prospettiva sull'applicabilità del tracciamento e controllo dinamico, attraverso l'utilizzo di camere esterne, nei processi di robotica industriale.

Parole chiave: robot industriali, FANUC, sistema di tracciamento, filtro di Kalman, controllo dinamico

Contents

List of Abbreviations

AIFM Absolute Interferometer 11

AKF Augmented Kalman Filter 66, 67, 71, 75, 77, 78, 108, 109

AP Pose Accuracy 3, 8

ART Advanced Realtime Tracking 24, 26, 29

AT Path Accuracy 3, 8

BKF Bucy Kalman Filter 75, 76, 78, 108

CAD Computer-aided Design 8

CAM Computer-aided Manufacturing 8

CCD Charge-coupled Device 11

CFRP Carbon fiber-reinforced plastic 1

CNC Computerized Numerical Control 6

DH Denavit-Hartenberg 7, 13

DMS Direct Measurement Systems 9

DoF Degrees of Freedom 15, 17, 23, 26, 53

DPM Dynamic Path Modification 19, 22, 31, 43, 44, 46, 48, 63, 89

EKF Extended Kalman Filter 62, 66, 69, 74, 75, 77, 78, 79, 108, 109

EtherCAT Ethernet for Control Automation Technology 23

FaRoC Fanuc Robot Control 20, 22

IR Industrial Robot . ix, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 15, 16, 17, 22, 29, 30, 32, 52, 63, 66, 69, 70, 74, 89, 112, 113, 116, 117

ISO International Organization for Standardization 2, 3

IWF Institut für Werkzeugmaschinen und Fabrikbetrieb 4, 17, 24, 29

KCL KAREL Command Language 19

LTI Linear Time-Invariant 63, 64

LTV Linear Time-Varying 62

MFs Membership Functions 51, 52, 54, 57
MIMO Multiple Input Multiple Output 10

PID Proportional Integral Derivative 52, 70, 71
PTZ Produktionstechnisches Zentrum ..... i, iii, ix, 17, 18, 29

RMS Root Mean Squared 61
RP Pose Repeatability 8
RSI Remote Service Interface 20
RTFP Real-Time Feature Pack 23

SD Standard Deviation 75, 76, 78, 109
SISO Single Input Single Output 10

T-Mac Tracker Machine 23, 24
TCP Tool Center Point 3, 7, 8, 9, 12, 15, 20, 24, 26, 27, 63
TCP/IP Transmission Control Protocol/Internet Protocol 19, 20
TF Transfer Function ..... ix, 30, 31, 37, 38, 39, 40, 43, 44, 45, 46, 63, 113
TUB Technische Universität Berlin 17, 20, 29

UDP User Datagram Protocol 21
UDP/IP User Datagram Protocol/Internet Protocol 20
UKF Unscented Kalman Filter 62
USM User Socket Messaging 19, 20

VDI Verein Deutscher Ingenieure 2

List of Figures

List of Tables

Introduction

Today, it is challenging to envision the manufacturing process of any product without the incorporation of industrial robots (IRs), due to their versatility, production flexibility, and technological capabilities. According to the International Federation of Robotics [1], the annual installation of new IRs worldwide keeps growing and has passed the half a million units since 2021, with China far long at the lead. Their major application areas are handling, accounting for the 48%, welding, and assembling, while only the 1% of the installed robots are exploited for processing [1]. In addition, the number of application areas is continuously increasing, due to the development in control engineering and of new technologies, for example in robot-assisted machining processes, such as deburring, milling, and grinding, which can achieve high accuracy leveraging high-resolution force-torque sensors and force control [2].

In both the aerospace, shipbuilding, and the automotive industry, the growing demand for flexible production and lightweight materials, such as aluminum or fiber composite plastics, requires industrial companies to adapt their manufacturing strategies. Therefore, moving from a background of high batch sizes and low component variance, which allowed for the use of specialized special-purpose machines with high accuracy in the micrometer range, is gaining interest the use of industrial robots to match the current high demand of efficient large scale machining operations [3].
For example, the finishing of carbon fiber-reinforced plastics (CFRP) for the production of lightweight structures is a time-consuming step, especially in the aerospace industry, and the use of component-enclosing gantry machines makes the adaptation to variable component sizes or machining strategy slow and cumbersome [4].
Moreover, the focus is also on the reduction of emissions and carbon footprint, which implicates utilizing integral design for minimizing overlapping, thus increasing component sizes and, as a consequence, forcing the use of CFRP components in structural parts. In particular, in the case of aerospace industry, they can account for up to 50% of the structural weight. [3]. In addition to the large dimensions, the mechanical finishing of these parts is particularly demanding on low manufacturing tolerances, hence requiring special machine tools with a rigid structure and variable clamping devices.

It is, indeed, thanks to its flexibility and automation that an IR can play a crucial role in the manufacturing process and result advantageous over conventional machine tools, especially in small batch production [5]. IRs are open chain mechanisms of multi-links coupling configurations, providing coupling motion of joints between consecutive links [6]. VDI Guideline 2860, defines them as "*universally applicable motion machines with several axes, whose movements are freely programmable with regard to sequence of movements and paths or angles and are sensor guided if necessary*" [7]. In contrast to manipulators they can be controlled automatically, and differently from machine tools, they are not specialized for a specific task.
ISO 8373, instead, gives the following definition of IR: "*automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or fixed to a mobile platform for use in automation applications in an industrial environment*" [8]. Therefore, IRs offer a valid and inexpensive alternative to machining centers, thanks to their comparatively low purchase price, the unnecessary heavy-duty foundation, and the small footprint compared to the working area, which additionally can be further extended by means of motion platforms to cover large dimensional structures or perform time-parallel component processing [3].

For this reason, with the goal to increase labor productivity and reduce costs at a constant level of quality, automated IR are used for welding, laying, painting, and other operations requiring repetition and high precision. In fact, in the 21st century is in course the transition from the individual automation of technological operations, such as pre-processing, thermal cutting of sheets and profiles, and components' welding, to full automation and robotization of the production [9]. Automation plays, indeed, a pivotal role in cost management, simultaneously ensuring the attainment of high-quality standards and compliance with safety regulations across the entire spectrum of production processes [10]. To realize enhanced production output and manufacturing efficiency, it is necessary to implement automation solutions throughout the entire production chain, encompassing component fabrication, finishing, and painting stages. Therefore, for example in ships' constructions, robotic tools are applied in assembly-welding and building stocks, foundry, turning, and painting, in form of a manipulator with tools, such as grippers and welding or cutting equipment [9].

However, though the many advantages of IRs, the poor absolute accuracy, due to serial construction, gear backslash, and low structural stiffness [11], restrain from extending their application range. In particular, the focus of this work is on 6-axis jointed-arm robots, which are used in the main share of industrial robotic applications due to their flexibility and variability of the design.

Motivation of the work

At the beginning, IRs were thought of as simple supports for pick-and-place or spot welding applications, where repeatability is the main factor and absolute accuracy has less importance. Just in recent years, heavy-duty 6-axis IRs have become very interesting also for machining, but in many fields, for example the aerospace industry, it remains still the exception due to really strict tolerances. In milling applications in particular, the path fidelity of the IR is decisive for the quality of the machining result.
Therefore, various approaches and methods are applied to increase the accuracy of IR, exploiting model-based approximation methods or compensation strategies. These models deal with many nonlinearities, such as friction or reversal effects in the gears, which need to be detected and compensated, with suitable control strategies, by using internal or external measurement systems [3].

Out of the different control strategies, it is possible to distinguish between classical and advanced methods. Classical methods of robot calibration are limited in real-time capability and due to model complexity and numerical instability of parameter identification, whereas, advanced approaches of direct position or path control rely on a real-time synchronized and bidirectional data interface with an external measurement system, and can additionally feature machine learning applications.
The focus of this thesis work is on the design and evaluation of a robust controller for the dynamic behavior of a heavy-duty IR, exploiting an external camera tracking system and additionally featuring a Kalman filter, and to validate its performances. To compare it with the standard robot configuration, many tests with different setups are used, though determining the same performance parameters: pose accuracy (AP) and path accuracy (AT), according to the normative ISO 9283 [12].

The investigation of these factors is essential in understanding the possible error sources and, thus, in optimizing the control strategy. In fact, if on one side, machine tools' kinematic is generally based on Cartesian independent axes, on the other side 6-axis IRs are equipped with swivel joints and designed as serial kinematics, resulting in a more complex transformation between the base and the tool center point TCP.
Furthermore, the accuracy is pose dependent, particularly in proximity of singularities or in the edge region of the robot workspace, where larger deviations happen compared to the center. An additional disturbance source are variable loads, due to the interaction with the workpiece. High process forces at the end effector, indeed, lead to an end effector deflection and, thus, further reducing the AP and AT of the IR [13].

Outline of the work

The work presented in this thesis follows a quite linear path, starting from a study of the state-of-the-art, from literature, of IR's control, with particular regard to the use of external tracking systems such as laser trackers and camera systems.
Subsequently, an analysis and a recap of the work previously done by the team at the *Institut für Werkzeugmaschinen und Fabrikbetrieb* (IWF) of TU Berlin is presented, concerning the implementation of communication and basic control for the IR FANUC M-900iB/700.

The innovative part of the work starts with a new parameter identification, which through step response analysis had been useful to extrapolate the dynamic characteristics of the robot, normal frequency and damping ratio, for different configurations in the working space. Thus, the identification of the transfer functions for each Cartesian coordinate, together with the assumption of negligible coupling effect between them all, lead to the implementation of a first linear control logic in the static case.

Furthermore, analyzing these initial results, some improvements have been made by implementing a fuzzy logic control which could enhance the stability and reactivity of the controller, depending on the dimension of the error with respect to the target position.
At the same time, to minimize noise from the tracking system and, thus, detrimental inaccuracies in the error measurements, a Kalman filter has been adopted. Different typologies have been developed to perform a comparison and try to exploit the best properties of each: a basic one, one considering the disturbances, and the last one extended in order to be position dependent and account for non-linearities of the transfer function.

In conclusion, the finalized dynamic control logic has been tested in different paths and at different speeds, taking example from ISO standards. The experiments resulted in an improving of performance, both in average error and in responsiveness, with respect to the previous work, and comparable to the results reported in literature with other sizes of IR, usually smaller than the one adopted in this work.
The most critical trajectory and configuration have been highlighted, and summary considerations on possible improvement areas, in the development of the control code and in filtering solutions, conclude the work.

1 State of the Art

The following is a brief introduction to IR control and programming methods, from the more classical calibration and offline methods to the most recent advancement in tracking with external measuring system and direct pose control. Furthermore, the most important sources of error affecting IR are discussed, together with an explanation of control and regulation of the aforementioned.

1.1. Basics of IRs' control

A robot system consists of kinematics, control and programming system, effector, and sensors. The kinematic model of an IR describes the geometric relationship between the various coordinate systems. External coordinates, such as the base fixed reference coordinates and the end effector ones, are expressed in the form of Cartesian coordinates, three positional and three orientational, and can be converted into each other by coordinate transformation. Internal joint coordinates, instead, are represented by the vector of the joint positions. Thus, a relationship between the two is required, since movements are usually planned in end effector coordinates whereas control of the axes is carried out in joint ones [2].

IRs' control system can itself be categorized in software, hardware, and calibration. The software system is responsible for the implementation of a theoretically programmed path and for the transmission of manipulated variables to the drives. Programming is the activity of defining sequences of instructions, such as movements and actions, in the form of a user program, that allows the robot to fulfill a specific task [2]. The programming can happen offline, through target motions which are a combination of straight and circle segments, or online (also called teach-in method), moving the robot to the desired targets and then storing the respective points, which has the advantage of being subjected just to the robot repeatability [14]. However, it is difficult, in this way, to program complex path geometries, particularly taking into account corner points or overshoot behavior during sudden changes of direction, and moreover it entails production downtimes during the programming of the robot [2].

Therefore, offline programming is more commonly used, for example for precise milling operations, being also comparable to the programming of conventional computerized numerical control CNC machines [3].

Apart from the programming, the control concept plays a crucial role in converting the manipulated variables into movements, putting in communication the operator with the robot and the process through appropriate interfaces. Additionally, as shown in Fig. 1.1, the control system comprehends sequence, motion, and action control, with the respective functions of interpreting the textual instruction set, interpolating the position specifications, and controlling the interaction with the process. To perform these functions in real time, on the hardware side it is necessary an industrial PC, sometimes in multicore systems, and fieldbus interfaces for managing input and output signals [2].

In servo control, the state of the art are decentralized control concepts based on axis-individual P-PI cascade controllers [14], which dividing the overall system into smaller sub-controller systems (position, speed, and current) can increase robustness and accuracy. Other strategies are, furthermore, feedforward control, which applies previously determined corrections, force control, based on forces measured by external sensors, and impedance control, which relies on the mechanical relationship between target force and motion [2].

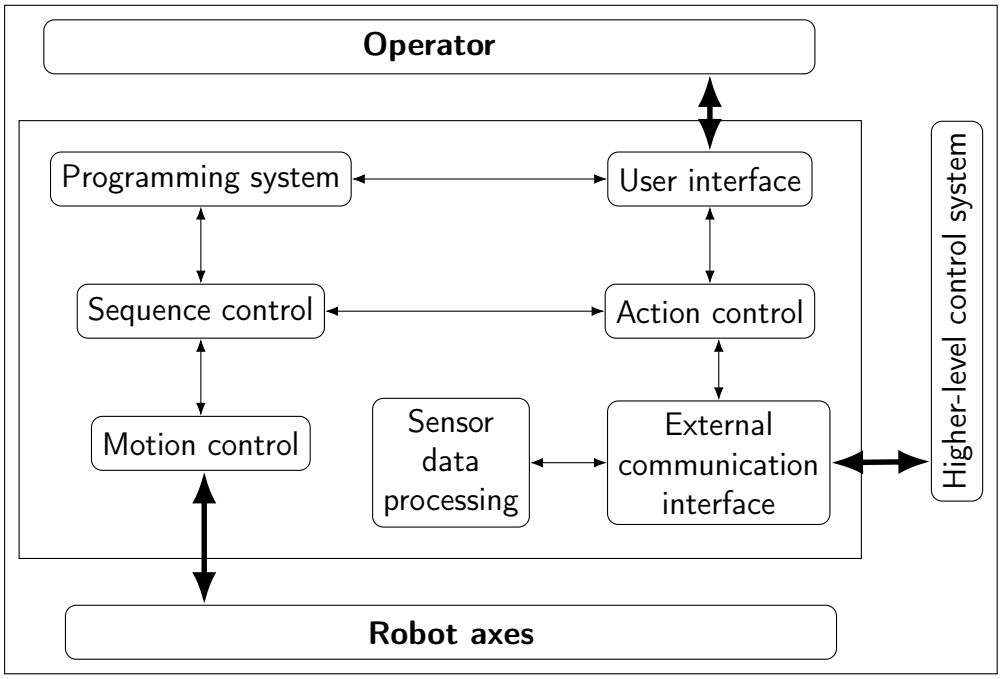


Figure 1.1: Structure of an IR controller [2]

Moreover, the software must also manage the tracking errors of the hardware, caused by dead times in the motors and thus the lagging behind the target path, which depends on the fieldbus systems, responsible for the signal propagation times, and that lead to reduced stability and dynamics of the IR. In addition, the measuring system, in the form of external or internal sensors, is a potential source of error [3].

Lastly, the calibration of the IR is a process which makes use of an external measurement system, such as a laser tracker, to generate independent pose data and define a model of the robot. The most common model today in use is the Denavit-Hartenberg (DH) model, based on four parameters per axis, two translational and two rotational. In the case of 6-axis jointed-arm robots, the first rotational and the translational ones are constant, and determined by calibration routines, whereas the latter is position-dependent and constantly measured, thus known. The TCP is, indeed, subjected to calibration to determine the homogeneous transformation to the robot flange and hence, carry out the path programming in the TCP coordinate system [3]. However, the model deviates from the reality, due to error which occur during the calculation of the parameters, and has therefore a critical influence on the robot's accuracy.

Moreover, the lack of openness of the robot controller from the manufacturer, that without current industrial interfaces does not allow accessing its lowest level, makes complex models useless for calibration purpose. This, however, is due to the fact that it is in the interest of robot manufacturers to protect the software architecture and compensation strategies, which account for the largest share of their know-how [15].
Therefore, for application requiring high accuracy, the only solution is to connect additional modules or industrial PCs which exploit a self-created robot model, calibrated through the use of external measurement technology such as a laser tracker. At least two different robot positions are necessary to determine the tool transformation, but the use of several measurements allows minimizing noise and calculation errors through over-determination [3].

To conclude, particularly when approaching numerically programmed poses, errors in the kinematic model cause the occurrence of deviations and the reduction of positioning accuracy, therefore a precise calibration is necessary for the robot controller to perform a correct compensation [2]. This type of compensation strategy, contrary to control optimization, relies on additional measures to determine forces, temperatures or displacements, and counteract them. Indeed, comparing the nominal and actual geometry with external measurement systems, it is therefore possible to determine deviations and to correct the pose.

1.2. Mechanical and thermal influences on accuracy

The international standard ISO 9283 [12] specifies performance parameters of IRs and their test methods, distinguishing them between static and dynamic ones. Furthermore, it intends accuracy in machine tools as the geometric accuracy, i.e., without loads attached or generated by the process.
The two most important values are, therefore, pose and repeat accuracy. The AP, or absolute accuracy, describes the difference between a nominal target pose and the mean value of actual poses. It can be further subdivided into positional accuracy, representing the difference of the translation terms, and orientation accuracy. On the other side, also Path Accuracy (AT) is composed of position and orientation accuracies, referring respectively to the maximum distance between the center of gravity line of 10 paths of equal length and to the difference between the nominal and actual orientations along the path. Pose Repeatability (RP) instead, describes the average distribution of n actual poses around the target pose. When the IR is used in taught applications, such as pick-and-place and welding processes, the repeatability is the principal factor, and absolute parameters are of low interest, while they become relevant in robot programs created offline via CAD or CAM [3].

Besides the control, also mechanical aspects, such as arm elements, joints, and drives, have an influence on IR accuracy, both in the static and dynamic case. Static mechanical influences are due to the compliance of individual structural elements, in particular of robot joints, which in IRs is much higher than in machine tools. Additionally, compliance of IRs is position dependent, so that static displacement compensation is particularly problematic for machining operations with large working area [3].
In literature, there is an agreement in pointing gearboxes as the primary cause of compliance, which furthermore is not linear but has the shape of a hysteresis curve [15].
On the other hand, dynamic influences are due to motors and drives, but many of the induced disturbances can be compensated by suitable control methods. A robust control prevents the motors' dynamic limit to be exploited, ensuring the stability of the IR [3].

For achieving the highest precision, the thermal behavior of the kinematics must be considered as well, since, especially during long operating sequences, the loads might cause the thermal expansion of the robot's arms, bearing, and gears. These thermally induced geometric changes lead to a deviation of the model, and hence of the TCP from its nominal position [3]. The causes for the machine structure heating can be divided into internal and external. The firsts comprehend the heating of motor shafts, gearboxes, and axle bearings, mainly due to friction.

The heating of the arms, instead, is significantly lower thanks to the larger mass and surface area. However, for cost reasons, IRs are usually not featured with liquid cooling for motors. External influences, instead, are the environment conditions affecting the robot, and contrary to the internal ones, affect the entire kinematic [16].
Anyhow, though thermal effects largely affect the absolute accuracy, they have no effect on the dynamic disturbances, and thus on the path accuracy. Therefore, thermal influences does not play the same role on accuracy as control technology and mechanics [3].

1.3. IR control strategies

IRs are put in motion by electric drive systems, consisting of an electric motor, a gearbox, and an encoder to determine the motor shaft angle. However, as mentioned above, due to the influence of friction, torsional torque, and thermal expansion, the corresponding angle on the reduction side never exactly corresponds to the true axis position [3]. These errors add up over the entire kinematic chain, with more influence the farther they are from the TCP. Therefore, since knowing the actual axis angle is essential for the robot positioning, a solution is installing encoders on the output side of the gearbox, a technique called direct measurement systems (DMS) and already in use by FANUC America and other companies, but still relatively new [17].
Two main areas of potential improvement can be identified: the increase in precision and robustness of the output-side measuring system, and the integration of the measurement system in the controller. However, since the main causes of absolute accuracy errors arise from the structure itself and consequently cannot be compensated in this way, no relevant improvements are expected for this technique and therefore, it is more promising to focus on the use of external measuring systems [3].

Indeed, the adoption of external measurements systems offers an effective way of controlling robotic arms, by providing feedback data of the actual IR position with respect to the desired one. Typically, the control logic works by transforming the measured data in the robot reference frame for computing the error and then providing the appropriate input for compensating it.
The IR pose control strategies can be of several types, but in particular two categories can be defined [18]:

- Control in pose space: it works by computing the difference between the desired pose and the measured pose, and sending the appropriate compensation input for minimizing the error. It is a simpler approach, and can be applied in the same way to robotics arms having different kinematic models.

However, it is limited by not considering the specific dynamic behavior of the robot, thus not allowing to reach the full potential. In this way, the system is non-linear MIMO (multiple input, multiple output), with the different control axes coupled together.

- Control in joint space: errors in pose space are converted as errors in joint angles, thus allowing to directly control each joint as a SISO (single input, single output) system.

In either of the two cases, the closed loop system has a similar structure, schematized in Fig. 1.2. A target reference is set, which can be in pose space or in joint space, depending on the preferred control type. The error between the target and the actual state is then obtained and fed into the control logic, which in returns gives the input to the system: a Cartesian offset in case of pose space control, or a joint movement in case of joint control. The resulting state is therefore measured by an external system, which measures have to be transformed into the target reference frame. This process involves the use of transformation matrices, relating the measuring frame to the robot frame, and of the inverse kinematic model if the joint space control is used, thus allowing to derive the joint angles from the robot pose.

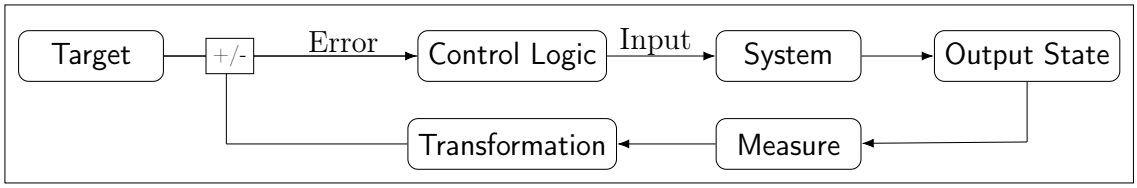


Figure 1.2: Closed Loop Control System

1.3.1. External measurement system

Out of the possible external measurement solutions, there are two that are being investigated nowadays: laser tracker and camera systems. The first ones are already affirmed in many industry applications, due to their reliability and the existing know-how, while the latter ones are relatively new, but are quickly improving and could greatly reduce costs, therefore becoming more common in the future.

Laser tracker

Laser tracking system are highly precise and advanced measuring methods that allow to determine the position and orientation of an object with low error and high sampling frequency. They are classified as part of the dynamic laser interferometers, having the capability of continuously tracking the object in space [3].

A laser tracker system is usually composed by two main elements: a laser source and receiver element, which emits laser light and receives its backwards reflection, and a laser target, constituted by reflective elements and additional measurements instruments installed on the component to be tracked. Based on the analysis of the reflected light, sensors data and time, the laser tracker is then able to compute the distance and the orientation of the target with respect to the fixed frame of the laser source [6] [18].

For instance, the laser measurements process is explained by Yang et al. [6] and shown in Fig. 1.3, in the case of a Leica TS15. In the laser target, three elements can be found. Firstly, a corner-cube prism, used to reflect part of the incoming laser light back to the source: this signal can be used for computing the target distance, deriving it from the phase shift and intensity of the beam, on the basis of the interferometric measurement principle [3]. Furthermore, for absolute distance measurement, a reference point measured by the manufacturer is needed, and this can be combined for creating an absolute interferometer AIFM, as for example in the modern Leica laser trackers [19].
Additionally, the target also contains a 2D inclinometer and a charge-coupled device (CCD) imaging system. The CCD imaging system is used for detecting the incoming laser light that has not been reflected by the prism, allowing then, by image processing, to compute the azimuth and pitching angles of the incident laser relative to the laser target. These values, combined with the data of the inclinometer, make possible to fully define the target pose [6].

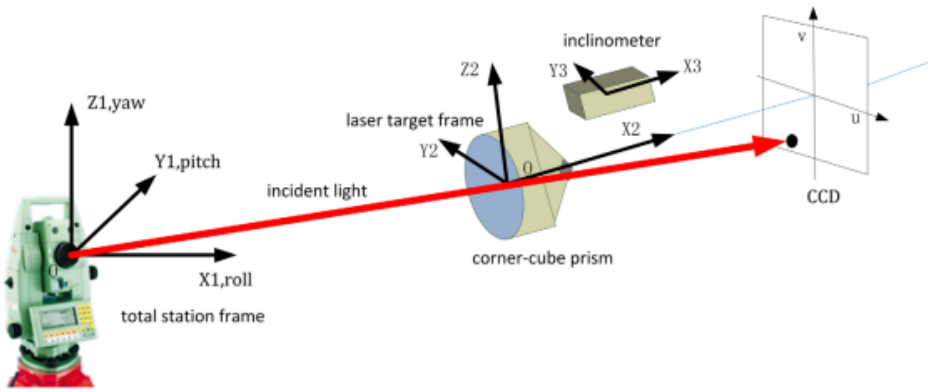


Figure 1.3: Laser tracker Leica TS15 pose measuring method, courtesy of Yang et al. [6]

In the case of IRs, the laser target is mounted in proximity of the tool end effector, thus making it possible to measure the real end effector pose and compare it to the target one. Moreover, to prevent further uncertainties and errors due to the diminishing of the overall accuracy over distance, the tracker must be installed in proximity to the robot [18].

Camera tracking system

A cost-effective alternative to laser tracking systems are camera systems. In order to be used for tracking purposes, photogrammetry and the stereo camera vision principle are used. For the stereoscopic vision to work, at least two cameras are needed, providing two different perspectives, and more than two can be used to increase the accuracy [3]. Similarly to the laser tracking system, a target must be installed on the element that need to be measured, but, in this case, it consists of just a pattern of passive markers that reflect the light also in unfavorable backlight situations, as shown in Fig. 1.4.
In the past, however, many factors influenced the adoption of this technology, such as the resolution and the amount of light required by cameras, the inaccuracy in calibration between robot and camera coordinate system, and computing time. The recent advancement in camera technology and computing power, though, have made it possible [20].

Firstly, a detailed calibration of internal camera parameters, which is usually made by the manufacturer, and of the relative position between the cameras and between the camera system and the robot is needed, introducing known scales or fixed points in the image. Furthermore, the measured marks placed on a mark holder attached to the robot end effector, *targets*, must be coded and calibrated, to ensure unambiguous assignment. Thus, the holder information is stored in the software, as a model to be compared with the measurements and thus, calculate the robot TCP and constant transformation T [3].

Figure 1.4: Camera vision setup system, courtesy of Gharaaty et al. [21]

An aspect to be considered, especially in camera vision, is the presence of noise, which can affect the quality of the measurements, as mentioned by Gharaaty et al. [21], where a root-mean-square method has been proposed in order to reduce the noise issue, while Möller [3], instead, used a PT_1 (first-order lag) filter element.

1.3.2. Coordinate transformation

Considering the DH model, the mechanical structure of the robot is described by the four parameters link offset d, joint angle θ, link twist a, and link length α, with the homogeneous transformation matrix T_i^{i-1} between two consecutive link coordinate frames (frame i with respect to frame $i-1$) defined as follows:

$$T_{i-1}^{i} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & a_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.1}$$

The pose of the end flange frame with respect to the robot base frame $\{0\}$, thus, can be expressed by the homogeneous transformation matrix $T_0^6 = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdot T_3^4 \cdot T_4^5 \cdot T_5^6$, and additionally the known matrix T_6^{EE} and T_W^0 relate it respectively with the end effector and with the world reference frame [6].

At the same time, the end flange frame can be identified through the external tracking system by means of the relation:

$$T_W^6 = T_W^S \cdot T_S^6 \tag{1.2}$$

where the transformation matrix between the end flange frame $\{6\}$ and the target sensor frame $\{S\}$ can be computed iteratively through calibration measurements, on the basis of the work of Wu and Ren [22] and Uhlmann et al. [13].
For establishing the pose control, then, it is necessary to identify the error pose between the measured robot pose from the external tracking system, referred to as *actual* pose $\{6_A\}$ and the current robot pose according to the robot controller, which defines the *goal* pose $\{6_G\}$. As shown in Fig. 1.5, they can be expressed, with respect to the robot base frame and with the addition of the camera frame $\{C\}$, with the following formulas:

$$T_W^{6_A} = T_W^C \cdot T_C^S \cdot T_S^{6_A} \tag{1.3}$$

and

$$T_W^{6_G} = T_W^0 \cdot T_0^{6_G} \tag{1.4}$$

Therefore, from Eq. (1.3) and Eq. (1.4), the error pose becomes:

$$T_{6_G}^{6_A} = (T_W^{6_G})^{-1} \cdot T_W^{6_A} = (T_0^{6_G})^{-1} \cdot (T_W^0)^{-1} \cdot T_W^C \cdot T_C^S \cdot T_S^{6_A} \tag{1.5}$$

The same relations are involved also in the case of laser tracker systems, with the substitution of transformations T_W^C and T_C^S with respectively T_W^{LT} and T_{LT}^S.

As a result, an iterative pose controller, as the one implemented by Möller et al. [11], calculates the corrective action $T_W^{6_R,i+1}$, with respect to the robot base frame, based on the end flange error pose, and the current robot pose:

$$T_W^{6_R,i+1} = T_W^{6_R,i} \cdot (T_{6_G}^{6_A})^{-1}. \tag{1.6}$$

In this way, the target pose can be adjusted by the error transformation, so that, at each iteration, the robot comes closer to the global target pose in the robot base frame.

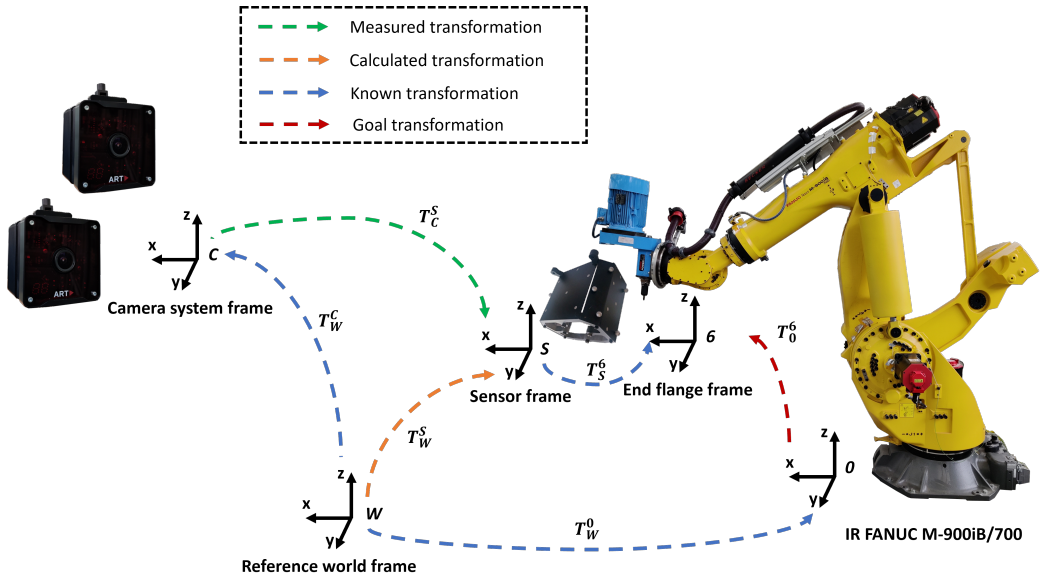


Figure 1.5: Schematic of reference frames and matrix transformations

Moreover, according to Gharaaty et al. [21], it is possible to specify the pose error in the x,y,z coordinates and w,p,r Euler angles for the orientation, and thus to obtain the position error by subtracting the *actual* position vector from the *goal* one: $P_{err} = P_W^{6_G} - P_W^{6_A}$. However, since FANUC uses the $Z_1 - Y_2 - X_3$ Euler angle convention [23], or more precisely Trait-Bryan one (*w for yaw, p for pitch, and r for roll*), and with

$$T_0^3 = \begin{bmatrix} c_1c_2 & c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 \\ -s_2 & c_2s_3 & c_2c_3 \end{bmatrix}, \tag{1.7}$$

one should first rotate $P_W^{6_A}$ about the x-axis, then about the y-axis, and last about the z-axis.

1.3.3. Examples in literature

Overall, the control logic between the laser tracking system and the camera system is similar, with the only difference being the external measurements instrument. Therefore, different results are expected in literature, due to the differences in cost and performance characteristics, such as accuracy, measure frequency, and measurement volume [3].

Laser tracker

An application of the laser tracking for tool pose feedback compensation is provided by Möller et al. [18], where a 6 DoF control logic implementation is used to control the IR *MABI MAX-150* [24] in joint space, significantly improving the precision both in static and dynamic tests. The *Leica Absolute Tracker AT960* [19] was used, together with the measuring probe *Leica T-Mac* [25], and the *Laser Tracker AT901*, to survey the TCP at 100 Hz. Each robot axis was provided with cascade-structure closed loop control, using a mathematical model of the robot dynamic to adapt the parameters with respect to the current pose and feedforward control.
As a result, the static error has been kept below the tenths of millimeters, so in the range of robot repeatability, and below 0.2 millimeters for the dynamic path. However, the dynamic tests show more limitation for what concerns high frequency error components, which is a characteristic dynamic of the robotic arm and difficult to compensate.

Similarly, Yang et al. [6], developed a control logic for a IR *KUKA KR 16-2* with joint space control, exploiting the above-mentioned *Laser Tracker Leica TS15*. Their work confirms the improvements in performances given by the external tracking system for the static case, obtaining an average positioning error of 0.048 mm and orientation accuracy better than 0.1 deg, which are around the robot offline calibration precision. Although the efficiency and reliability of the presented method, further improvement are to be made to enhance the real-time compensation and trajectory accuracy.

In the work of Shi et al. [26], instead, a pose space control compensation, through the *FARO Laser Tracker Xi*, is shown on a *KUKA KR5* arc robot. This method, still, shows a marked improvement in precision on straight-line and circular arc path, even under weight load, through a PID control acting every 12 ms.

Camera tracking system

Some examples of use of camera tracking systems in IR control are provided by Gharaaty et al. [21] and Möller et al. [11], where stereoscopic vision is exploited in order to obtain the position measurements of the tool end effector, resulting in a pose error measured by the cameras below the tenths of millimeters, as in the laser tracking case, since the control logics use similar compensation methods.
Gharaaty et al. [21] used the *C-Track 780* from Creaform on two IRs, the FANUC LR Mate 200iC and the FANUC M20iA, achieving an accuracy of $\pm$ 0.050 mm for position and $\pm$ 0.050° for orientation by performing the corrections in Cartesian space.

Möller et al. [11], instead, used the *AICON MoveInspect* stereo camera system, having measuring frequency up to 10 Hz, on the *MABI MAX-150*. They found out that the camera control loop increases the positioning accuracy, especially for insufficiently calibrated systems, but also lead to deteriorating results when the static positioning accuracy of the robot is higher than that of the camera system. In fact, while the pose error measured by the camera system falls to zero, the one measured by an external laser tracking system shows a steady state error, due to the inaccuracies of the camera measurement, target holder calibration and calibration between the laser tracker and the camera. Therefore, the actual final precision of the robot is mainly defined by the precision of the measuring instrument used. Furthermore, the correction values are affected by latency, due to asynchronous clocks of the camera and the robot [3].

Comparison

In conclusion, it is possible to assess that laser trackers offer better performances, whereas camera systems still need to be optimized to increase the precision in the calibration of the coordinate systems and in the underlying transformation relationships [3].
However, the cost is one of the major disadvantages of laser trackers, which can be very expensive, in particular if equipped with options for orientation measurements. Additionally, they can track only one body at a time, and, since they cannot be solidly attached to the workpiece or to the workspace, environmental factors such as vibrations and air currents can significantly lower their measurement precision. Even if less accurate in a laboratory setting, therefore, camera tracking systems can provide a most cost-effective and valid alternative compared to laser trackers [3].

2 | Setup description

In this section, it is made a description of the equipment and machinery used for conducting the experiments and the tests, performed at the *IWF*, in the *Produktionstechnisches Zentrum* of the *TUB*, which have been an essential part of this research work.
The working system is composed of several elements: a robot arm system, which is the main object of this study; an external tracking system, used for providing measurements and data of the robot's end flange position; and a communication and control interface, to read data from the external tracking system and communicate with the robot the commands and feedback inputs.

2.1. Industrial robot FANUC M-900iB/700

The heavy-duty Industrial Robot M-900iB/700 [27] by the company FANUC K.K., Oshino, Japan, is a six axes articulated arm robot, i.e., having six Degrees of Freedom thanks to six axes of rotation. The robot benefit from a generous work envelope and it is capable of handing up to 700 kg at the wrist, also thanks to its weight of 2800 kg and a maximum reach of 2832 mm, though maintaining a repeatability of $\pm$ 0.1 mm at rated load, based on the norm ISO 9283 [12].
Moreover, the robot features a spring-based gravity compensator, to reduce the gravitational loading of axes two and three, and a parallelogram-mechanism to increase the overall stiffness and maximal payload [28]. Therefore, this robot "is perfectly cut out for the kind of heavy loading, unloading and tending applications found in the metal, automotive, and construction industries", [27]. Additionally, on the flange of the robot and present during all tests, is mounted a blue spindle POWERmaster BEX35 by the company Otto Suhner AG, Lupfig, Switzerland, having nominal power of 9.5 kW [28].
The IR is also equipped with secondary encoders at each axis, to compensate for backslash, elastic deformation, and transmission errors of the gearboxes, in particular during torque and pressure demanding operations. However, from a study conducted by Uhlmann et al. [13] on the same robot, it was evinced that the performance with the control of the secondary encoders decreases, probably due to de-calibration. Therefore, the secondary encoders have been deactivated and not used during the experiments with the robot.

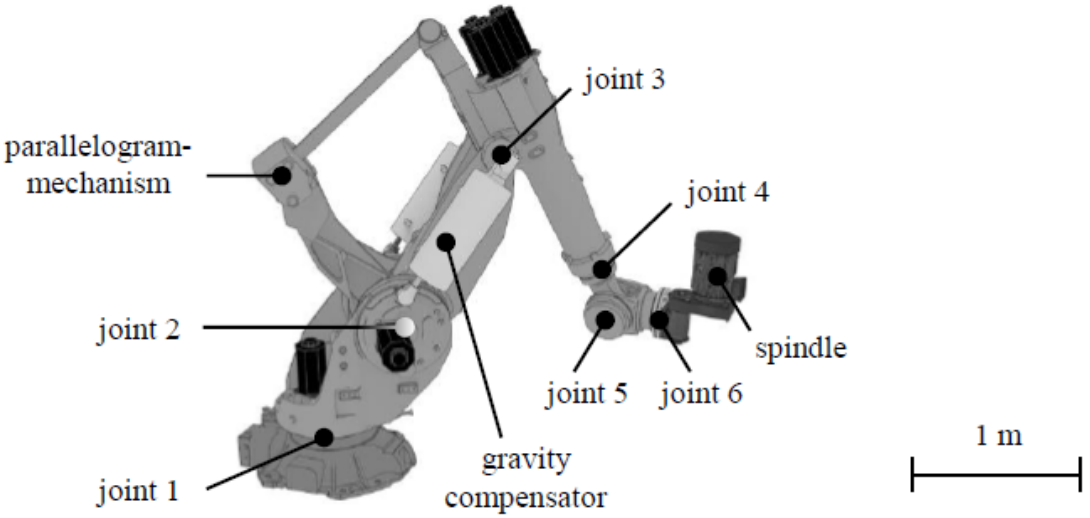


Figure 2.1: Representation of the FANUC M-900iB/700, courtesy of Blumberg et al. [28]

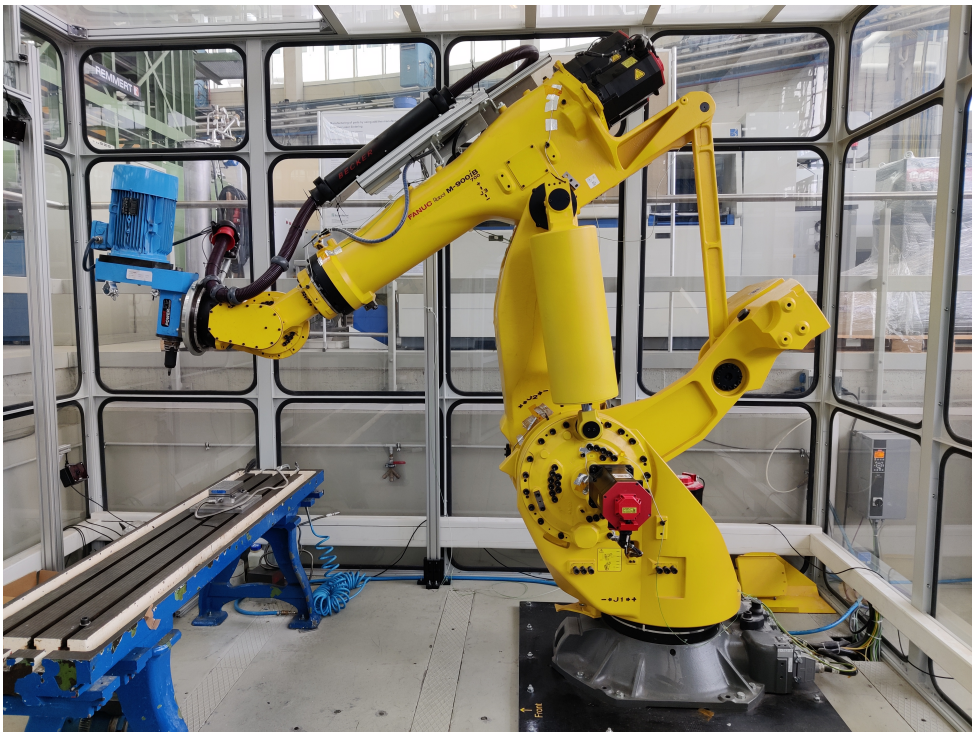


Figure 2.2: Robot FANUC M-900iB/700 at the PTZ

2.2. Controller R-30iB

The robotic arm is paired with the controller R-30iB, also from FANUC K.K., coming in a cabinet, which also provides full connectivity over an Ethernet network to easily connect robots, remote computers, and other hardware [29]. The controller allows moving the robot, to run programs, reading data, and more, and in particular, it features many FANUC's options, such as:

- R632 - KAREL: for executing and compiling KAREL programs
- R648 - User Socket Messaging (USM): for real time communication with the PC
- R739 - Dynamic Path Modification (DPM): for real time modification of the robot pose.

FANUC-KAREL is the proprietary robot programming language for FANUC's industrial robots. It is derived from high-level languages, such as Pascal, and machine control languages [30], and consists of declarations and executable instructions stored in a source code *.kl* file and then convertible into an internal p-code *.pc* file. This file extension is loadable to the controller using the teaching pendant or KAREL Command Language (KCL) [31].
The USM option, instead, enables data exchange, for example process information, between networked robots and a remote PC through the Transmission Control Protocol/Internet Protocol (TCP/IP). Moreover, socket messaging supports client and server tags, thus additional semantic to raw data, such as timeouts, or data formatting commands can also be provided [30].

Last but not least, the DPM option is the one that allows to dynamically control the tool end effector, which is the aim of this work. The control is made through dynamic corrections, called offsets, that can be set by the user via system variables' interface [32]. The corrections can be applied in two modes: along the entire path via *Modal* DPM, or only at the target positions via *Inline* DPM. It is also possible to send accumulated and non-accumulated offset, giving respectively the possibility to sum the offsets' values or to delete the previous ones, and in addition, is there a stationary tracking option to make corrections while the robot is in a certain position [32].
In the developed controller, it is chosen to use the mode *Modal* DPM to perform corrections over the entire trajectory with respect to the user coordinate system (User-Frame), and in addition, for static experiments, the stationary tracking was activated with the *FINE* option, which means that the robot continuously tries to reach the exact target position, while the user can apply corrective offsets and therefore control its movements.

2.3. FaRoC

The communication with the robot controller is done through FaRoC data interface, acronym for Fanuc Robot Control, a Python library for communication with FANUC robots, reading data and executing TP and Karel programs [33]. Developed at the TUB by Wael Hojak during his master thesis work [31], under the supervision of the research engineer Julian Blumberg, this data interface is based on two previous works: Fanucpy [34], a Python package for FANUC industrial robots, and Fanuc DPM Mouse Demo [35], a demo of Fanuc DPM controller.

FaRoC package consists of two parts: a robot interface code, written in Python programming language and running on the PC, and a FANUC robot controller driver, written in KAREL and FANUC teach pendant languages. The data interface makes use of the USM option, and connects a PC (client) with the robot controller (server) via the TCP/IP and UDP/IP protocol to transfer raw data in binary form, achieving a soft real-time with cycle time around 8.5 ms [31].
The python libraries contain several files that serve the purpose of reading and/or writing data. In particular, three classes of TCP client, implemented in Python, are defined:

- *FaRoC_Reader*: allows reading data from the controller
- *FaRoC_Writer*: inherits the previous class and additionally allows changing certain data on the controller
- *FaRoC_Mover*: inherits the previous classes, finally allowing the execution of external programs on the controller.

On the other side, two Karel programs act as TCP servers:

- *FaRoC_SERVER*: can connect with *FaRoC_Reader* and *FaRoC_Writer*, reading and writing data on the controller
- *FaRoC_MOVER*: connects to *FaRoC_Mover* starting a server also able to start external programs.

This separation, chosen for security reason to avoid operating errors, is summarized in Fig. 2.3.
A total 8 data clients can be executed in parallel, each of which can send a maximum of 32 values [31]. In addition, also tracking system and RSI data can be handled. RSI stands for Remote Sensor Interface, an additional Fanuc option to send real-time position data of two types, *measured* and *forward* [23].
Thus, with this option also the pre-planned position up to 23 interpolation times ahead can be sent, at a comparable cycle time with relation to FaRoC's Data Client [31].

To do so, an User Datagram Protocol (UDP) server for each client must be set, which runs independently after initialization either as a thread or as a process, as shown in Fig. 2.4. In this way, data that are pushed by the clients, becomes available in the form of variables and can be read by the main process.

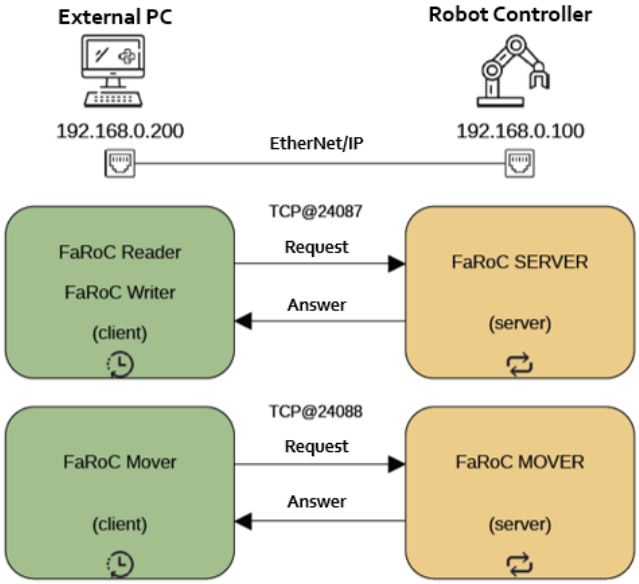


Figure 2.3: Structure of FaRoC data interface, courtesy of Hojak [31]

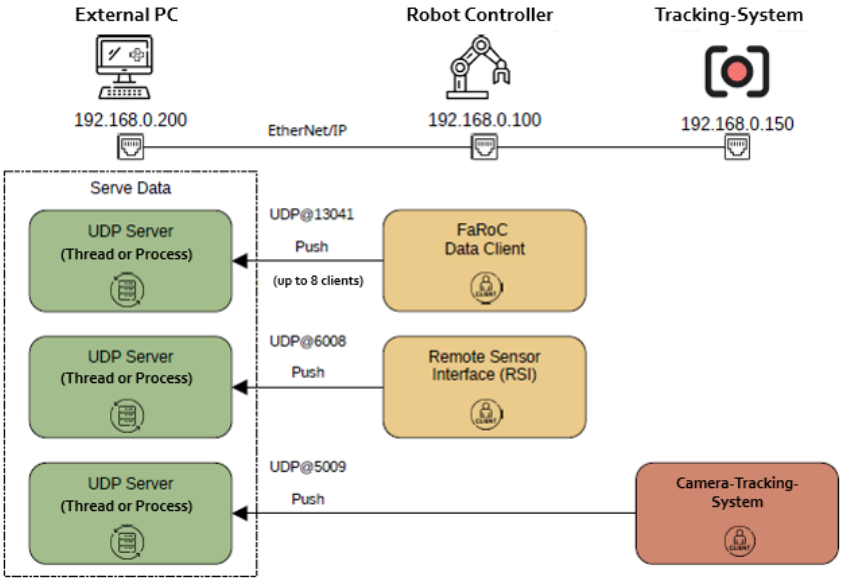


Figure 2.4: Structure of data provision from multiple clients, courtesy of Hojak [31]

Therefore, FaRoC provides *fanucpy*'s functionalities and extend them, enabling sophisticated applications. For instance, it contains two Karel programs, one static and one dynamic data client, providing robot data, such as current joint angles and offset values, for real-time and DPM-based control [31].
From the joint angles, through the IR's forward kinematics, the unique pose can be obtained, and furthermore, by comparing it with the current pose provided by the tracking system and converted into the robot base coordinate system, the pose error is calculated.

In this way, it is possible to dynamically control the end effector position by computing the offset values with the control logic, writing them to the corresponding system variables through the *FaRoC_DPM_MN* program, and lastly applying the corrective offsets through the DPM function, as shown in Fig. 2.5.
The block diagram also presents the computational steps performed by the *FaRoC* logic: the coordinate frames transformation and the derivation of the pose error, and at the same time the integration with the internal robot control.

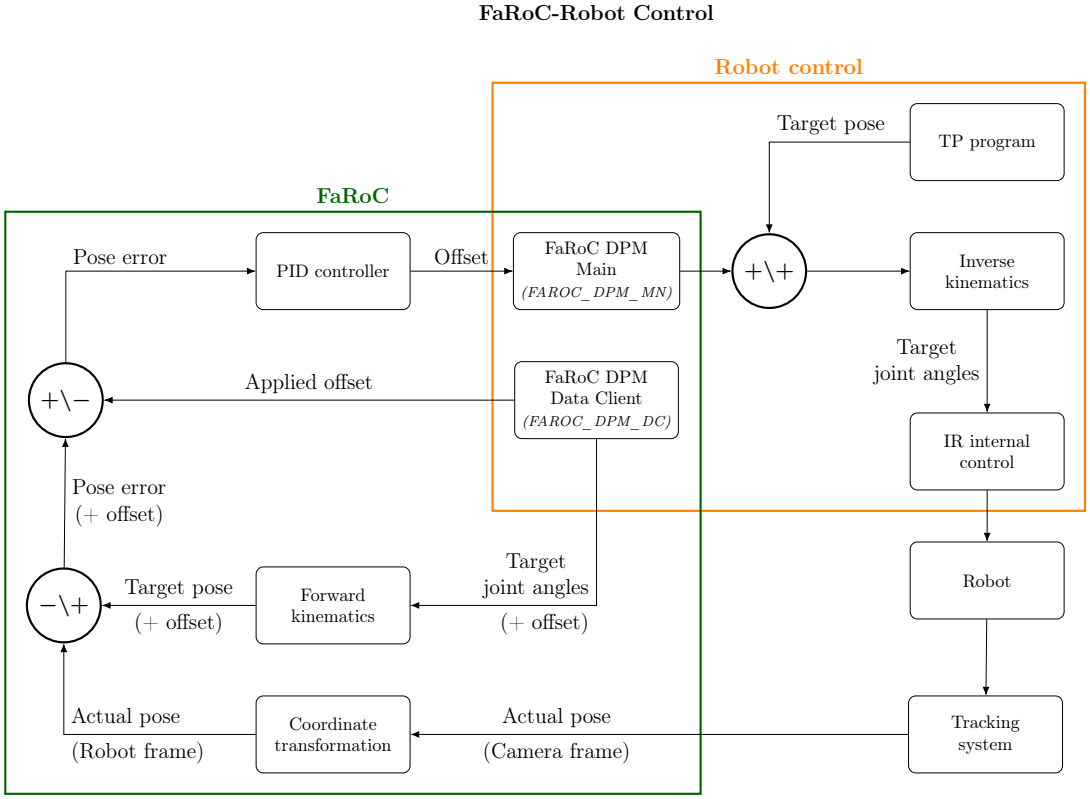


Figure 2.5: Block diagram of FaRoC control

2.4. Roboguide

Roboguide [36] is the simulation software from FANUC K.K. used to write robot programs and simulate the robot behavior. The version adopted in this work is the V8.30.

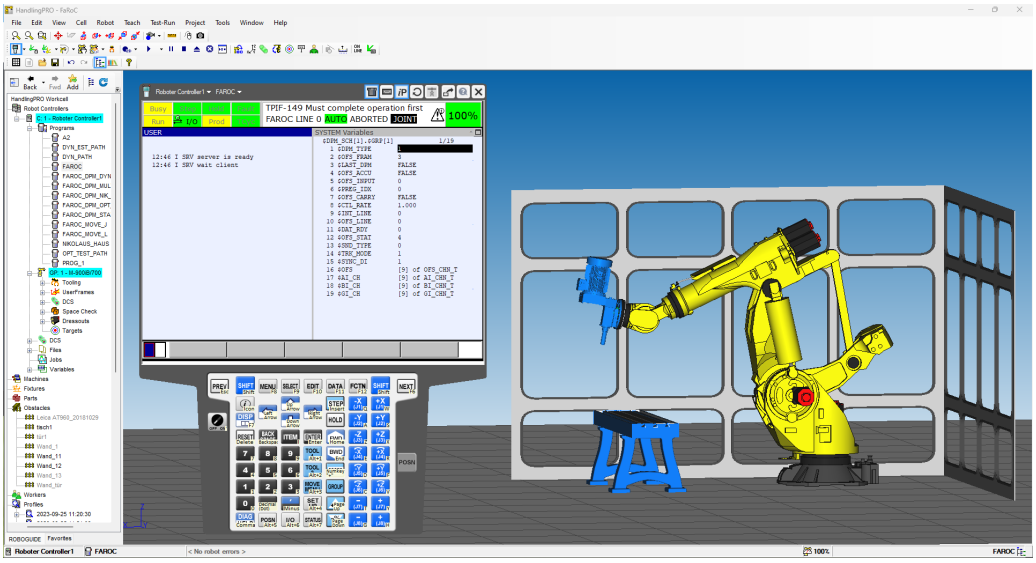


Figure 2.6: Roboguide cell for M-900iB/700 with FaRoC programs

2.5. External tracking system

Furthermore, to implement the position-based control strategy, thus measuring the exact position of the robot's end effector in real time, it is made use of two external tracking systems, a laser tracker and a camera system. The two systems differs in characteristics, but both possess advantages and disadvantages that are being unraveled in the following paragraphs. Therefore, they have been both used for carrying out this work experiments, mostly depending on their availability and on the objectives of the research.
At a later stage, exclusively one of the two, the camera system, has been used to drive the control and to perform the tests.

2.5.1. Laser tracker

The Leica Absolute Tracker AT960, powered by the company Hexagon AB, Stockholm, Sweden, is a high-performance 6DoF laser tracker that can be used for precise measurements at high frequencies, up to 1 kHz combined with the Real-Time Feature Pack (RTFP) [19]. Working together with the real time interface EtherCAT and the Leica Tracker Machine (T-Mac) control sensor, the laser tracker is enhanced with a 7DoF feature (six spatial and one temporal), enabling real time machine control even at high dynamic [25].

The system is composed by a main central unit, the laser tracker head (Fig. 2.7a), which is mounted on a fixed support: it serves as a laser source and provides the measurements data. On the other hand, the position is obtained using the T-Mac (Fig. 2.7b) and combining the laser tracker position measurement with a photogrammetric orientation measurement, obtained through the detection of infrared LEDs mounted on the T-Mac. For this reason, the target device is mounted on the robot tool end, as close as possible to the Tool Center Point (TCP) [3]. However, so far at the IWF it has not been possible to use it for real time applications, since the corresponding communication protocol has not been purchased yet, and an equivalent self-made protocol is up to be developed.
In fact, the main drawback of this machine is the high hardware and software cost, which makes it less commercially interesting.

(a) Leica Absolute Tracker AT960

(b) Leica T-Mac [25]

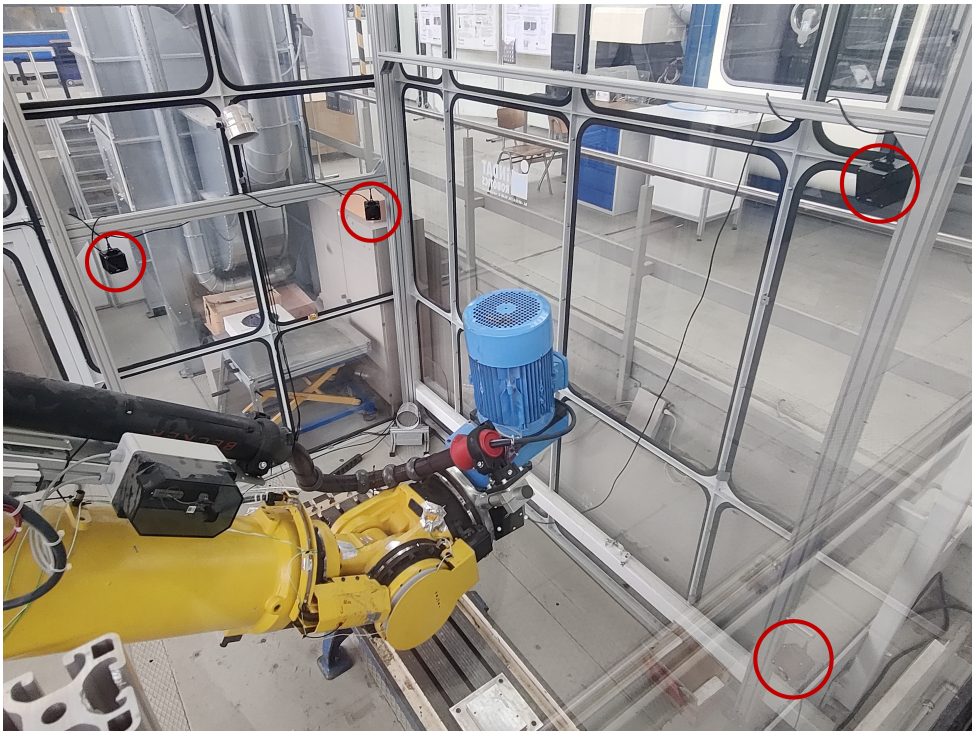
Figure 2.7: Leica laser tracker from Hexagon AB [19]

2.5.2. Camera system ARTTRACK5

The camera system, instead, has been fully integrated with the robot system and can provide real time data for the online robot control. It is, in fact, the optical tracking system ARTTRACK5 [37] from the company Advanced Realtime Tracking (ART) GmbH & CO. KG, Weilheim, Germany, based on infrared technology and able to measure the position and orientation of moving objects.
The used setup consists of four infrared cameras (Fig. 2.8), mounted on the cage frame which surrounds the robotic arm, and of a specific target fixed on the robot working tool, in the immediate vicinity of the TCP. The cameras emit synchronized infrared flashes and can operate up to a frequency of 300 Hz, but their operational frequency is set to 150 Hz for a full frame resolution of 1280x1024 pixels. The produced grayscale image is the result of the detected infrared beams that have been reflected by the target, which is designed with a series of markers that allow, through an image processing algorithm, to reconstruct the target position with respect to a fixed reference frame.

(a) Camera in the robot cage

(b) Camera design from ART GmbH [37]



(c) Camera system configuration

Figure 2.8: Camera ARTTRACK5

The markers are passive markers, therefore retro-reflective, i.e., they reflect the incoming infrared (IR) radiation into the direction of the incoming light, even in unfavorable backlight situations. They can be of different types:

- spherical: visibility from all sides, IR Range: 4-10 m, sensitive surface
- spherical coated: visibility from all sides, IR Range: 3-7 m, robust surface
- flat: visibility limited ± 45°, IR Range: 4-10 m, cheap, most robust surface.

Single markers would be sufficient for positional tracking with 3DoF, but for tracking also the orientation and obtain the 6DoF, special targets are needed.

During the work, therefore, two kinds of target geometries (Fig. 2.9), consisting of rigid arrays of markers, have been used: the first one, with 8 spherical markers, simple geometry pattern, and 3D printed, the second one with 10 spherical markers and 16 flat markers, more complex geometry, and made out of aluminum. Since the last one was better performing, once available, the conclusive testing phase has been carried on with it only, mounted as shown in Fig. 2.10.

In fact, to operate in different working positions, it is better to have several markers at different locations, and minimize the interfering contour maintaining, at the same time, high visibility of as many markers as possible. Furthermore, from the principle of "stereoscopic vision", in order to properly assess the target position at least two cameras, seeing 4 or more markers, are necessary. Hence, more iteration on the realization of the target from ART GmbH were necessaries.

Figure 2.9: ARTTRACK5 targets: first and latest design

In addition, the cameras are subject to "room calibration", to determine their internal parameters and their relative position to each other, and then to "body calibration", to learn to recognize the unique arrangement of the markers on the target [31]. In fact, the target introduces fixed points into the captured image, which allow determining the position and orientation of the robot TCP through geometric imaging laws based on the principle of central projection [3]. By calculating the beams' intersection point, thanks to the path distances of the infrared light, it is possible to obtain a representation of the markers' position (see Fig. 2.11), and subsequently, the 6DoF data for calculating the position and orientation of the target and therefore of the TCP [31]. For this reason, it is important that the distance between any pairs of markers is unique, so that it can be recognized without ambiguity.

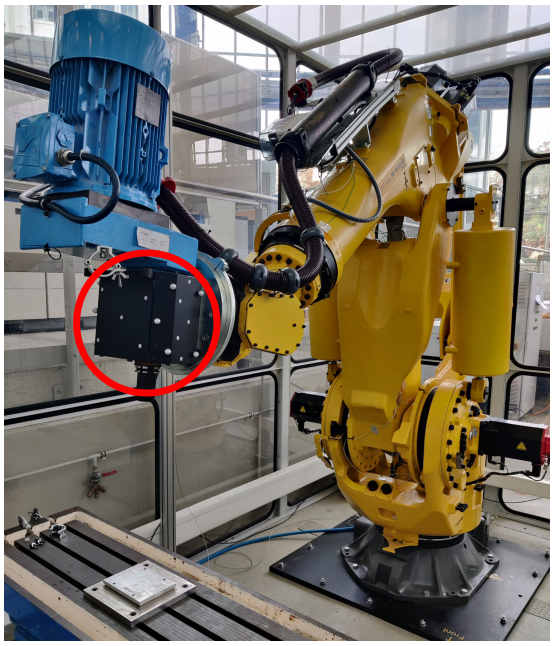


Figure 2.10: ARTTRACK5 target mounted on robot tool

For performing these calculations, the camera system comes with a related software, DTrack3 [38], which support the setup of the camera configuration, storing the target information, and then calculate the real time orientation and positions by comparing the acquired measurements from the cameras with the model. The calculated points can then be used to obtain, through the constant transformation matrix T and the inverse kinematic model, the pose, and joint configuration of the robot TCP.

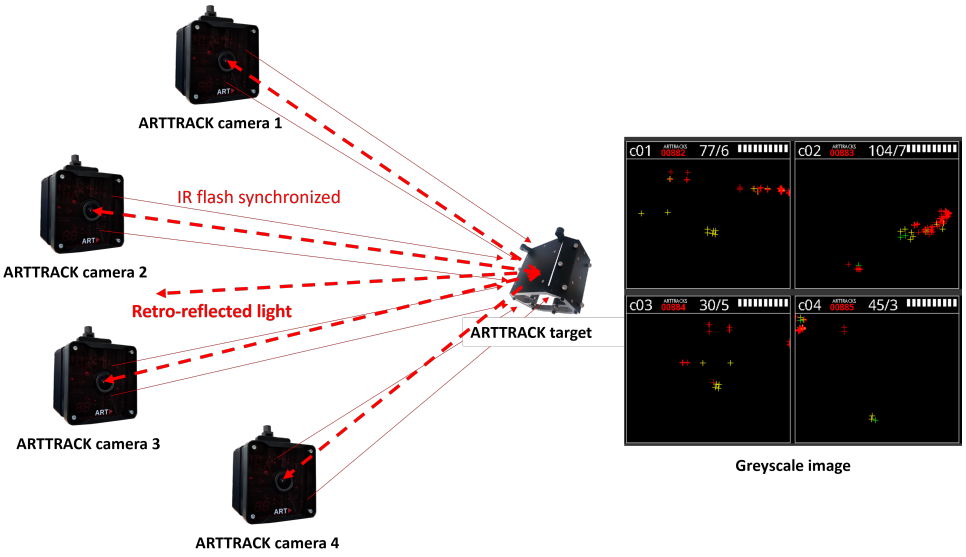


Figure 2.11: Target recognition through ARTTRACK5 cameras

3 | Methodology

3.1. Objective and approach

The scope of this thesis work is the development of a robust control logic aimed at improving the absolute precision and accuracy of an IR based on an external tracking system, such as cameras or laser trackers.

The research work is carried out on a FANUC M-900iB/700 available at the *Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF)* (Institute for Machine Tools and Factory Operations), in the *Produktionstechnisches Zentrum (PTZ)* (Technical Production Center) of the *Technische Universität Berlin (TUB)*. The control logic is based on a previous thesis work [31] focused on the development of the communication system with the FANUC IR, which made possible to read data and send inputs commands to the robot.

The external tracking system on which the work is based is a system of four infrared cameras ARTTRACK5 [37] from the company Advanced Realtime Tracking (ART) GmbH & CO. KG, Weilheim, Germany. Additionally, a laser tracker Leica Absolute Tracker AT960, powered by the company Hexagon AB, Stockholm, Sweden, has been used for precise measurements of the robot dynamic behavior.

The work has been assessed by comparing the IR performances before and after the action provided by the control logic, in terms of precision with respect to the reference pose, both in static and dynamic cases.
As part of the control logic development, the implementation of a Kalman filter is discussed, necessary to filter out the noises of the measurements of the cameras, which could impact the final result.

After having obtained a better understanding of the problem from the state-of-the-art research and having familiarized with the system instrumentation and communication protocol, a first parameter identification of the system is going to be carried out and eventually experimental tests are performed in order to assess the optimal control strategies and control gains of the controller.

3.2. Dynamic modeling and parameter identification

In order to properly control and stabilize the tool end effector, it is required to identify the dynamical system parameters and its transfer function. By measuring the system response to particular types of inputs, it can be possible to determine many relevant characteristics of the system, such as its natural frequencies, the stiffness, the damping factor, and the delays.

The identification of these parameters can happen *offline*, hence firstly storing the measured data and then processing them, or *online*, when it is performed parallelly to the robot execution. Further differentiations can be made with respect to how the model is determined, whether recursively, as each measurement becomes available, which is typical for real-time identification, or with non-recursive methods, for offline processing only, which can be either direct or iterative, i.e., respectively in one pass or step-wise [39].

For processes with good signal-to-noise ratios, as in the specific case of IR tracking with a laser tracker or camera system, it is possible to determine the characteristic values analyzing step or impulses responses, and exploit them in combination with a Kalman filter, which is used for state estimation of dynamic systems.
In the step response analysis, which is one of the most used and interesting tests, a step function is used as the input and the resulting motion is studied, giving as a result, usually, an approximated system of the first order with dead time:

$$\bar{G}(s) = \frac{K}{1 + T_D\, s} e^{-T_S\, s} \tag{3.1}$$

where T_D is the time constant and T_S the delay time [39].

This approach, was also used by Hojak [31] in his work, by applying a manipulated variable to the controller and externally measuring the system's response. Therefore, the TF input is the applied offset of the controller, and the output is the displacement measured by the tracking system. In that case, were utilized jumps of 100 mm for the *x, y,* and *z* axes and of 30 degrees for the *w, p, and r* axes, and with a duration of around 10 s. Thus, identified TFs were based only on the step responses, ignoring parameters such as the inertia of the robot, frictional effects, stiffness and damping of the motion axes, and other nonlinearities of the system [31]. They turned to be very similar, so a unique control system was developed and used to design the controller, with the simplifying assumption to be an integral element with dead time of 64 ms [31].

$$\bar{G}(s) = \frac{37.74}{s} e^{-0.064\, s} \tag{3.2}$$

However, in this work, it has been considered not sufficient, and therefore, a second order Transfer Function (TF) has been exploited, computing the natural frequency and the damping from the oscillation frequency and from the peak's logarithmic decrement. Thus, the TF is defined as follows:

$$\bar{G}(s) = \frac{\omega_0^2}{s^2 + 2\ \zeta\omega_0\ s + \omega_0^2} = \frac{\omega_d^2 + \alpha^2}{(s+\alpha)^2 + \omega_0^2} \tag{3.3}$$

with ω_0 natural frequency, ζ damping ratio, $\omega_d = \omega_0\sqrt{1-\zeta^2}$ damped frequency, and $\alpha = \zeta\ \omega_0$. In particular, in prevision of the design of a robust dynamic control, the slowness of the jumps and the consequent lack of consideration for inertial and dynamical parameters has been deemed inadequate, and therefore it has been carried on an investigation on the system dynamic properties and on the differences between axes and configurations.

In fact, the step input can be either in the joint space or in the Cartesian space, with the difference that while the inputs in joint space are uncoupled, movements in the Cartesian space are coupled, being each motion composed by the sum of the joints rotations. Therefore, both kind of tests have been designed to collect more data as possible, but then just the results in Cartesian space have been useful for determining the model, due to limited access to the robot control software that allowed the correction, through DPM interface, of the Cartesian pose only [32].
To account for the six DPM input channels, i.e., the Cartesian coordinates *x, y, z, w, p,* and *r*, the same test has been performed distinctly for each of them. Though, being the motions coupled, the resulting movements are given by the contribution of each single joint behavior, it is possible to define independently the second order transfer function between the input and the output motion, under the simplifying assumption, supported by experimental evidence, that the influence on the other coordinates is negligible and moderate by the internal robot control. This offers advantages in controlling the motion sequence, since the movements in each coordinate can be controlled independently, and in computing the Kalman Filter parameters.

However, it is necessary to study the robot behavior in different spatial configurations, since the end effector dynamics can change based on the pose. For this reason, some relevant positions have been defined in order to correlate the change in dynamics with the change in pose. The principal influence is due to the first three joints, which mostly affect the translational spatial coordinates and the arm elongation, that therefore posses a more complex dynamics.

Therefore, 14 configurations have been selected for the dynamic estimation of the robot, trying to cover the most of the available workspace. These configurations range from 1400 to 2100 mm in radius and from 400 to 1200 mm in height, considering the robot base, i.e., the first joint frame, as reference. Fig. 3.1 shows the reached positions, while Table 3.1 shows the coordinates in joint and Cartesian space for the 14 configurations.

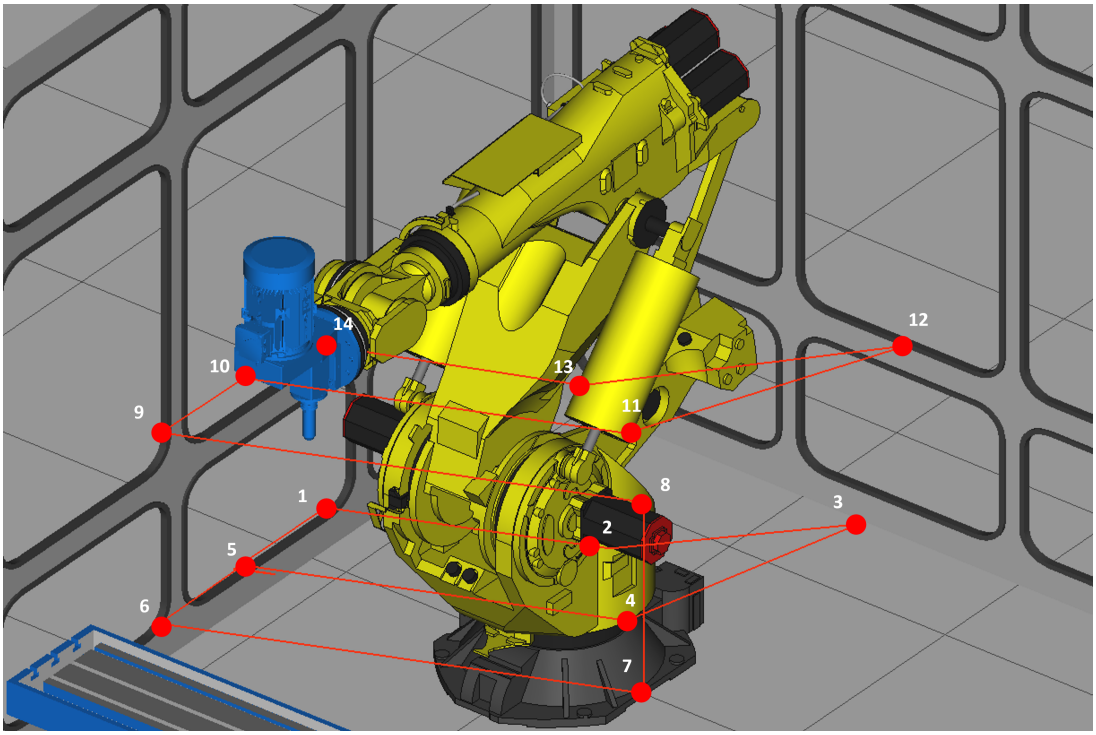


Figure 3.1: Dynamic estimation positions

The idea between the selected positions is to characterize the dynamic of the robot for as many configurations as possible, extrapolating the values thanks to linearization between known points. In particular, four main areas were characterized, with regard to the radius distance and the height from the robot base frame, as represented in Fig. 3.2. The radius is considered, in this case, because the x and y axes are interchangeable in reason of a rotation of the first joint.
Computing the average of the dynamic characteristics of the positions, some representative value for each section can be extrapolated, and through interpolation they can be estimated for the whole workspace volume. Conscious of the fact that a better approximation can be done, enlarging the position dataset, though this approach can serve an overview of the IR's behavior change when the arm is elongated or contracted, lifted or lowered, and therefore the parallelogram mechanism's configuration varies.

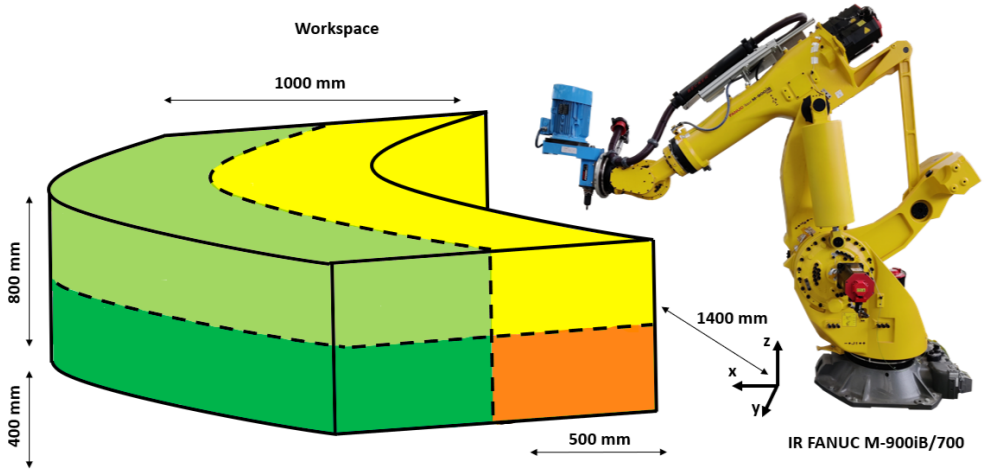


Figure 3.2: Dynamic estimation workspace

Table 3.1: Dynamic estimation positions

Pos.	Joint (deg)	Cartesian (mm / deg)
1	[0.0, -24.0, 39.0, 0.0, -39.0, 0.0]	[1407.4, 2.0, 407.8, -46.3, -82.2, -129.9]
2	[45.0, -24.0, 39.0, 0.0, -39.0, 0.0]	[993.8, 996.6, 407.8, -46.3, -82.2, -84.9]
3	[90.0, 5.0, 44.0, 90.0, -90.0, -46.0]	[297.0, 1607.6, 401.9, -46.3, -82.2, -129.9]
4	[45.0, 5.0, 44.0, 0.0, -44.0, 0.0]	[1343.9, 1346.8, 401.9, -46.3, -82.2, -84.9]
5	[0.0, 5.0, 44.0, 0.0, -44.0, 0.0]	[1902.6, 2.0, 401.9, -46.3, -82.2, -129.9]
6	[0.0, 28.0, 38.0, 0.0, -38.0, 0.0]	[2399.3, 2.0, 393.8, -46.3, -82.2, -129.9]
7	[45.0, 28.0, 38.0, 0.0, -38.0, 0.0]	[1695.2, 1698.0, 393.8, -46.3, -82.2, -84.9]
8	[45.0, 20.0, 4.0, 0.0, -4.0, 0.0]	[1688.1, 1691.0, 1211.2, -46.3, -82.2, -84.9]
9	[0.0, 20.0, 4.0, 0.0, -4.0, 0.0]	[2389.4, 2.0, 1211.2, -46.3, -82.2, -129.9]
10	[0.0, -6.0, 7.0, 0.0, -7.0, 0.0]	[1895.8, 2.0, 1204.4, -46.3, -82.2, -129.9]
11	[45.0, -6.0, 7.0, 0.0, -7.0, 0.0]	[1339.1, 1342.0, 1204.4, -46.3, -82.2, -84.9]
12	[90.0, -6.0, 7.0, 84.0, -53.0, -80.0]	[234.7, 1780.9, 1207.3, -47.0, -81.2, -91.9]
13	[45.0, -33.0, 4.0, 77.0, -20.0, -77.0]	[1044.5, 907.1, 1101.4, -40.0, -82.2, -110.7]
14	[0.0, -33.0, -4.0, -0.0, 4.0, -0.0]	[1396.3, 2.0, 1098.1, -46.3, -82.2, -129.9]

3.2.1. Step response and axes coupling analysis

Let's take, for example, the oscillation on the x-axis caused by a step input of 1 mm on the x-channel for the second test position, and its comparison with the responses originated by the same input but on the y, and z axes, shown in Fig. 3.3. It is possible to notice a similar pattern of oscillations, but of course it is evident the action of the internal robot control in avoiding a displacement at the steady state. A plausible explanation is the combined action of different joints, that with particular emphasis in certain configuration compared to others, originates a vibration on different Cartesian coordinates which is however confined through the action of the control logic. Furthermore, noise has to be taken into account, and it is assumed to be stationary and stochastic, i.e., with constant mean $En(t) = 0$.

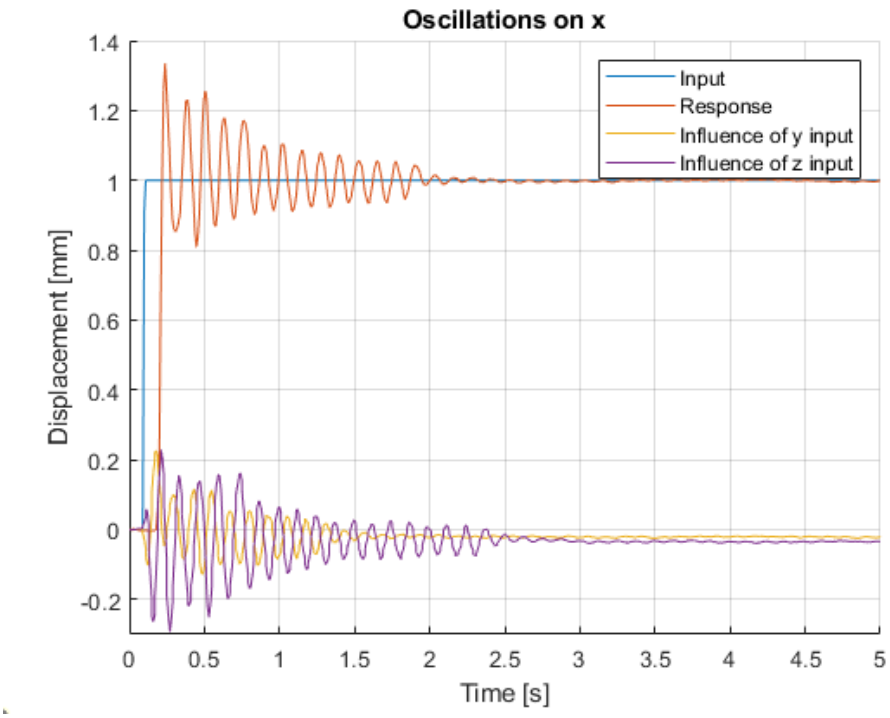
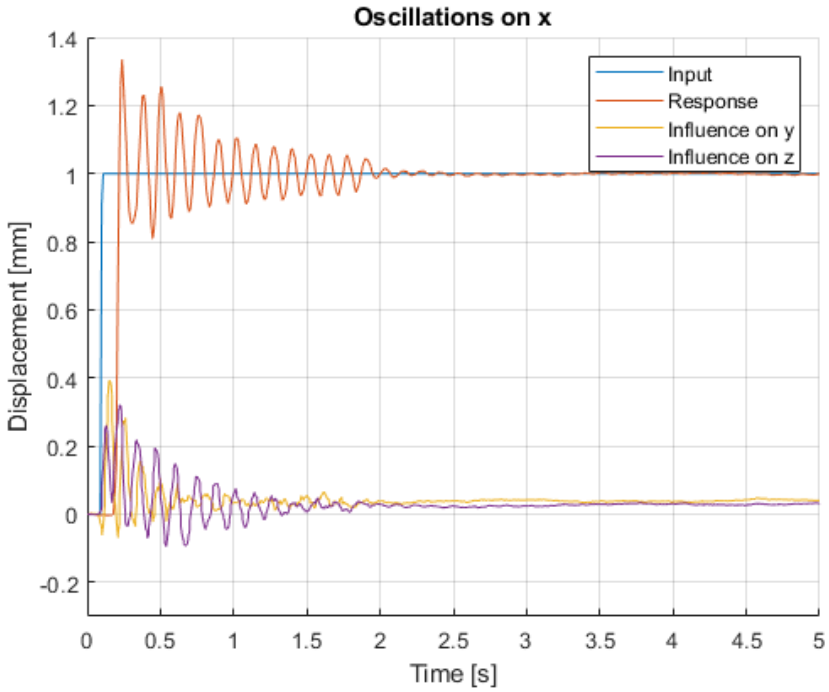


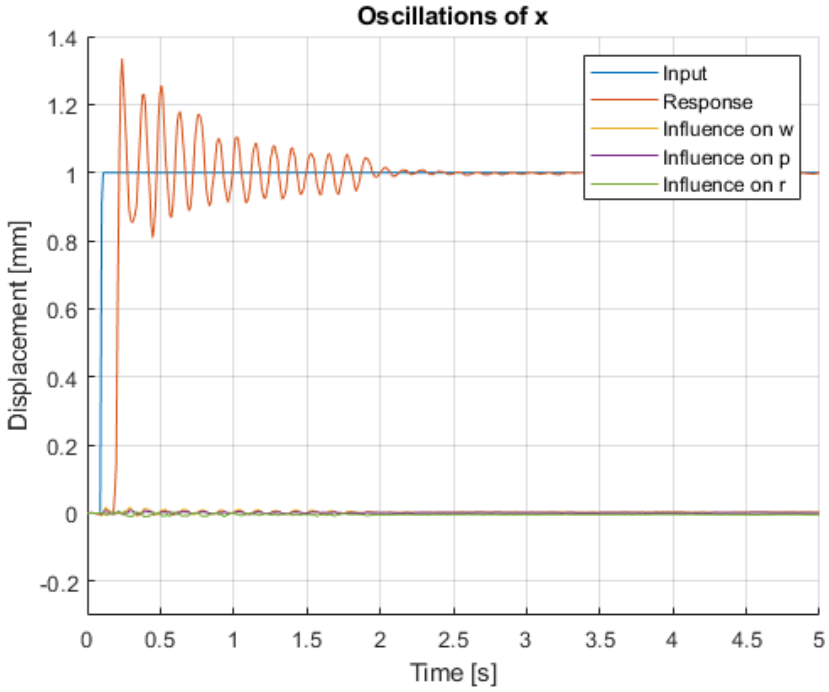
Figure 3.3: Oscillations on x-axis influenced by y and z for Pos. 2

At the same time, Fig. 3.4 shows the influence of the input on the x-channel on various axes. In Fig. 3.4a, it is possible to see how, though the magnitude of the oscillations on the y and z axes are comparable with the one on x-axis, the robot control act to damp them and to avoid undesired displacement. It is also interesting how their responses are quite immediate, whereas there is a delay time for the x-axis, of around 10 sampling times, possibly explainable by some joint re-orientation to perform the required movement or motors' dynamic limitations.

The same, also, is the result of its influence on w, p, and r, the Euler angles, which oscillations however are so small to be negligible (see Fig. 3.4b).



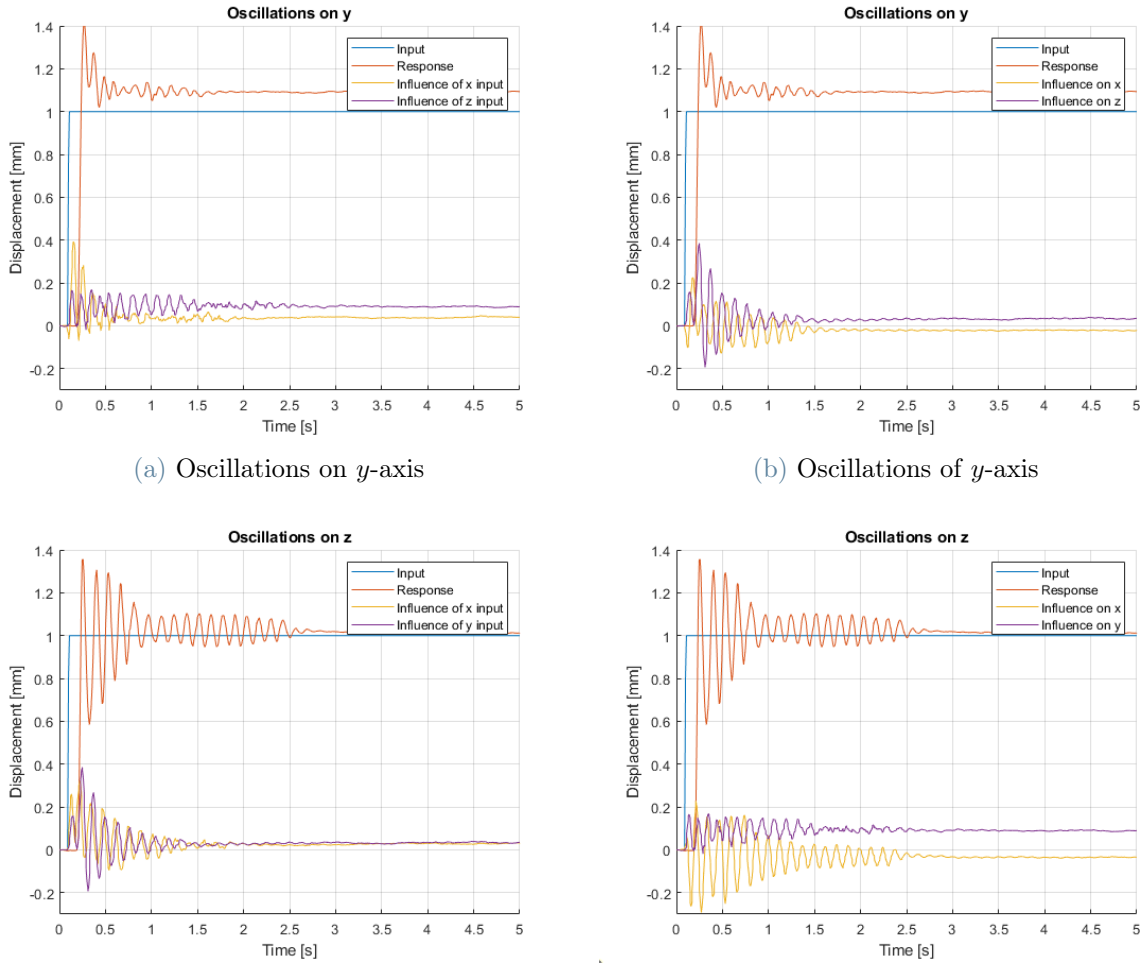
(a) Influence on y and z



(b) Influence on w, p, and r

Figure 3.4: Oscillations on x-axis for Pos. 2

Of course, the same behavior happens on the other coordinates, and in particular on the y and z axes shown in Fig. 3.5.



(a) Oscillations on y-axis

(b) Oscillations of y-axis

(c) Oscillations on z-axis

(d) Oscillations of z-axis

Figure 3.5: Oscillations on y and z for Pos. 2

3.2.2. System delays

The system presents delays, which can be obtained from the measured data of the step response tests. For instance, it is known the exact instant at which the control action is sent. Additionally, the nominal applied offset at the tool end effector and the nominal joints angles are given by the robot through the *FaRoC Data Client*, together with the joints angles measured by the robot's encoders. Therefore, the difference between the instant at which the input is sent and the encoders measurements detect the first movement is the system delay.

Fig. 3.6 shows the system response to a step input for the y-axis (same behavior is seen also for the other coordinates), when an offset of magnitude 1 mm is sent to the y-axis channel at time zero. A first delay is present between the instant in which the input is sent and the one in which the system starts to apply it, by computing the joint angles movements. The second one is present between the nominal joint angle, computed at the previous instant, and the joint movement measured by the encoders of the robot.
As a result, the estimated overall delay is in the range of 45 to 60 ms.

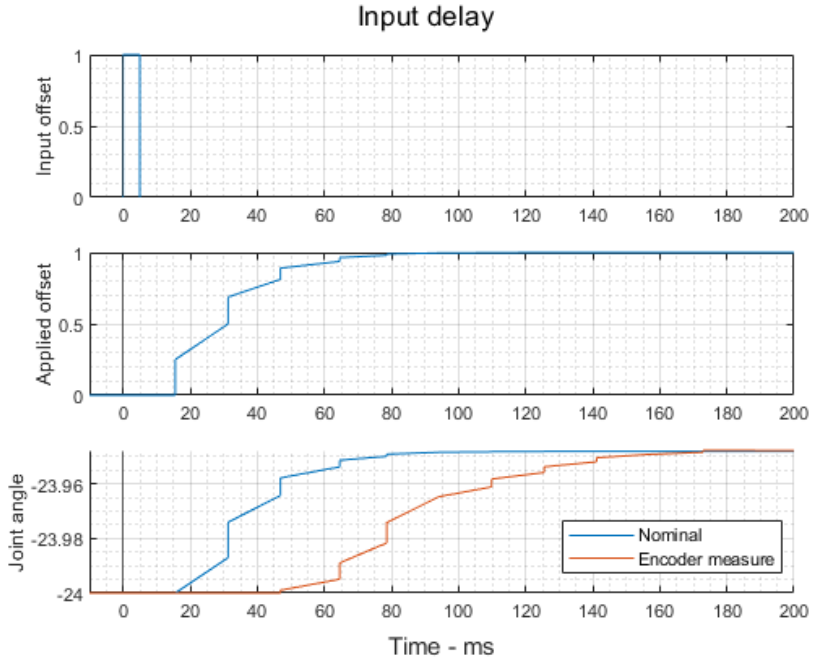


Figure 3.6: Delay comparison

3.2.3. Transfer functions

The study of the transfer functions, estimated from the resulting oscillations in response to the step input, has been carried on in MATLAB, by means of the *tfest* function. The approximation has been made with 6th order TFs, considering unknown time delay and discretizing them over the sampling time of each signal (approximately 0.01 s).
The so obtained TFs, though satisfactory representatives of the system behavior, as shown in Fig. 3.7, are far beyond the use of this research, but useful for retrieving the main natural frequencies and their damping factor, and thus derive a 2nd order TF as previously assessed.

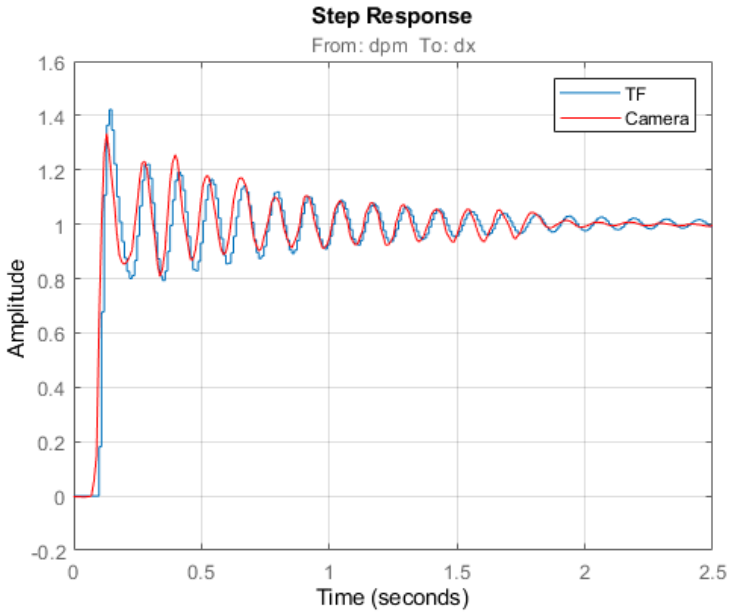


Figure 3.7: Step response and its 6th order TF on x-axis for Pos. 2

This approach was preferred to obtain a construction easily parameterized, so useful for interpolation along the robot workspace and, subsequently, for building an extended Kalman filter. Fig. 3.8 present a comparison between different orders of TF and how they approximate the measured response.

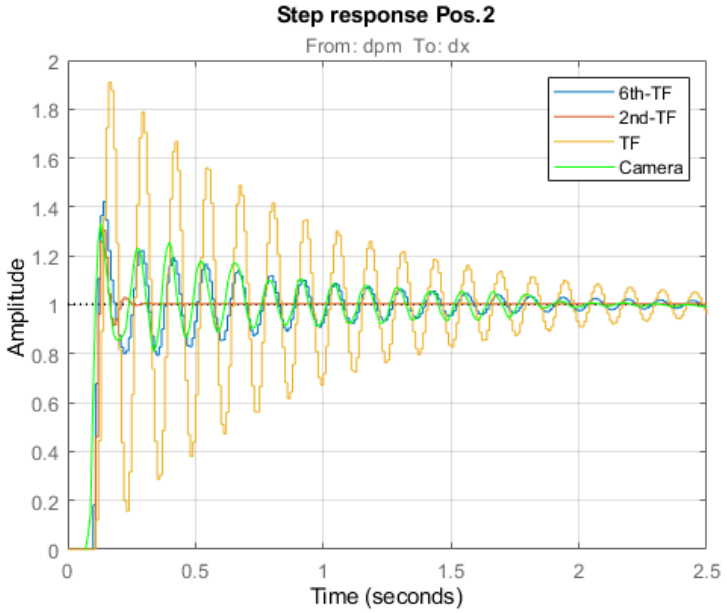


Figure 3.8: Comparison between TFs on x-axis for Pos. 2

As it is evident that the best one is the one of highest order (in blue), it is to note how the TF constructed by the derived main normal frequency and damping factor (in yellow) better represent the dynamic compared to the *tfest* computed 2nd order one (in red), though initially overestimate the response with a higher overshoot with respect to the measured signal (in green).

Eventually, also the applied offset and joint motor response can be defined. In particular, a first order TF can be obtained from the measurements' analysis, and defined as follows:

$$TF = \frac{\mu}{1 + s\,\tau} \tag{3.4}$$

with μ, the steady state response value, and τ, related to the speed of response.

The applied offset TF, shown in Fig. 3.9, is characterized by $\mu = 1$ and $\tau = 0.02$. For what concerns the joint angles (Fig. 3.10), the value of μ can vary, since the joint movement is not fixed with the linear coordinates, while τ is equal to 0.04, due to the slower joint response. Additionally, the delay between the input and the nominal applied offset is in the order of 15 ms, while between the input and the joint angles' movement is of 50 ms.

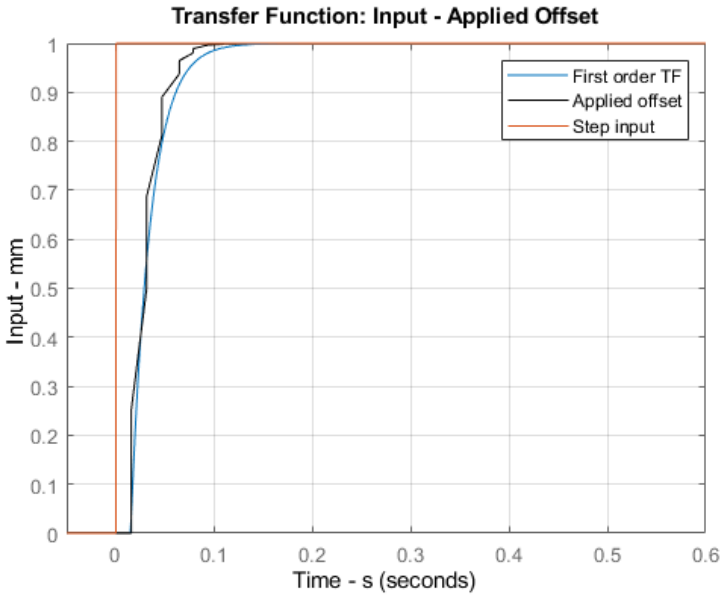
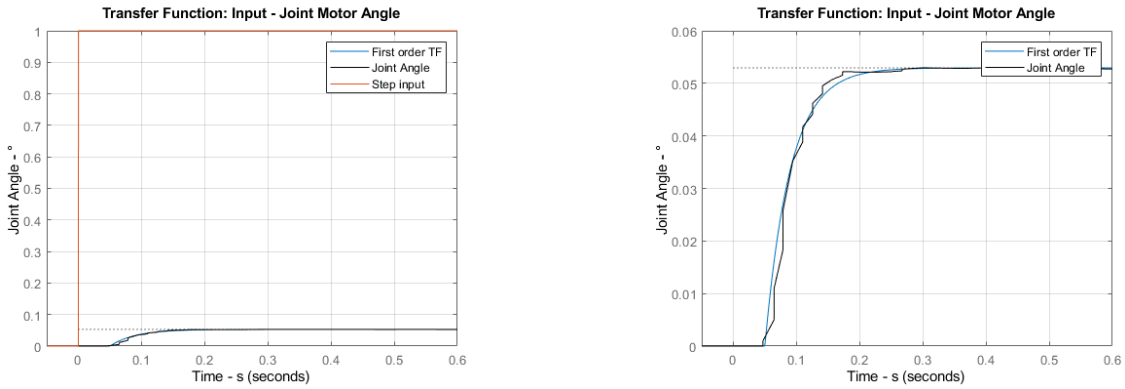


Figure 3.9: Applied offset step response



(a) Full size (b) Zoom

Figure 3.10: Joint angle step response

Therefore, if also the actual motor behavior is considered within the model, the simulated response becomes quite similar to the measured one (see Fig. 3.11). In this case, the overall response is given by the first order TF between the input and the motors and the second order TF between the motors and tool movement.

$$TF = \frac{\mu}{1 + s\tau} e^{-ds} \frac{\omega_0^2}{s^2 + 2\ \zeta\omega_0\ s + \omega_0^2} \tag{3.5}$$

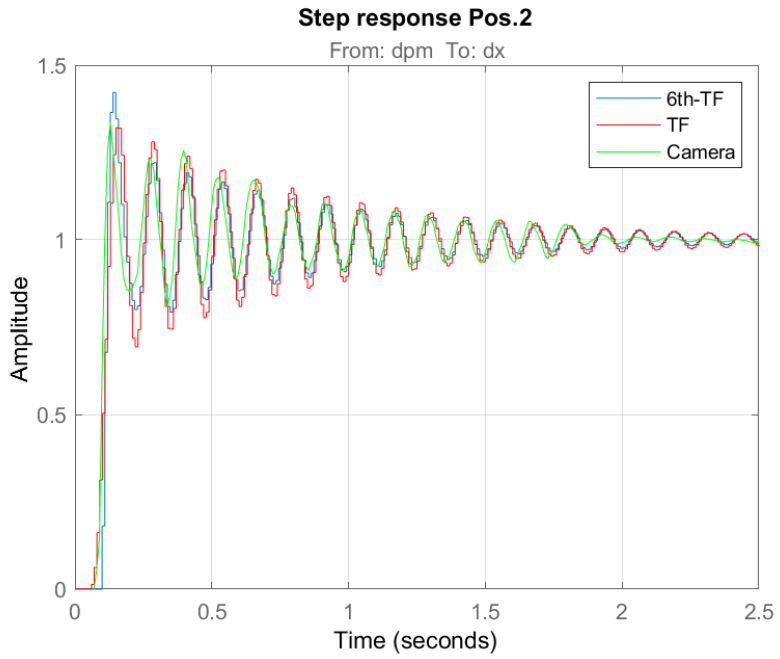


Figure 3.11: Rate limited step TF on x-axis for Pos. 2

Thus, the frequency-damping approach was carried on, and the parameters presented in Table 3.2 and Table 3.3 have been extrapolated. Then, for performing the desired interpolation over the robot workspace, the following assumptions were made:

- symmetry between x and y axes due to joint 1 rotation
- influence mainly due to radial and vertical distance
- end effector orientation influence is negligible
- more data can be averaged to obtain information over a specific area.

Thus, a fuzzy-like logic was adopted to perform the interpolation (algorithm 3.1).

Algorithm 3.1 Interpolation for Extended Kalman Filter

```
Import frequencies and damping ratios table
Get the end effector position
for each axis x, y, z, w, p, r do
    Get the average over small (1400 mm), medium (1900 mmm), and big distance (2400 mm) both for the lower (400 mm) and upper (1200 mm) positions
    if height outside height range then
        Assign extreme values
    else
        Find interpolate values between lower and upper ones thanks to bisect method
    end if
    if radius outside the distance range then
        Assign extreme values
    else
        Find interpolate values between small, medium, and big distance ones thanks to bisect method
    end if
    Use obtained values for deriving the state space matrices
end for
```

Table 3.2: Main normal frequency and damping parameters for x, y, and z

	Distance (mm)		Frequencies (Hz)			Damping ratios		
Pos.	**Radius**	**Height**	**x**	**y**	**z**	**x**	**y**	**z**
1	1400	400	9.92	5.99	7.95	0.25	0.45	0.03
2	1400	400	7.88	8.92	7.23	0.03	0.13	0.05
3	1400	400	6.81	6.93	6.57	0.48	0.22	0.24
4	1900	400	6.96	6.53	6.82	0.06	0.42	0.02
5	1900	400	7.27	2.78	7.20	0.03	0.63	0.02
6	2400	400	4.96	4.13	6.82	0.80	0.42	0.04
7	2400	400	6.38	6.50	6.52	0.18	0.05	0.02
8	2400	1200	5.92	6.07	6.08	0.08	0.02	0.02
9	2400	1200	6.29	4.36	6.42	0.03	0.28	0.02
10	1900	1200	6.63	5.00	6.79	0.04	0.24	0.03
11	1900	1200	6.45	6.42	6.48	0.02	0.02	0.01
12	1400	1200	5.95	4.53	3.80	0.30	0.34	0.01
13	1400	1200	7.57	7.32	11.5	0.01	0.37	0.43
14	1400	1200	7.60	6.55	7.44	0.02	0.19	0.10

Table 3.3: Main normal frequency and damping parameters for w, p, and r

	Distance (mm)		Frequencies (Hz)			Damping ratios		
Pos.	**Radius**	**Height**	**w**	**p**	**r**	**w**	**p**	**r**
1	1400	400	10.0	13.3	7.24	0.14	0.25	0.63
2	1400	400	10.5	8.37	17.4	0.67	0.20	0.37
3	1400	400	14.1	10.3	4.32	0.05	0.08	0.36
4	1900	400	12.9	10.5	8.21	0.58	0.42	0.57
5	1900	400	7.21	6.44	1.62	0.60	0.51	0.05
6	2400	400	0.81	11.6	11.3	0.69	0.53	0.45
7	2400	400	15.9	1.26	13.4	0.13	0.58	0.63
8	2400	1200	12.6	8.98	7.84	0.54	0.29	0.50
9	2400	1200	0.93	18.9	8.36	0.71	0.21	0.62
10	1900	1200	0.81	6.95	7.01	0.73	0.62	0.45
11	1900	1200	8.59	20.7	9.45	0.53	0.18	0.57
12	1400	1200	14.0	15.2	12.0	0.40	0.05	0.53
13	1400	1200	6.66	7.76	8.34	0.53	0.23	0.67
14	1400	1200	8.24	7.04	6.35	0.42	0.46	0.84

3.3. Static Linear Control

3.3.1. Control logic design

After having defined the system dynamics and its approximated second order TF, with its stiffness and damping, it is possible to start designing the control logic.
The first approach is done with static control: the target is a fixed point in space and the robot is perturbed around it. The key performance indicators of this case study are represented by the absolute accuracy, the compensation speed, i.e., the speed at which the position error returns below a certain threshold after being perturbed, and its ability to damp unwanted oscillations and disturbances.

The control logic works, as previously described in section 2.3 and especially in Fig. 2.5, by taking as input the position error between the actual position measurements coming from the cameras and the target pose, and providing the offset needed to compensate for it. The target pose is given at each time instant by the robot, and can be read with the *FaRoC* library. Furthermore, the control logic has to be written in discretized form, so to act at the sampling time at which the communication with the robot and the cameras operates through the *FaRoC* data interface.
The implementation and connection had already been developed and tested in a previous thesis work from Hojak [31], therefore the following work is focused on the improvements and tuning of the parameters and logic.

As stated in the robot controller description (section 2.2), the DPM functionality is used to dynamically control the tool end effector: this allows to move the robot end effector in Cartesian space by a specified amount for each Cartesian component. The input are therefore arrays that contain 6 elements: the first three referring to the x, y and z linear coordinates, while the last three are referred to the angular rotation w, p and r.
Moreover, for modal DPM with *FINE* motion, stationary tracking over a programmed pose is supported [32].

From the parameter identification, section 3.2, it can be seen that the robot behavior in Cartesian space can be approximated by a second order TF, which can change based on the movement or on the robot configuration and describes the system response to an applied offset.

$$TF = \frac{\omega_0^2}{s^2 + 2\ \zeta\omega_0\ s + \omega_0^2} \quad \text{with} \quad \omega_0 = 2\pi f, \quad \zeta = \frac{d}{2\pi}, \quad \text{and} \quad \alpha = \zeta\ \omega_0 \tag{3.6}$$

Moreover, from an analysis of the obtained values for f and ζ, have been derived the rounded average values of 7 Hz (approx. 44 rad/s) and 0.16 respectively.

Additionally, it can be defined a first order TF between the input sent by the user and the joint motor movement.

$$TF = \frac{\mu}{1 + s\tau} \tag{3.7}$$

The system robotic arm dynamic is thus determined by a first order and second order TF, representing the relation between the input and the joint angles and the joint angles and the tool movement respectively.

For what concerns the control logic, the modal DPM function provides an integral action: each input is sent and summed to the already applied offset. This would allow the zero error at infinite time, with the end effector converging to the target position.

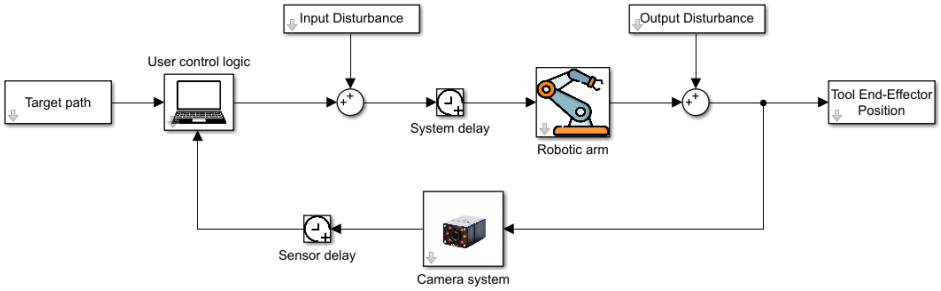


Figure 3.12: Scheme of the control logic system and its delay times

The controlled system can be described with the given TF, with C(s) being the control logic used for computing each input offset, followed by the integrator, the first order TF of the joint angles with its delay, and the second order TF of the structure:

$$TF = C(s)\frac{1}{s}\frac{\mu}{1 + s\tau}e^{-ds}\frac{\omega_0^2}{s^2 + 2\ \zeta\omega_0\ s + \omega_0^2} \tag{3.8}$$

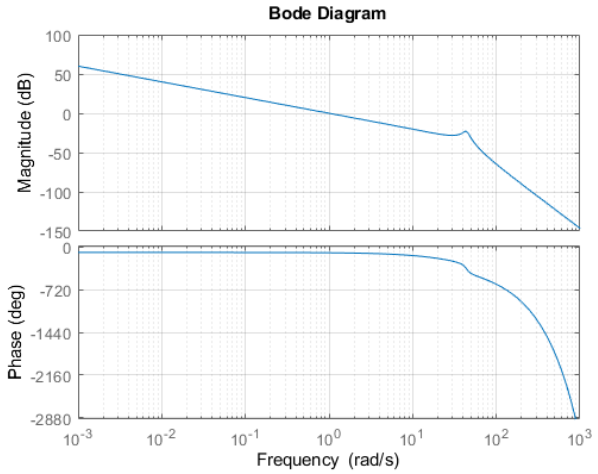


Figure 3.13: Bode diagram of the overall uncontrolled system TF

The first approach for the control action has been done with a linear control logic. By looking at the TF bode diagram in Fig. 3.13, the following control logic, which should be fast and at the same time reject the higher frequency disturbances, has been designed:

$$C(s) = 100 \cdot \frac{s + 0.01}{s \cdot (s + 30)} \ . \tag{3.9}$$

The final TF of the controlled system, constituted by the control logic and the system dynamics, has the following Bode diagram (with phase margin of 65 deg and crossover frequency of 3.3 rad/s) and simulated step response:

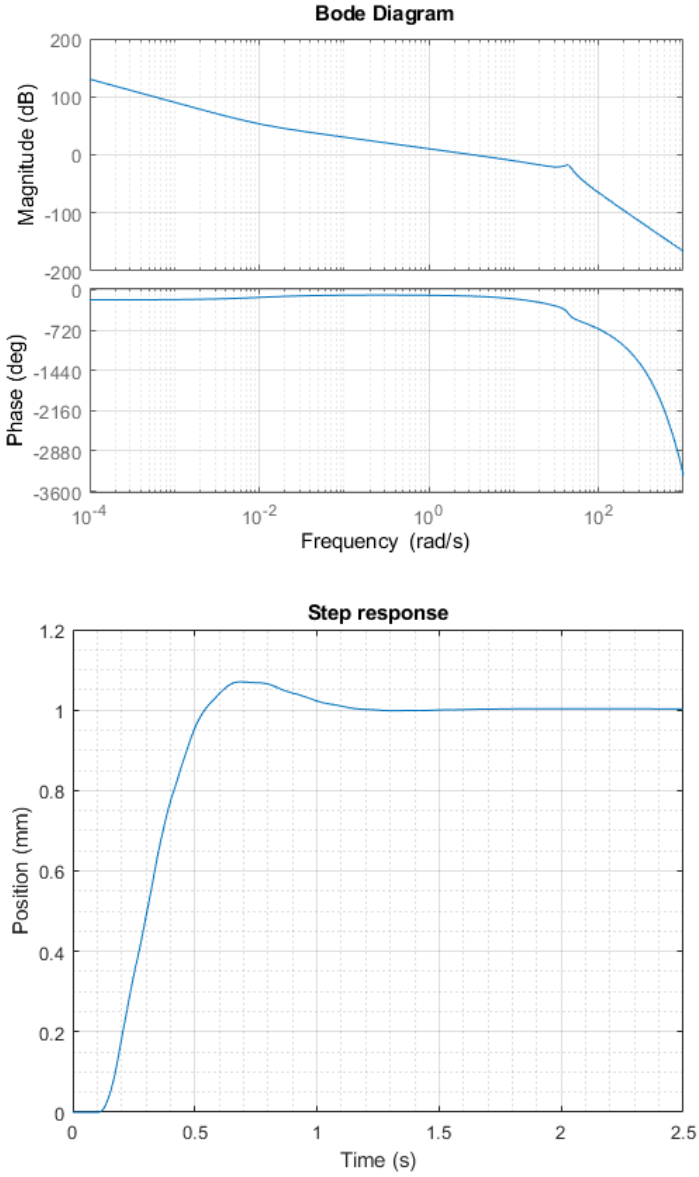


Figure 3.14: Bode diagram and theoretical response for the controlled system

The controller is then discretized using the Tustin method. Regarding the sampling time, it is determined by different factors: the camera sampling frequency, 150 Hz, which accounts for circa 7 ms, the computing time of the pc that executes the control logic computations and the rate at which the robot controller is able to apply the inputs.
The latter can reach a ceiling: if too many inputs are sent in a very short time, the robot controller will not be able to apply them all. Indeed, if the input sent by the PC has not been fully executed by the robot, the controller will ignore all the inputs coming in the meantime.
It has been possible to keep the sampling time around a constant value by using some pauses (sleeps) in the control logic, which has allowed to reach a sampling time of circa 15 ms. As a result, the discretized control logic TF is the following:

$$C = 100 \cdot \frac{s + 0.01}{s \cdot (s + 30)} = \frac{0.6123\, z^2 + 9.184 \cdot 10^{-5} z - 0.6122}{z^2 - 1.6327\, z + 0.6327} \,. \tag{3.10}$$

The nominator is multiplied by the sampling time, to account for the integration action of DPM, which works as a Forward Euler integrator. Eventually the control action is rearranged to obtain the explicit input action formula, that can be computed iteratively at each time step:

$$u_k = 0.009184\, e_k + 1.378 \cdot 10^{-6} e_{k-1} - 0.009183\, e_{k-2} + 1.6327\, u_{k-1} - 0.6327\, u_{k-2} \,. \tag{3.11}$$

The control logic and the system response has been then simulated on Simulink to check the correct functioning. The step tracking, input disturbance and output disturbance step response gave the following results:

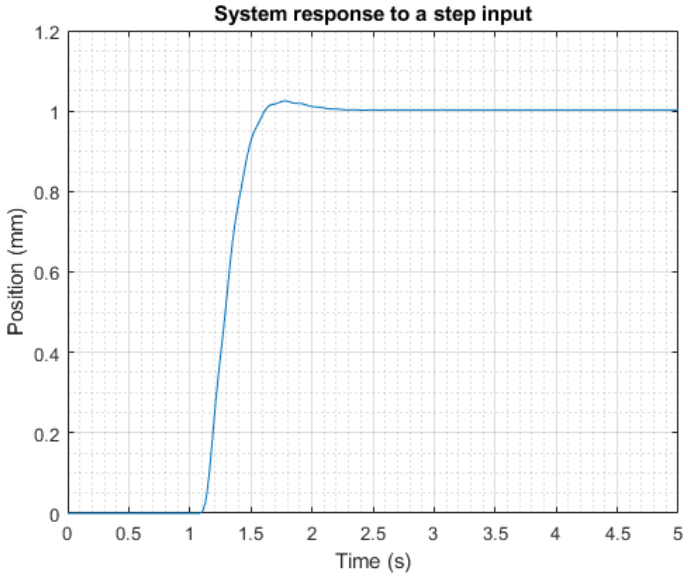
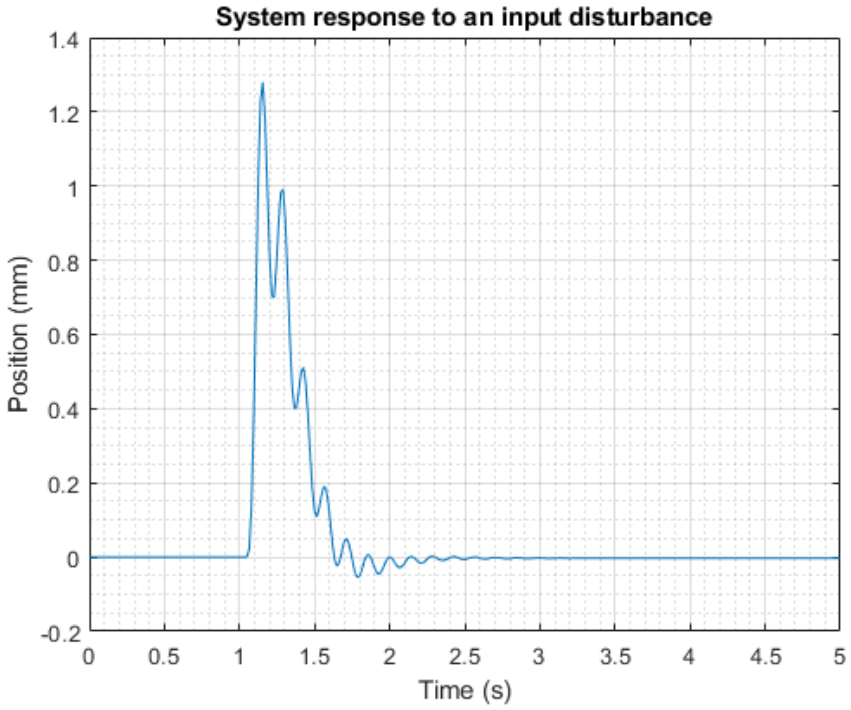
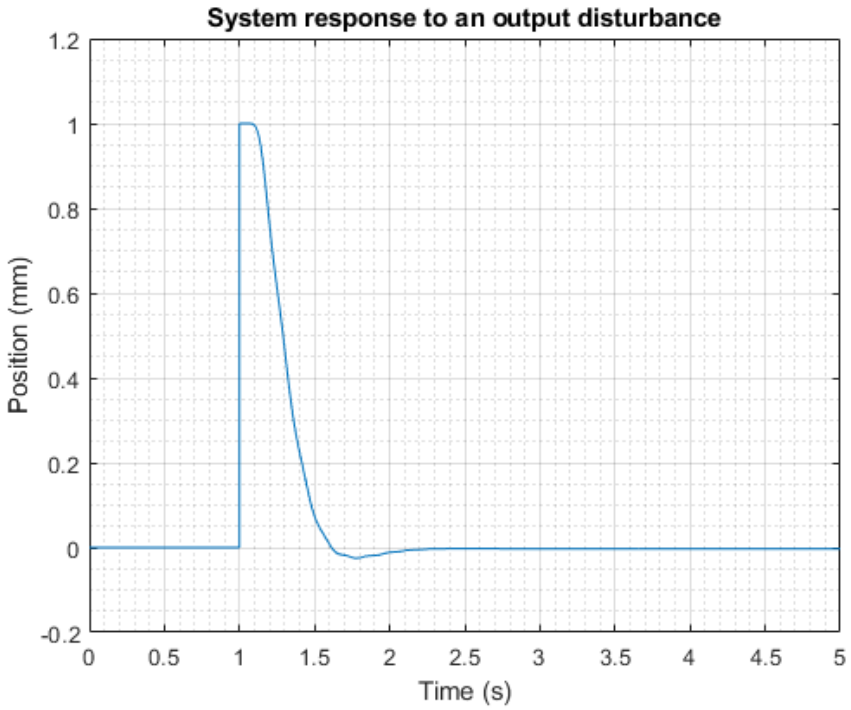


Figure 3.15: Simulated step response



(a) Input disturbance



(b) Output disturbance

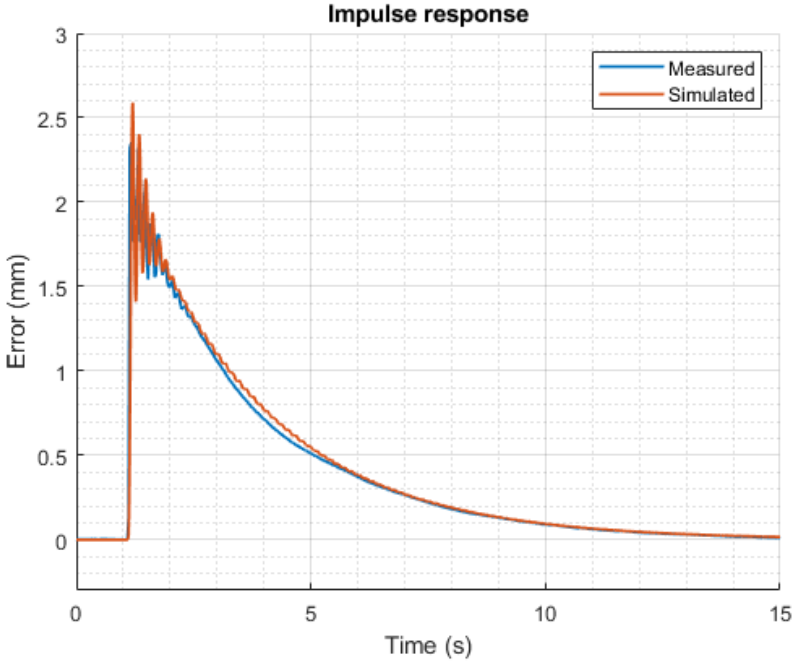
Figure 3.16: Simulated system responses to step disturbances

3.3.2. Tests and results

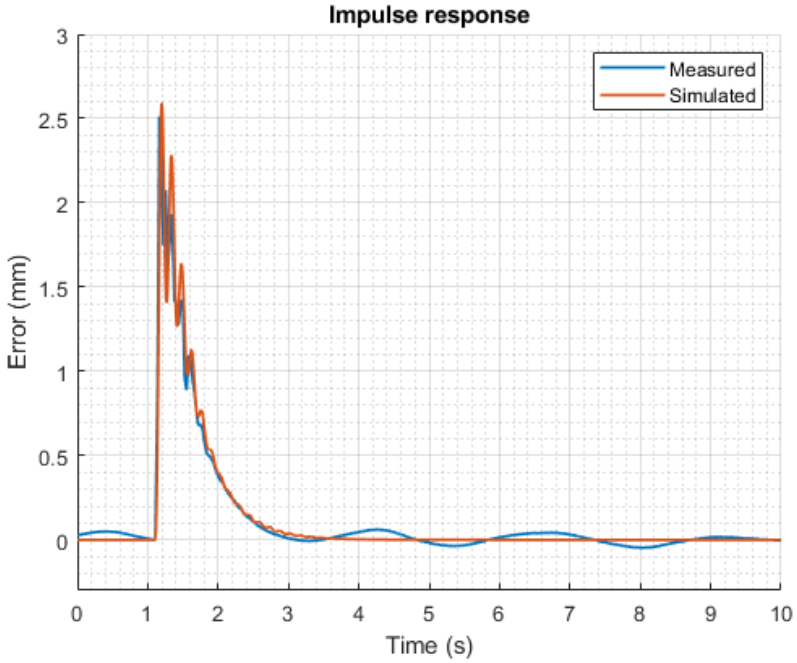
To test the control logic, a simple *Karel* program has been designed: the robot arm start from an initial position and then moves to the target one, according to its kinematic model. Once the robot has reached its (supposed) target, the DPM function is turned on and the controller corrects the end effector position in order to reach the desired pose.

Eventually, when the target has been reached, and the end effector is stabilized around it, the robot is perturbed by manually sending offset inputs with the Teach Pendant, simulating input disturbances in the closed loop circuit.
At first instance, the test have been done with just a proportional control, with each input being equal to the measured error multiplied by a fixed gain. Thus, in Fig. 3.17 are presented the results for two test conducted with different values of control gains.
The increase in the gain leads to an increase in the speed at which the robot returns to the neutral position, as expected. However, a too large value of proportional gain leads to an oscillating behavior, with low frequencies oscillations around the neutral position.

Furthermore, it can be evinced how the theoretical model presents a similar response time to the actual test (see Fig. 3.18). However, some differences can be seen during the transitory phase, where some non-linearities can influence the system, and more importantly at steady state, where the higher gain proportional control generates some low frequency oscillations, which in the theoretical model are not present.



(a) Kp = 0.01



(b) Kp = 0.05

Figure 3.17: Response to an input disturbance for static proportional control

Eventually, the test with the discrete controller have been carried out. However, as already highlighted by the proportional control response, one of the main limits of the model is the undamped oscillation at low frequency, which end up affecting the performances of the control. Indeed, the unforecasted oscillations can get amplified by the control action, thus making the system unstable.
Therefore, it has been necessary to lower the gains of the control due to the oscillations that generate once the system reaches the target pose, ending up with performances similar to the proportional control, as shown in Fig. 3.18.

The oscillating behavior is quite noticeable especially on the z-axis component of the robot dynamics, shown in Fig. 3.19, the one subject to the force of gravity. In particular, the z-axis dynamics has an impact on the other coordinates, generating a 3-D oscillation in space.
The system oscillations happen at a frequency of circa 0.3 Hz.

In order to compensate for this fact, a lower value of gains can be used on the z component, thus allowing to keep the performances on the other axis. However, the proposed approach, with a linear control action, does not allow increasing the performances too much, limiting the system response to accurate control.

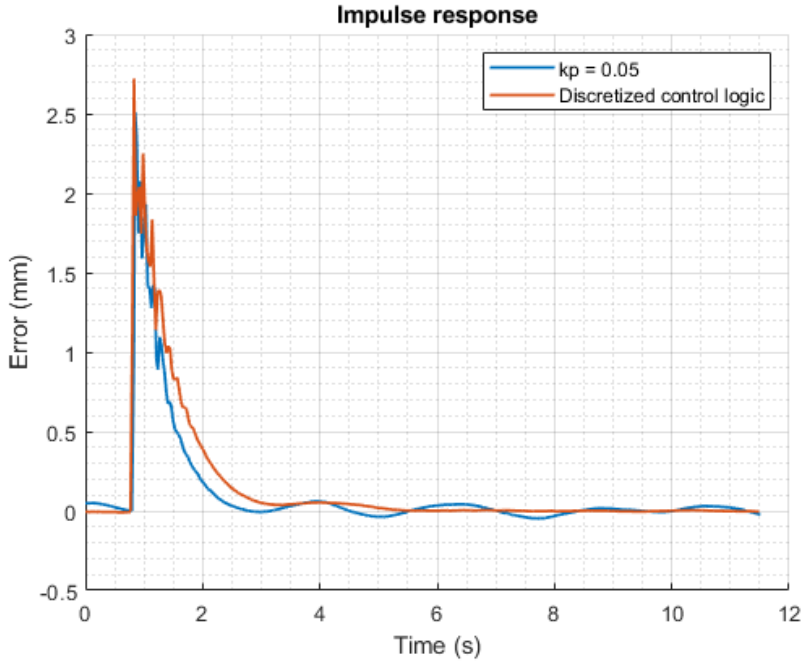


Figure 3.18: Static impulse response

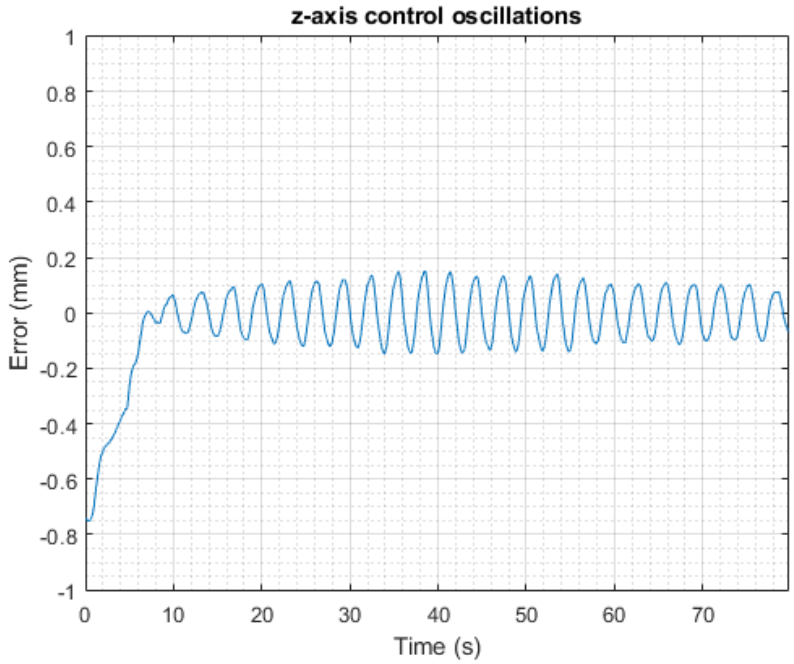


Figure 3.19: Z-axis oscillatory behavior for high control gains

3.4. Fuzzy Logic Control

Since the controlled system displays some limitations, in particular for what concerns non-linearities and complex dynamical behavior, the need for a different approach than linear control arise.
In particular, a fuzzy logic control has been implemented and tested.

3.4.1. Fuzzy logic theory

In fuzzy logic, as opposed to classical or Boolean logic, where the truth value can be only either 0 or 1 (completely true or completely false), the truth can vary gradually between 0 and 1, according to defined rules.
A fuzzy logic reasoning can be split into three processes [40]:

1. Fuzzyfication:
   converts classical data into fuzzy data, using defined fuzzy sets and Membership Functions (MFs).
   "A fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function, which assigns to each object a grade of membership ranging between zero and one" , [41]. For instance, while in boolean logic a value can be only 0 or 1, so for example a temperature can be either hot or cold, in fuzzy logic the temperature can have varying degrees of definition, and not belonging completely to hot or completely to cold. The process is done through the application of the initially defined MFs, for both the inputs and the outputs

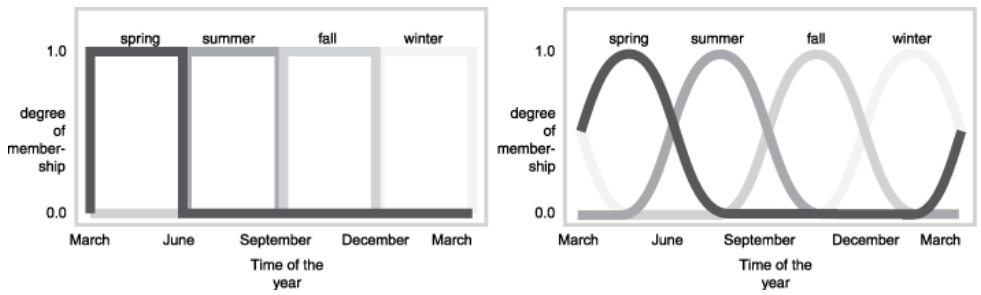


Figure 3.20: Seasons fuzzy sets and MF, courtesy of MATLAB [42]

In case of the above Fig. 3.20, the difference between the classical boolean logic and fuzzy logic is shown with the seasons example. In classical logic, the seasons are well-defined, with sharp borders that corresponds to exact dates on the calendar.

In fuzzy logic, instead, the seasons, i.e., fuzzy sets, can be treated with smooth boundaries, and each calendar month can have partial belonging to one seasons or another.

2. Fuzzy Inference Process:
   combines the membership functions and sets with the control rules to derive the fuzzy output.
   The rules are defined based on the knowledge of the system and the experience. This process is defined by a series of *If-Then* cases, that relate the input with the respective output in fuzzy set form.
   For instance, in the case of control system, it can be of the form: If (system parameter) is (fuzzy set 1) – Then (control gain) is (fuzzy set 2).
   A common way of defining the interference process is through a fuzzy mapping rule, by the use of a table that defines the relationship between the input and the output, in terms of fuzzy sets.

3. Defuzzyfication:
   returns, through the application of the interference process and the defined MFs, the final output value. There can be different ways of computing the output result. One of the most popular methods is the Center of Gravity method. In this method, based on the belonging to each output fuzzy set, a corresponding area of the output MFs can be defined, and eventually the center of gravity of the combined areas of all the MFs is calculated.

The fuzzy logic process can be translated into a control logic. In this case, the input can be a system parameter, like position, velocity, temperature, and the output are the control gains. The rules of the fuzzy logic can then be defined based on experience and the knowledge of the system.

Some example of fuzzy control in robotics applications are provided by Li et al. [43], where a camera vision based control system has been tested on a *KUKA KR500-3* IR. Li et al. [43] suggested an implementation for a PID controller based on the following rules:

- if the error magnitude is big, the integral action should be set to zero and the derivative gain should be small, while the proportional gain appropriately selected to avoid vibrations.
- if the error magnitude is medium, all the gains should be appropriately increased.
- if the error magnitude is small, the proportional action can be increased to amplify the control effect, as well as the integral effect, while the derivative one should not be too large.

A different approach has been used by Adar et al. [44], where both the error magnitude and the error speed are considered in the computation of the gains. The proposed rules for the proportional gain can be summarized in a 2-D table:

Table 3.4: Rule for $\boldsymbol{K_P}$ based on position and velocity errors, courtesy of Adar et al. [44]

$\mathbf{e} \setminus (\mathbf{d}e/\mathbf{d}t)$	N	Z	P
N	small	medium	small
Z	medium	big	medium
P	small	medium	small

In this case, also the error speed must be available, and serves as input into the fuzzy rules. The final value of proportional gain is then determined by the membership function of the two inputs, in this case negative, zero or positive, and the output, small, medium, or big. Similar rules can also be written for the integral and derivative gain.

A more complex example is instead given by Ligutan et al. [45], where the control action for the joint movements of a 6-DoF robot are functions of three inputs: the base joint angles and the Cartesian error along the x and y coordinates.

3.4.2. Fuzzy logic implementation

In the context of this thesis, it has been designed a fuzzy logic control that changes gain values with respect to the error magnitude. The error speed has not been considered since it not directly available, and its estimation comes with additional delays that could degrade the system performance.
The first approach has been done with a logic with two fuzzy sets, small and big, for both the input (the error magnitude), and the output (the control gain).
A more detailed fuzzy set definition, with the addition of intermediate sets between small and big, such as medium, as not been considered at first, so to keep the logic more simple to test and tune.
Two possible combinations are available:

1. the gain of the control is big when the error is large, so to be fast and approach the neutral position in a short time, and gets progressively lower when the error is small, so not to cause any unwanted oscillation.
2. the gain is smaller when the error is large, so that the offset input is kept within a reasonable amount and doesn't cause any overshoot, and the gain gets bigger as the error gets smaller, meaning stiffer control around the neutral position.

The MFs for the input and the output have been defined with linear Z and S functions.

The rules set is quite simple, since the input and output are only two, therefore two rules are needed to completely define the inference process:

- IF error IS big, THEN gain IS big
- IF error IS small, THEN gain IS small

The output gains are inverted for the second case. This would result in a linear change of gains as the error changes between 100 percent big to 100 percent small. Additional fuzzy sets, between the small and big, would result in a more complex transition between the maximum and minimum values, like a parabolic curve or intermediate changes of slope. However, the main trend remains the same.

Eventually, the logic is implemented in Python.
To replicate the gain values of the control, a lookup table has been defined, which correlates the error with the corresponding gain.
The linear profile in Fig. 3.22 can be obtained by defining 4 values: the gain at low and high error magnitude and the threshold between which the gain changes linearly.

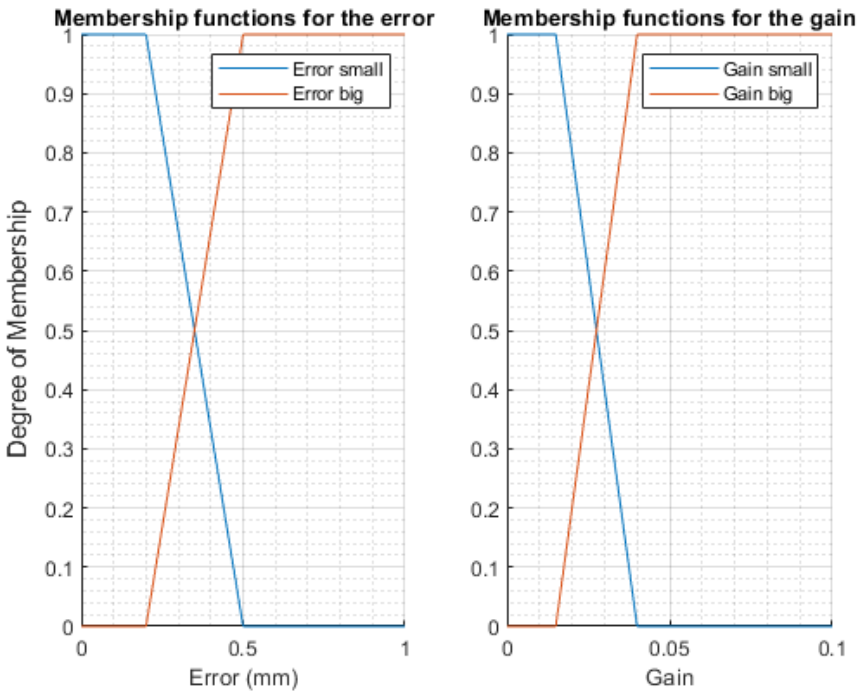


Figure 3.21: Error fuzzy sets and membership functions

This logic is intuitive and easy to be computed by a PC, therefore it does not add any further delay to the system. The tests can be conducted by changing one or two parameters and fixing the others, so to compare the different cases and results.

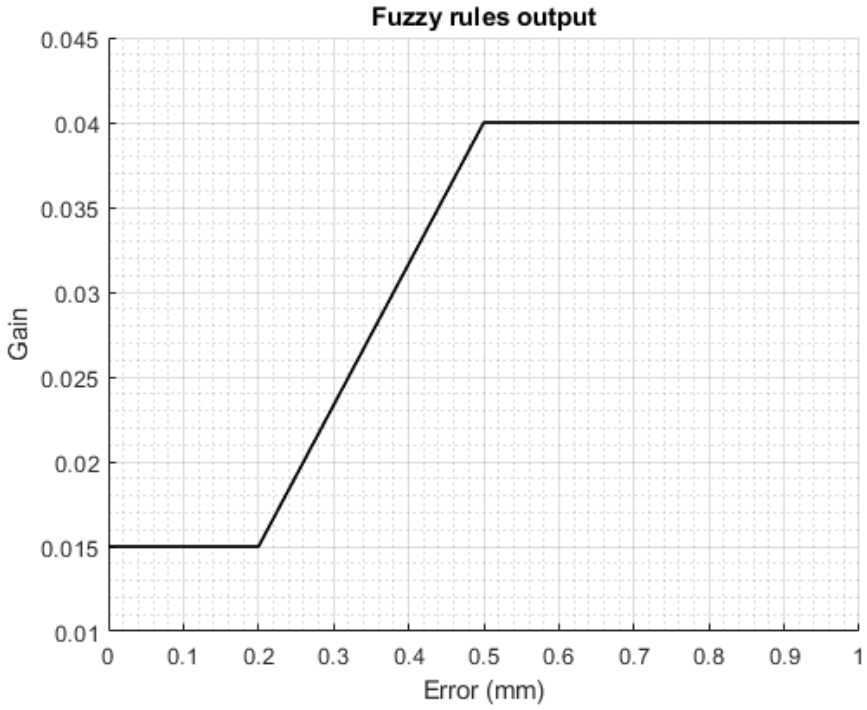
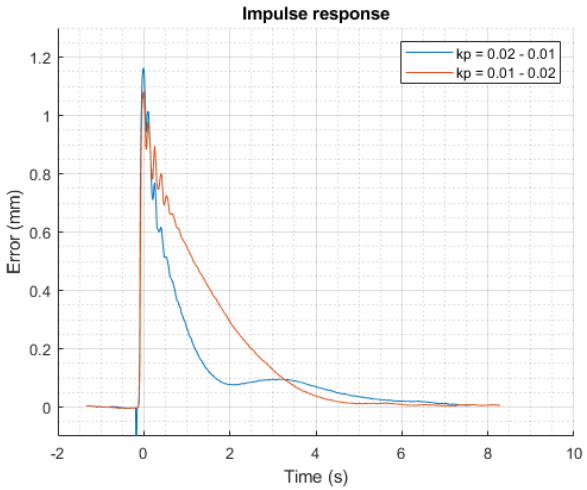


Figure 3.22: Gain fuzzy sets and membership functions

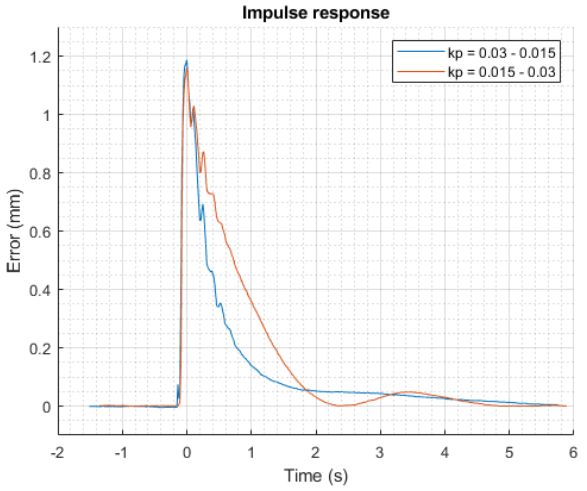
3.4.3. Tests on fuzzy logic

The first tests have been done to compare the two different rules implementation, with increasing or decreasing gains based on the error magnitude.

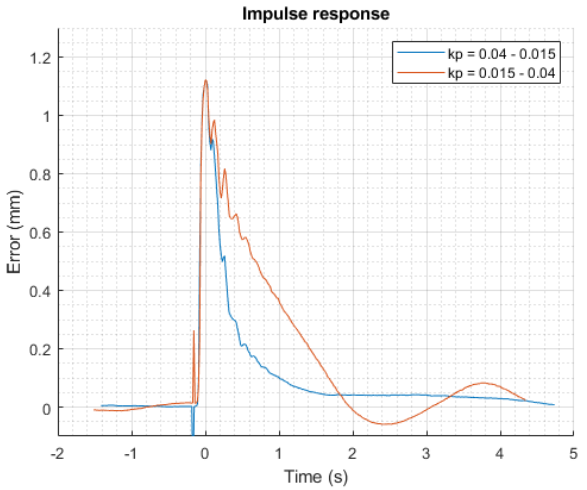
In the following Fig. 3.23, it can be seen the tests done with same control gains values but with opposed rules. The blue line corresponds to decreasing gains with decreasing error magnitude, while the orange one is the opposite.
The three images display tests done with gradually increasing values of gain pairs. It can be noticed that the decreasing gain with decreasing magnitude shows a faster response to the impulse disturbance if compared to the opposite case. Moreover, the higher gains at low amplitudes can still generate overshoot and unwanted oscillations. Therefore, the first approach is the more suitable one, which was also as expected based on the results obtained in the linear control case.



(a) Gain pair: 0.2 and 0.1



(b) Gain pair: 0.3 and 0.15



(c) Gain pair: 0.4 and 0.15

Figure 3.23: Impulse responses with different gain control rules

Set the control logic type, the optimal the threshold parameter have to be determined. In Fig. 3.24, it can be seen the impact of the lower threshold of the MFs of the error amplitude. In the image, the control gain is set to the lower value for the error magnitude of respectively: 0.1, 0.15, 0.2 mm. This slows down the response at low error amplitude, with the lines becoming lest steep at different instants.

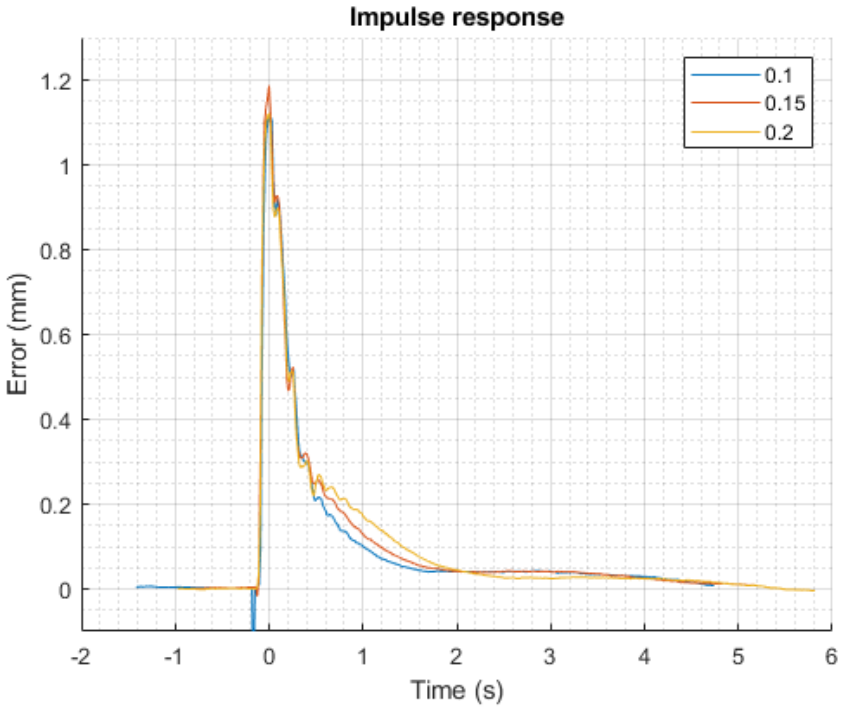


Figure 3.24: Impulse response with threshold: 0.1 - 0.15 - 0.2

Similar results can be obtained by changing the higher threshold, since the gain would start linearly decreasing at higher error magnitudes, therefore slowing down the response, but reducing the risk of instability and overshoot.

Another aspect to be considered is the steady state response. Indeed, one of the main challenges of the linear control is the instability generated by higher gains, needed instead for a faster response.

Fig. 3.25 display the system behavior once reached the target pose and no impulse disturbance is acting on it. The three measurements are referred to tests with increasing values of gains for low magnitude error. As the gain value increases, the oscillations increase as well, as the system becomes more and more unstable.

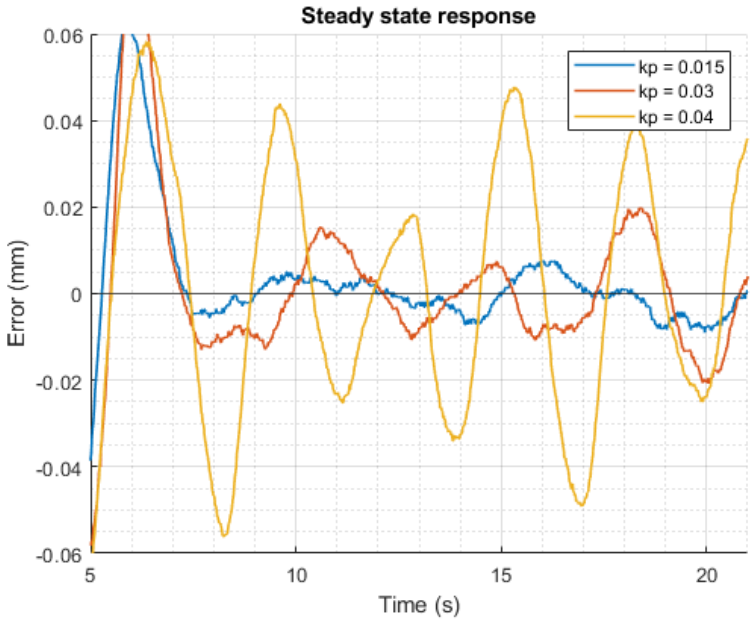


Figure 3.25: Steady response with gains: 0.015 - 0.03 - 0.04

3.4.4. Multi-pose test

The fuzzy logic control has been tested in different working positions, already used during the parameter identification phase, so to identify the optimal values of control gains, based on the pose.

In particular, the test have been conducted along the following paths:

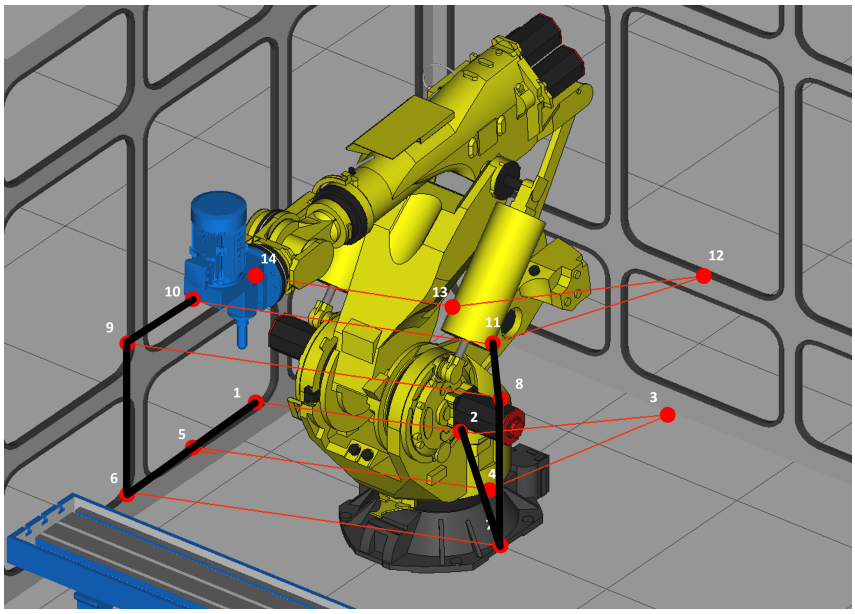


Figure 3.26: Poses used for the static control tests highlighted in black

As it can be noticed from Fig. 3.26, the two paths are respectively parallel to the x-axis, and at 45 degrees between the x and y axes. This test would therefore allow determining the behavior of the controlled system with respect to the distance of the tool center point from the robot frame origin, the height and the angular position.

The test have been performed by selecting a fixed set of controller parameters for all the robot poses and analyze the different controlled response based on the target position. The resulting controlled response is shown in Fig. 3.27 and Fig. 3.28. In each position, the robot is perturbed by sending 1 mm inputs to x, y and z channel, in this order.

It can be noticed from the measured error how the behavior changes and some observations can be made:

- the poses with the tool at higher altitude seems to be more stable and oscillate less.
- at low height and along the x-axis, the distance in position 5, with the arm in neutral pose, has the most stable behavior, followed by position 1, contracted, while the extended pose, position 6, is the most unstable. In particular, the biggest error appears to happen due to input sent on the y channel, cause by the higher rotational inertia.
- at 45 degrees, the x and y axes disturbances show a similar response, since they involve the same joint movements.

From these results, multiplying coefficients could be defined so that the control gains are scaled adequately depending on the robot pose, so to avoid getting close to instability. In particular, they can be functions of the tool end effector distance and height, and the first joint angle.
Alternatively, the maximum control gains could be fixed based on the most unstable pose.

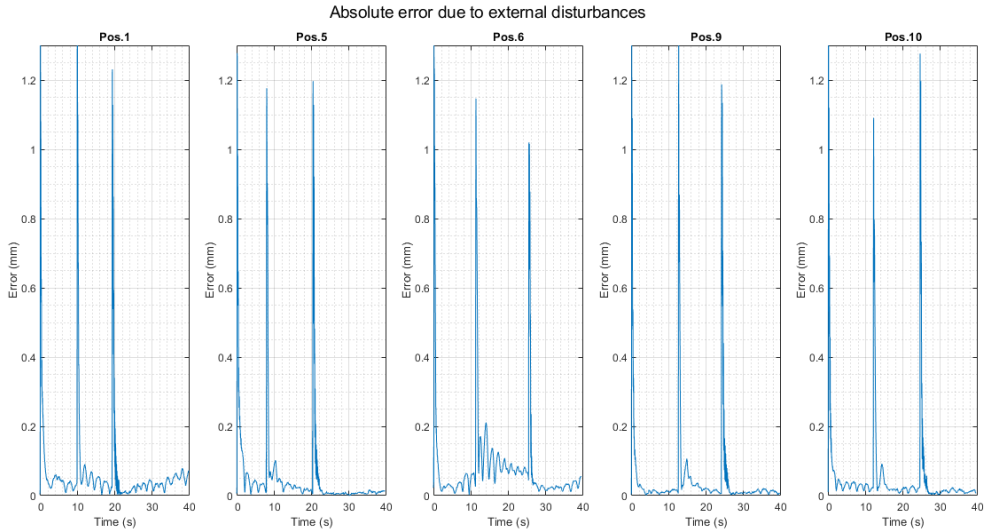


Figure 3.27: Absolute error - position along the x-axis

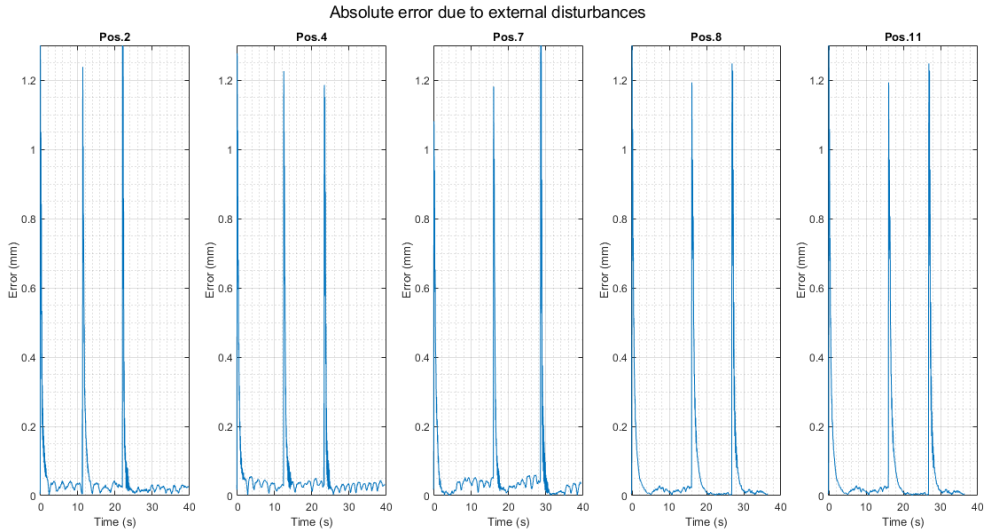


Figure 3.28: Absolute error - position at 45°

3.5. Kalman Filter

In this chapter, the design and implementation of the Kalman filter will be presented and analyzed. The motivation that led to the need of a Kalman filter model is the possible presence of noise coming from the camera measurements during the dynamic path control, which can affect the pose estimation accuracy and reduce the performances of the system. The same behavior was also encountered by Gharaaty et al. [21], which in their work used the root mean squared RMS technique to filter the measurements.

Indeed, the following Fig. 3.29 shows the resulting measurements for the pose error, taken by the ARTTRACK5 (section 2.5.2) camera system, during the first iteration of the dynamic test with circle path, where no control is applied.
As it can be noticed, the measurements are affected by high frequency oscillations, which can be due to camera noise.

Therefore, thanks to the modeling of the dynamical system in section 3.2, the Kalman filter is expected to improve the controller stability and its accuracy performances, even though possible delays due to the incremented computation needed might affect the controller responsiveness.
Lastly, the tuning of the filter parameters, especially of the disturbance covariance matrix, would be a pivotal activity in the optimization process, and in assessing the best performing out of the different type of Kalman filter implemented.

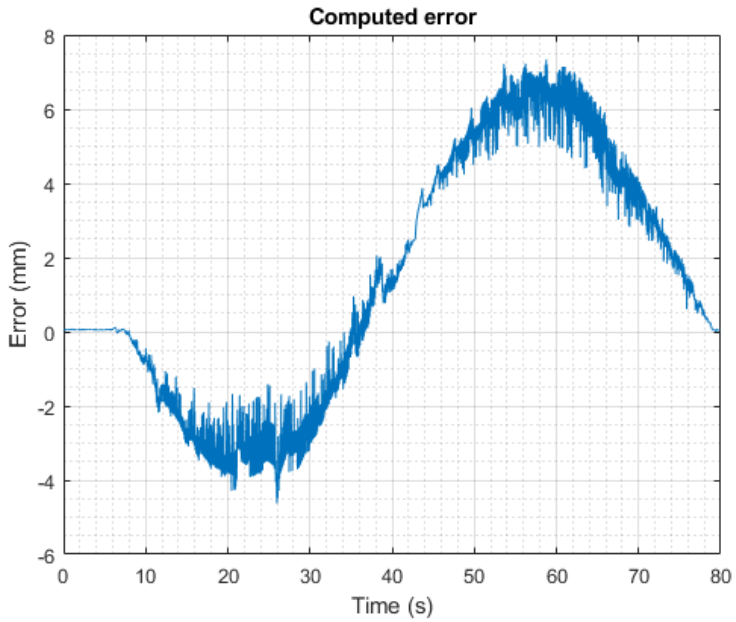


Figure 3.29: Measured noise in pose error data during dynamic test

3.5.1. Kalman model development

The Kalman filter is a strategy to estimate state variables of a process from noisy measurements, minimizing the mean of the squared errors, even if the modeled system is not precisely known [46]. In particular, it is an optimal estimator in the case of linear dynamic systems and Gaussian random variable disturbances [47].

The filter uses the theoretical model of the system and the measurements coming from the sensors, in order to estimate the state based on the reliability of the two data, weighting each one depending on the uncertainties of the model and the noises of the sensors. In this way, the performance can vary, either relying more on the internal predicted state or on the measurement information [47].
Moreover, it can work in real-time, by computing the algorithms online, therefore allowing to be used for control purposes or to predict the state of the system at the next instant. Additionally, the Kalman filter can also be used for non-linear systems, in the form of Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF).

The continuous Kalman filter, also known as the Kalman-Bucy filter, is constituted by a set of equations, that are computed recursively as soon as the sensors provide new data. Considering a Linear Time-Varying (LTV) system of the type:

$$\begin{cases} \dot{\boldsymbol{x}}(t) = A(t)\boldsymbol{x}(t) + B(t)\boldsymbol{u}(t) + L(t)\boldsymbol{w}(t) \\ \boldsymbol{z}(t) = C(t)\boldsymbol{x}(t) + D(t)\boldsymbol{u}(t) + \boldsymbol{v}(t) \end{cases} \tag{3.12}$$

where the state vector $\boldsymbol{x}$ is composed by the position and velocity error, $\boldsymbol{u}$ is the input vector, $\boldsymbol{w}$ is the disturbance vector, and the measurement vector $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{v}$ is the sum of the output and measurement noise vectors. Moreover, A is the state (or system) matrix, B and C are respectively the input and output matrices, D is the feed-through matrix, and lastly L is the disturbance matrix.

Then, the Kalman-Bucy filter is defined by the following equations and two initial conditions on $\boldsymbol{x}$ and P:

1. State estimate:
$$\dot{\hat{\boldsymbol{x}}}(t) = A(t)\hat{\boldsymbol{x}}(t) + B(t)\boldsymbol{u}(t) + K_0(\boldsymbol{z}(t) - C(t)\hat{\boldsymbol{x}}(t)) \tag{3.13}$$
2. Covariance estimate:
$$\dot{P} = AP + PA^T + LQ'L - PC^TR^{-1}CP \tag{3.14}$$
3. Filter gain computation:
$$K_0 = PC^TR^{-1} \tag{3.15}$$

The estimated state at each iteration is therefore computed by combining the prediction based on the theoretical system model and the input used, and corrected by adding the error between the estimated state and the measurements, properly scaled by a gain K_0. The larger the gain, the higher the weight on the measurements, while the smaller the gain, the higher the estimated state influence, as it is evinced from Eq. (3.13).

K_0 is computed based on state covariance estimate P, derived from a nonlinear, ordinary differential equation (Eq. (3.14)) governed by the statistics of the expected measurement error covariance (R) and disturbance covariance (Q') [47].

As previously stated in section 3.2, it has been decided to model the IR kinematic with a transfer function of the second order, determining the dynamic parameters through the step response tests and linearizing the system around a finite set of points. Since higher-order dynamic terms are discarded, modeling errors are introduced, and they must be accounted in the noise covariance with appropriate tuning of the parameters [47].

The second order TF, taken from Eq. (3.3), is as follows:

$$TF = \frac{\omega_0^2}{s^2 + 2\ \zeta\omega_0\ s + \omega_0^2} \ \text{ with } \ \omega_0 = 2\pi f, \ \ \zeta = \frac{d}{2\pi}, \ \text{ and } \ \alpha = \zeta\ \omega_0 \tag{3.16}$$

with f being the principal natural frequency of the system in Hz, and d the damping.

Therefore, the Linear Time-Invariant (LTI) system is characterized by the following matrices:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\alpha \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} 0 \\ \omega_0^2 \end{bmatrix}, \ \boldsymbol{L} = \begin{bmatrix} 0 & 0 \\ 0 & \omega_0^2 \end{bmatrix}, \ \boldsymbol{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \boldsymbol{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.17}$$

In this system (Eq. (3.12)), the input u is the total applied offset to the robot end effector, i.e., the resulting sum of the offsets applied to the TCP up to that moment, according to Modal DPM offset mode (see section 2.2). Therefore, it is already considered the integral action of all the step inputs sent by the controller.

However, the previously defined Kalman-Bucy filter, is referred to continuous system, while the robot control logic is working in samples, with a sampling time of approximately 15 ms, due to transmission and control cycle time [31].

Therefore, the discrete Kalman filter form is used, based on *a priori* and *a posteriori* state estimates [46]:

1. State estimate extrapolation:
$$\hat{\boldsymbol{x}}_i^{(-)} = \Phi_{i-1}\hat{\boldsymbol{x}}_{i-1} + \Gamma_{i-1}\boldsymbol{u}_{i-1} + \Lambda_{i-1}\boldsymbol{w}_{i-1} \tag{3.18}$$

2. Covariance estimate extrapolation:
$$P_i^{(-)} = \Phi_{i-1}P_{i-1}^{(+)}\Phi_{i-1}^T + Q_{i-1} \tag{3.19}$$
$$Q_{i-1} = \Lambda_{i-1}Q'_{i-1}\Lambda_{i-1}^T \tag{3.20}$$

3. Filter gain computation:
$$K_i = P_{i-1}^{(-)}C_i\ [C_iP_i^{(-)}C_i^T + R_i]^{-1} \tag{3.21}$$

4. State estimate update:
$$\hat{\boldsymbol{x}}_i^{(+)} = \hat{\boldsymbol{x}}_i^{(-)} + K_i\ [z_i - C_i\hat{x}_i^{(-)}] \tag{3.22}$$

5. Covariance estimate update:
$$P_i^{(+)} = (P_i^{(-)^{-1}} + C_i^TR_i^{-1}C_i)^{-1} \tag{3.23}$$

   Or alternatively, for faster computation:
$$P_i^{(+)} = (I - K_iC_i)P_i^{(-)} \tag{3.24}$$

where Φ, Γ and Λ are the state-transition matrices, which allows deriving the state at a generic time t starting from t_0, due to respectively the state, the input and the disturbances.
In fact, in the case of Linear Time-Invariant (LTI) systems, where A, B, L are constant, and $\boldsymbol{u}(t)$ and $\boldsymbol{w}(t)$ do not vary in the time interval, the computation of their integral form is reduced to an infinity matrix series, called matrix exponential:

$$\begin{aligned}
\boldsymbol{x}(t) &= \boldsymbol{x}(t_0) + \int_{t_0}^{t} \left[A\boldsymbol{x}(\tau) + B\boldsymbol{u}(\tau) + L\boldsymbol{w}(\tau)\right] d\tau \\
&= \Phi(t-t_0)\boldsymbol{x}(t_0) + \Gamma(t-t_0)\boldsymbol{u}(t_0) + \Lambda(t-t_0)\boldsymbol{w}(t_0) \\
\text{with} \qquad \Phi &= e^{A(t-t_0)} \\
\Gamma &= A^{-1}(e^{A(t-t_0)} - I)\ B \\
\Lambda &= A^{-1}(e^{A(t-t_0)} - I)\ L
\end{aligned} \tag{3.25}$$

In this form, the filter algorithm resembles a *predictor-corrector* one, with *time update* equations (Eq. (3.18) and Eq. (3.19)) predicting the next time step and *measurement update* equations (Eq. (3.21), Eq. (3.22), and Eq. (3.23)) apporting the corrections [46].

The discrete Kalman filter works under the following assumptions:

1. The matrices Φ, Γ, Λ and the control action $\boldsymbol{u}$ are known without errors; this can be formalized by saying that the expected value is known and that the value of the covariance is zero
2. The initial conditions are Gaussian random variables
3. Input disturbances $\boldsymbol{w}$ and $\boldsymbol{v}$ are random Gaussian with zero mean and are uncorrelated with respect to each other (Q' diagonal) and with respect to time
4. Cross variances between $\boldsymbol{u}$ and $\boldsymbol{w}$ are null, i.e., disturbances are completely independent of the control actions.

Anyhow, the first and third assumptions do not completely hold for the robotic arm system. Indeed, the system is not linear, its dynamic is pose dependent, and the disturbances are not random Gaussian, although they can be used as a first approximation for the implementation of the filter.
In fact, the matrices Q' and R, that weight the uncertainties respectively on the model and on the measurements must be defined and can be tuned to mitigate this issue. The measurement noise covariance R can be evaluated taking some sample measurements and therefore assumed to be known, while the process noise covariance Q', which is more difficult to determine, can be selected by tuning [46, 47].

For example, Fig. 3.30 shows a Matlab simulation of the influence that different order of magnitude of the disturbance covariance matrix Q over the noise covariance R might have on the Kalman filter estimation.

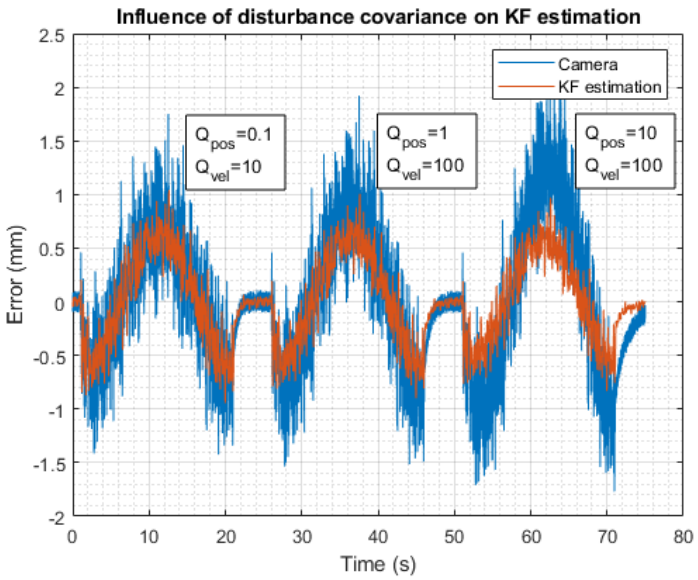


Figure 3.30: Influence of disturbance covariance Q in the Kalman filter

3.5.2. Kalman Filter expansions

Despite the Kalman-Bucy filter robustness, the estimation performances are degraded by the presence of system uncertainties. Therefore, the Kalman model can be further expanded to account for disturbances or non-linearities. In particular, in this work, have been implemented an augmented Kalman filter (AKF) which includes the disturbances as state variable, and an extended Kalman filter (EKF) taking in account the positional dependence of the dynamic characteristic of the IR.

Augmented Kalman Filter

The disturbances affecting the robot dynamic, as evinced from the experiments, do not hold the assumption to be random Gaussian with zero mean and uncorrelated with respect to each other. Furthermore, and most importantly, they have a noticeable impact on the trajectory and appear to be deterministic, even though to compute their deterministic effect for every kind of situation would be complex and probably insufficient.

The Kalman model, therefore, can be greatly improved by also adding unknown disturbances into the system state, making the estimation more effective. Thus, the state matrices must be expanded with the addition of the unknown input disturbances, so that the state vector might contain three elements: the position error speed, the position error and the disturbances.

The new dynamic model is the following:

$$\dot{x_a} = \begin{bmatrix} A & E \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} x \\ d \end{Bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} L & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} wx \\ wd \end{Bmatrix} = \boldsymbol{A}_a x_a + \boldsymbol{B}_a u_a + \boldsymbol{L}_a w_a \tag{3.26}$$

where $\boldsymbol{A}_a$, $\boldsymbol{B}_a$ and $\boldsymbol{L}_a$ are the augmented state matrices. Additionally, x_a is the augmented state that takes into account also the unknown input disturbances, while w_a is the disturbance vector that comprehends also an artificial (fictitious) noise wn that it is introduced to allow the Kalman filter to modify the estimation of the disturbance parameter.

Therefore, the model state matrices derived from Eq. (3.16) have the following values:

$$\boldsymbol{A}_a = \begin{bmatrix} 0 & 1 & 0 \\ -w_0^2 & -2al & w_0^2 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{B}_a = \begin{bmatrix} 0 \\ w_0^2 \\ 0 \end{bmatrix}, \boldsymbol{L}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & w_0^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{C}_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \boldsymbol{D}_a = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The equations Kalman algorithm remains the same as before, however the $\boldsymbol{\Phi}$ and $\boldsymbol{\Gamma}$ matrices change, due to the change of the state matrices.

In particular, the $\boldsymbol{A}_a$ matrix is now singular, so the computation of $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ require the form:

$$\Gamma = \int_{t_0}^{t} e^{A(t-t_0)} B \, dt \tag{3.27}$$

$$\Lambda = \int_{t_0}^{t} e^{A(t-t_0)} L \, dt \tag{3.28}$$

This new approach allows being more precise in the estimation, providing also the unknown disturbances as output.

In Fig. 3.31, indeed, the estimated disturbances acting on the system are represented, coming as output of the state space model.

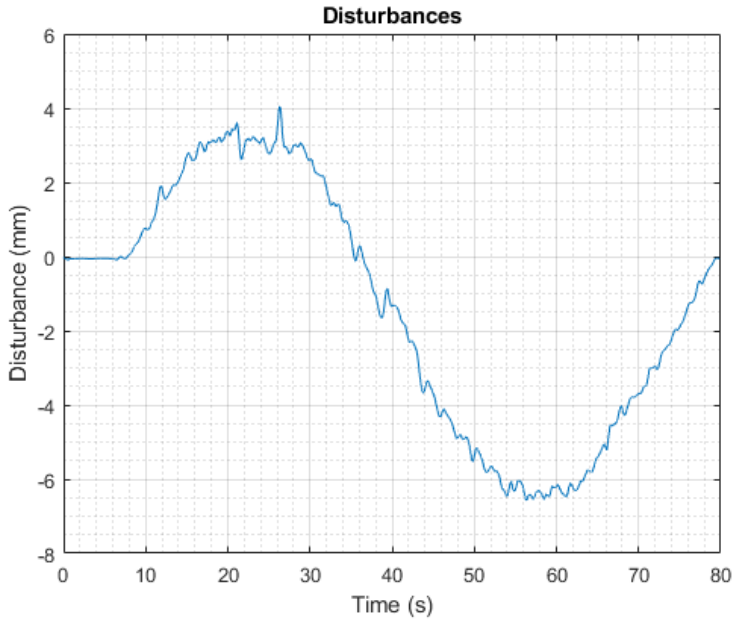
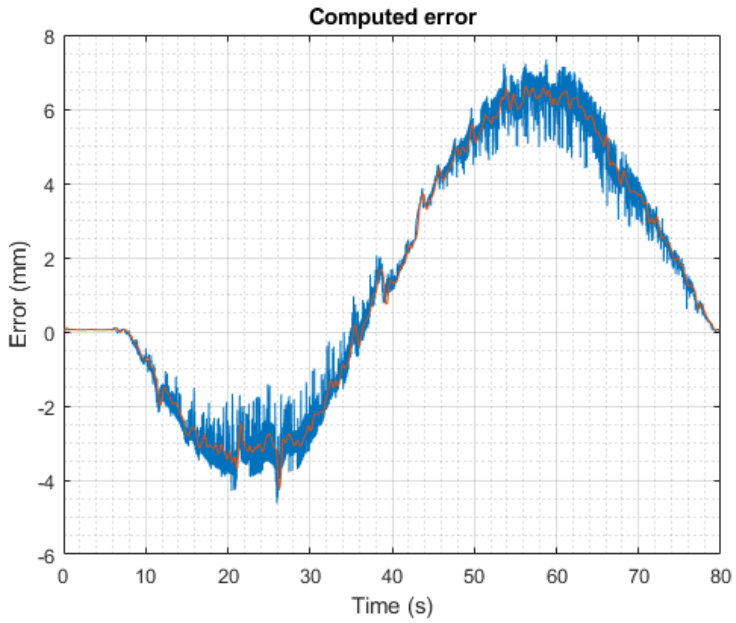


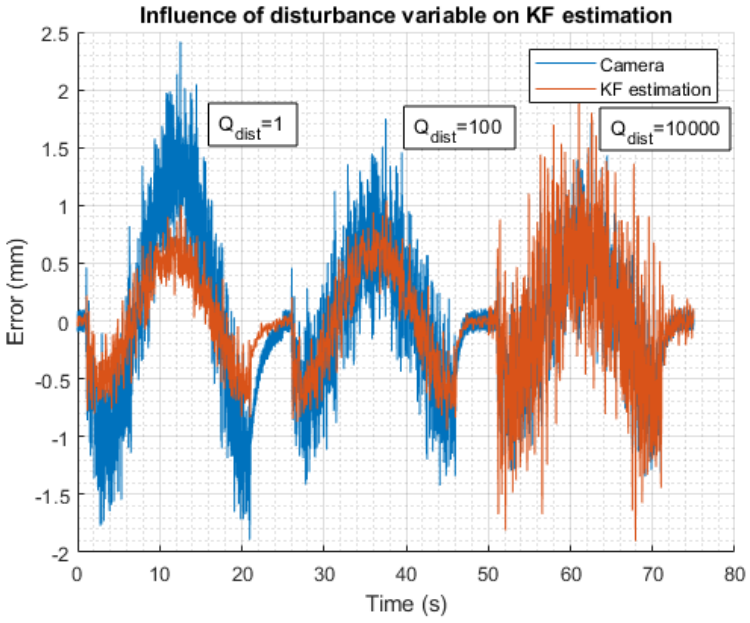
Figure 3.31: Computed disturbances during circular path

Then, for example, Fig. 3.32a shows the result of the disturbances, filtered through the AKF, during the execution of a circular path. As it can be noticed, the noise is greatly reduced and the estimated error is much smoother, but still quite accurate.

And, in conclusion, Fig. 3.32b shows the role that the disturbance variable plays with respect to its weight in the disturbance covariance matrix. The cost of this filtering action is, however, the addition of small delays in the system due to the heavier computation load.



(a) Filtered disturbances



(b) Simulated disturbance influence

Figure 3.32: System disturbances during circular path

Extended Kalman Filter

Eventually, also non-linearities can be added: the natural frequency, as well as the stiffness and damping, are a function of the IR robot configuration. However, the possible different configurations given by the six joints are too many, and difficult to model. Therefore, a practical solution would be to consider only the first three joints, which are responsible for the most remarkable movements in radial and vertical distance. Since these distances and the joint configurations are correlated, it was easier to model the dynamic parameters directly in the workspace coordinates, coming directly from the measurements, and on this basis implement an EKF.

The already developed model becomes then a function of the position p, in the form:

$$TF(p) = \frac{w_0^2(p)}{s^2 + 2\,\zeta(p)\,w_0(p)\,s + w_0^2(p)} \tag{3.29}$$

Because of this, the $\boldsymbol{A}_a$ and $\boldsymbol{B}_a$ matrices change as well with respect to the position, and thus, the $\boldsymbol{\Phi}$ and $\boldsymbol{\Gamma}$ matrices have to be recomputed at each sampling time, according to the change of the system dynamics, while the algorithm equations remain the same.

This additional computation time proved to be particularly detrimental for the controller performances, being cause of much greater delay, nearly twice the one of the other filter types. Therefore, the solution adopted in this work was to update the model matrices not at any step, but for example once every 100 iterations, so to maintain a fast dynamic in the response.

Some factors are however still neglected: indeed, the second order linear transfer function is still an approximation of the behavior of the system, which can have a more complex dynamics. Moreover, it is not possible to exactly determine the change in damping and stiffness of the system because it would require to test all the possible pose combinations. However, implementing a more complex model, taking into account also all the sixth order transfer function, would bring not so significant results for the aim of the control logic, which is limited more by other, more impacting, factors. The final results that it is possible to achieve with the implemented Kalman filter is a noticeable improvement with respect to the initial situation, being the measurements' noise greatly reduced.

3.5.3. Matlab/Simulink Model

Firstly, a model on the software Simulink from MathWorks, has been built, in order to test the model and equations.
The robot arm system, linearized around the target pose or trajectory, can be described as in Fig. 3.33. In reason of the linearization, the state variables of the system are the positional and velocity error, i.e., the offset from the nominal pose.
The scheme comprehends also the deterministic disturbance that act on the system, which depends on the system pose or on the particular motion involved, and which represent the robot accuracy error given a certain target. At the same time, on the system is applied an offset, the result of the sum of all the correcting inputs computed by the control logic. Eventually, also unknown random disturbances will be acting on the system.

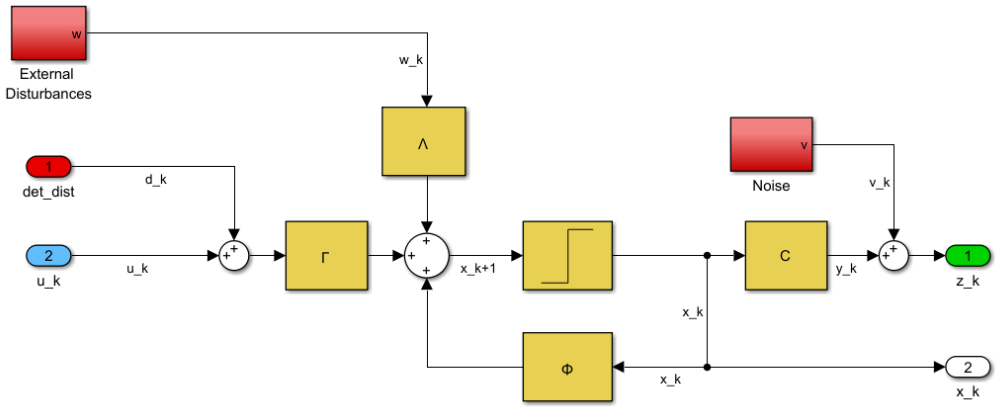


Figure 3.33: Block diagram of the modeled system

Then, the system is inserted in a feedback loop together with the Kalman filter and the control logic, see Fig. 3.35. According to the equations from 3.18 to 3.23, the filter takes as inputs the pose error measured by the system and the applied offset computed by the controller at the previous iteration.
Fig. 3.34 shows the Kalman filter subsystem, with the velocity filter and the prediction-correction cycle. Thus, the state for the actual step is estimated, and it is convoyed to the control logic to close the loop (see Fig. 3.35).

The controller adopted in the simulation is, at least at the beginning, a PID control, with optimized values taken from the static experiment configuration. The main aim of this simulation is, indeed, to tune the covariance matrices (as in Fig. 3.30) and verify the functioning of the algorithm. The optimization of the control parameters is yet tried, and considered as a starting point for the proper tuning of the controller on the IR, since, also according to experimental tests, it does not take in account robot limitations.

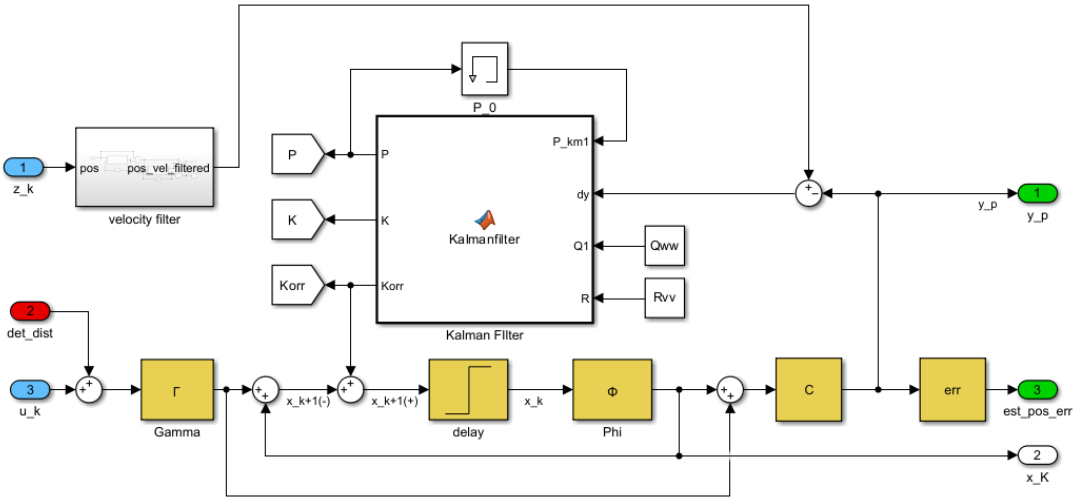


Figure 3.34: Kalman filter estimation block

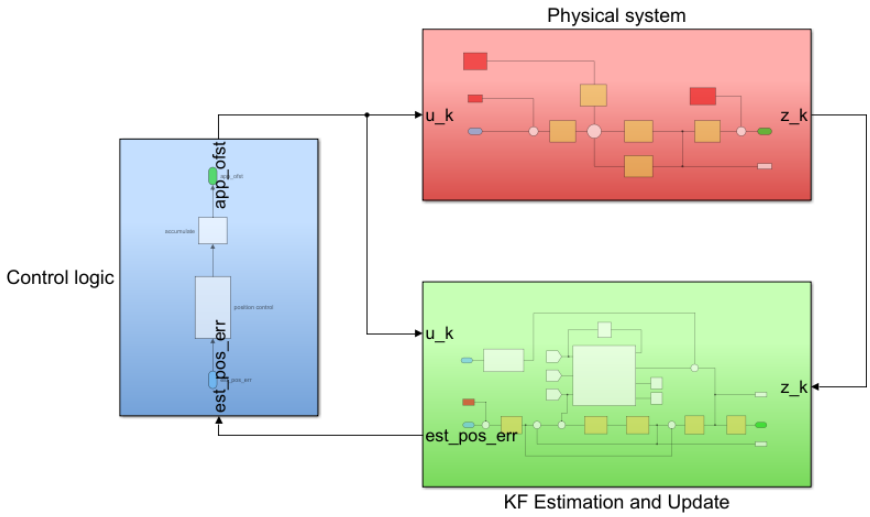


Figure 3.35: Schematic representation of filter loop and control logic

Thanks to this model, it was possible to replicate the behavior of the robot under control during the circular path. Indeed, Fig. 3.36a simulates the x-axis pose error due to the disturbance represented in Fig. 3.29, in the first half uncontrolled and in the second under control for comparison.

Therefore, it can be seen both the filtering effect of the Kalman filter (in this specific case the AKF) and the corrective action of the PID control. Furthermore, the control parameters have been optimized to reach the best performances under the simulation conditions. In particular, the optimal value of $K_P = 0.2$ was found, while for the integral and derivative control gains they were both found detrimental.

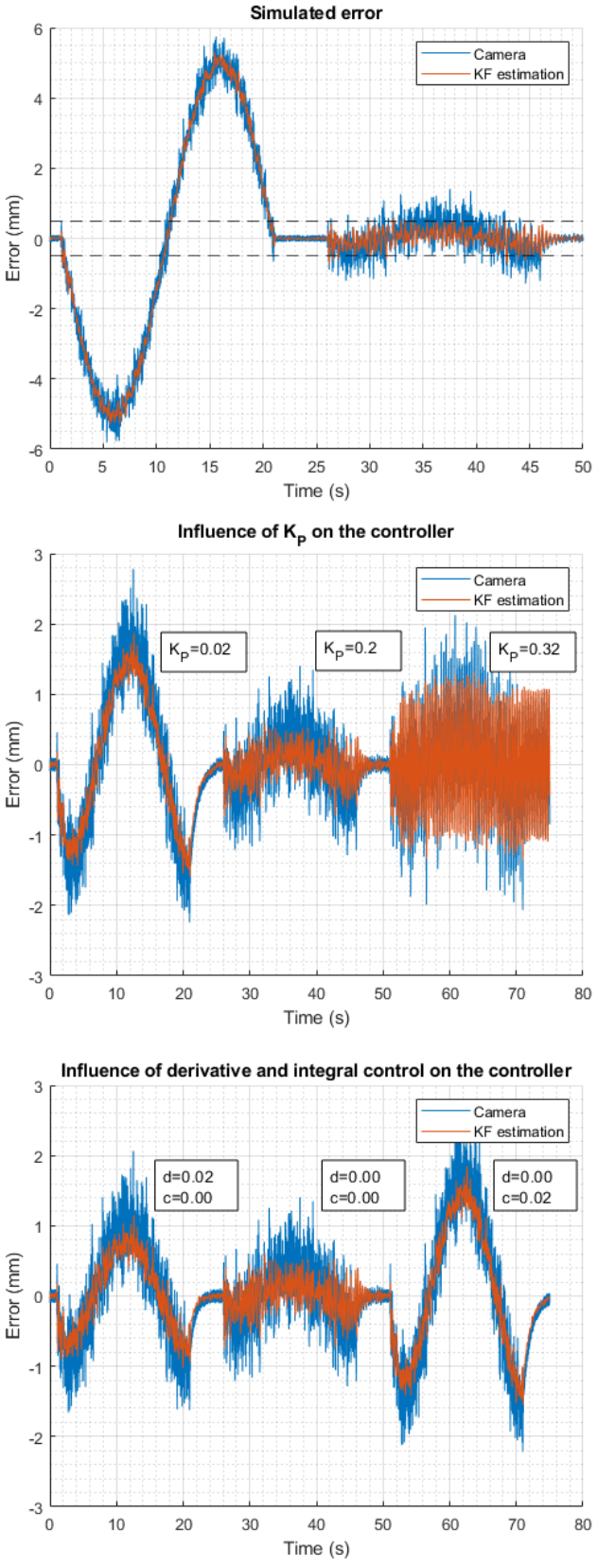


Figure 3.36: System disturbances during circular path

3.5.4. Python implementation

Velocity estimation

For what concerns the measured quantities, related to the second equation $y = Cx+Du+v$ of Eq. (3.13), the camera system is providing the position measurements and therefore also the error estimate. Additionally, in order to have a first approximation of the error velocity to feed into the Kalman filter, a velocity estimation function has been implemented.

This works by interpolating a series of measurements points with a polynomial function, and then deriving it at the current instant. In the Python implementation, written using the *NumPy* package, it can be split in three different processes:

1. Polynomial interpolation of the previous position errors: $\boldsymbol{k} = np.polyfit(\text{t, p, D})$
2. Derivation of the polynomial function: $\boldsymbol{kder} = np.polyder(\text{k})$
3. Evaluation of the derivative at the current time instant: $\boldsymbol{vel} = np.polyval(\text{kder, t})$

However, since often the measured error can have noise presence, in order to make the process more effective, a first low pass filter is implemented in order to filter the measured input to the velocity estimator. This avoids the need to use an excessive amount of points for the polynomial interpolation, while reaching the same overall result. Moreover, since the noise level can change over time and during the test, due to the change in light conditions that can affect the cameras, the low pass filter allows having a more or less consistent noise level input to feed to the velocity estimation.

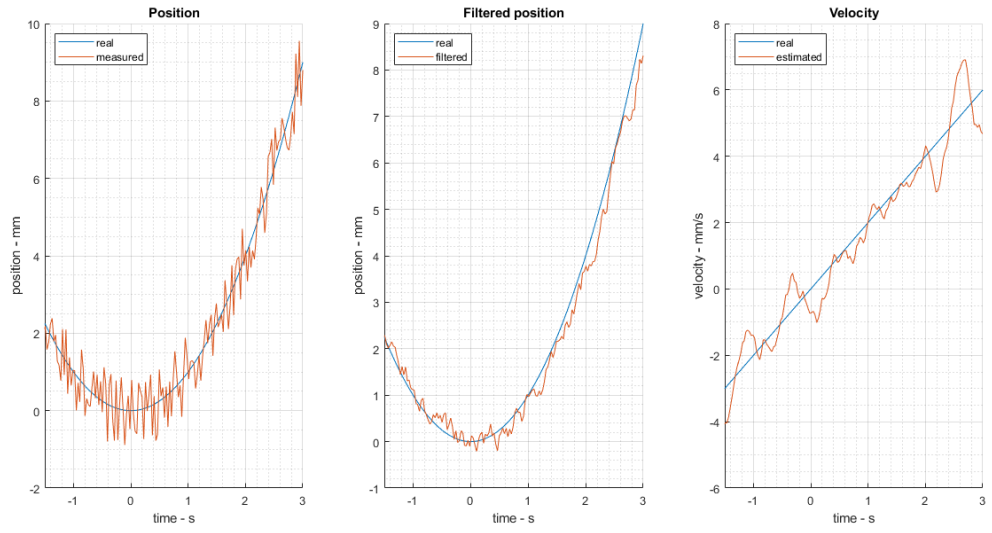


Figure 3.37: Velocity estimation process: 1. Position, 2. Filtered position, 3. Velocity

Kalman tools script

As previously mentioned, three type of Kalman filter have been implemented, and collected together in the same script *kalman_ tools.py* in form of callable functions. For each of them, the Bucy, Augmented, and Extended KF, the main functions are *set* and *run*, the first one to initialize the filter with state space matrices computation and discretization, and the second to update at each iteration the filter gain K, the covariance estimate P, and the state estimate. As a result, also the estimated pose error is available for the controller.

The case of the EKF is slightly more complex, because once the various pose frequency and damping ratio parameters are collected and interpolated, to create a fuzzy net depending on the end effector position, the state space matrices are to be updated each time, occurring in higher computational costs and delays. For this reason, after comparing the delays of the three Kalman filters, it has been decided to slower the updating frequency of the EKF, also considering the speed at which the IR is moving during the test paths.

Algorithm 3.2 Kalman tools script algorithm

> **def** function *set*:

1. Compute state continuous matrices from frequency and damping ratio
2. Compute state discrete matrices with sample time T
3. Set error and disturbance covariance matrices
4. Initialize covariance and state estimate

return Discrete matrices and estimate matrices

> **def** function *run*:

1. Update filter gain and covariance estimate
2. Compute new state estimate and pose error

return State estimate, pose error, velocity error, and disturbances

> **def** function *set_ EKF*:

1. Initialize covariance and state estimate
2. Get EKF parameters to form frequency and damping ratio looking table

return Frequency and damping ratio looking table and estimate matrices

> **def** function *run_ EKF*:

1. **if** n iterations are passed **then**

   Get pose dependent frequency and damping ratio (algorithm 3.1)

   Compute state continuous matrices from frequency and damping ratio

   Compute state discrete matrices with sample time T
2. **else**

   Get old state discrete matrices from queue
3. **end if**
4. Update filter gain and covariance estimate
5. Compute new state estimate and pose error

return State estimate, pose error, velocity error, and disturbances

3.5.5. Kalman Filter Performance

In this section, a comparison between BKF, AKF, and EKF is presented, together with the tuning of the disturbance covariance matrices. In the following tuning process, and in all the experiments carried on, the noise and disturbance covariance matrices R and Q have been treated as diagonal matrices, thus only the traces are considered from now on.

For the first iteration of the tuning process, after having selected $R = \begin{bmatrix} 0.1 & 0 \\ 0 & 10 \end{bmatrix}$ with $Tr(R) = [0.1, 10]$, the Q values are changed, with progressive increment of order of magnitude. For assessing the best performing, the average and maximum error, and the standard deviation, are estimated over triplets of measurements along the circular dynamic path, travelled at 120 mm/s.

Fig. 3.38 shows the performance of the BKF. Not much difference is visible between the trials, and the filtering effect is almost negligible, with exception for the first ones with the lowest ratio on the positional element ($\boldsymbol{Q}(1,1) = 1$), which causes phase shifting on the estimated error and, thus, worse performance.

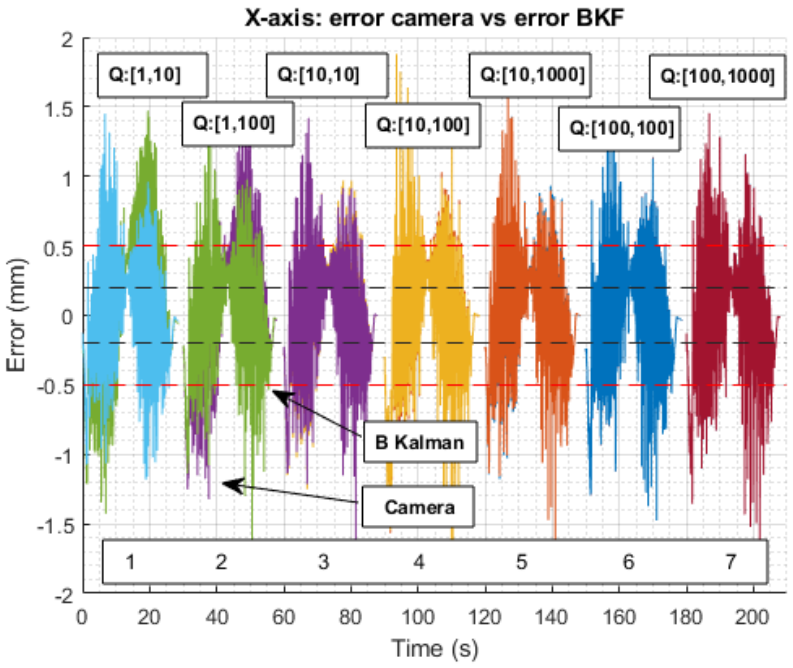


Figure 3.38: First iteration of tuning of Q matrix

Table 3.5: Errors for first iteration of Q tuning for BKF

Error values for Q tuning (mm)							
	1	2	3	4	5	6	7
Mean	0.62	0.59	0.48	0.48	0.49	0.51	0.50
Max	1.64	1.78	1.58	1.90	1.71	1.71	1.91
SD	0.31	0.30	0.23	0.29	0.26	0.26	0.26

Therefore, the three best performing have $\boldsymbol{Q}(1,1) = 10$, and further measurements are taken to better refine the tuned parameters of each of the Kalman filter's typology.

In the case of the BKF, in particular, only one additional variable needs to be tuned, and the resulting measures (of which only one of the triplet is plotted for clarity) are shown in Fig. 3.39. According to this experiment, the values of 10 and 100 for the trace of $\boldsymbol{Q}$ have been selected and therefore used in the experiments of chapter 4.
Indeed, the desirable effects of the Kalman filter are mainly improving the mean error, maintaining at the same time the standard deviation as small as possible. The maximum error, though being a key parameter, is in this case greatly influences by noise and computational errors, therefore its importance is secondary compared to the other values.
In the following tables then, showing the error values, the best configuration is highlighted in green, whereas the second best one in yellow (as in Table 3.6).

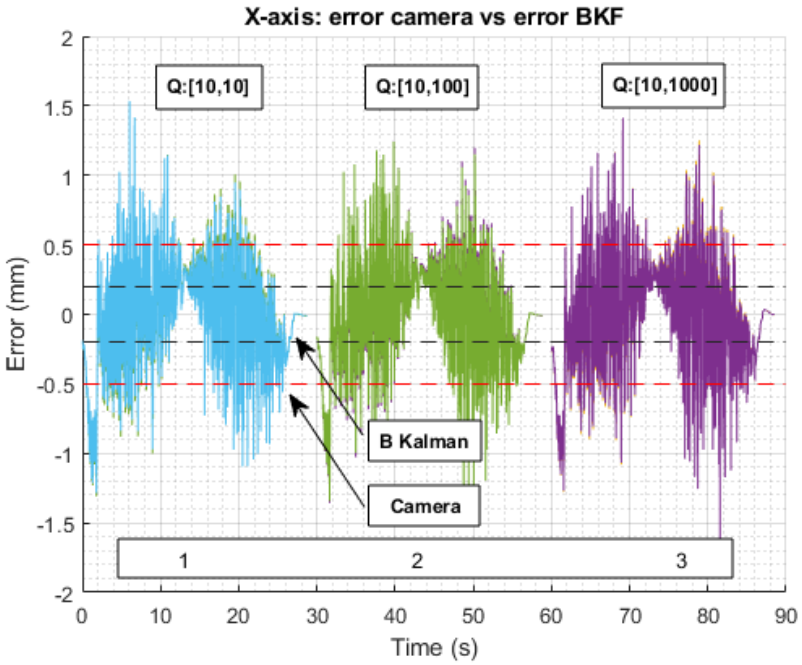


Figure 3.39: Tuning of BKF

Table 3.6: Errors for tuning of BKF

Error values for Q tuning (mm)			
	1	2	3
Mean	0.50	0.49	0.50
Max	1.63	1.69	1.95
SD	0.25	0.25	0.25

The same tuning process is then applied on the other two filters, the AKF and EKF, which additionally have the disturbance state variable and, thus, a 3x3 Q matrix.
Therefore, additional configurations with varying values of $\boldsymbol{Q}(3,3)$ are tested, while $\boldsymbol{Q}(1,1)$ is kept at the already assessed optimal value of 10.
The resulting error values are listed in Table 3.7 and Table 3.8.

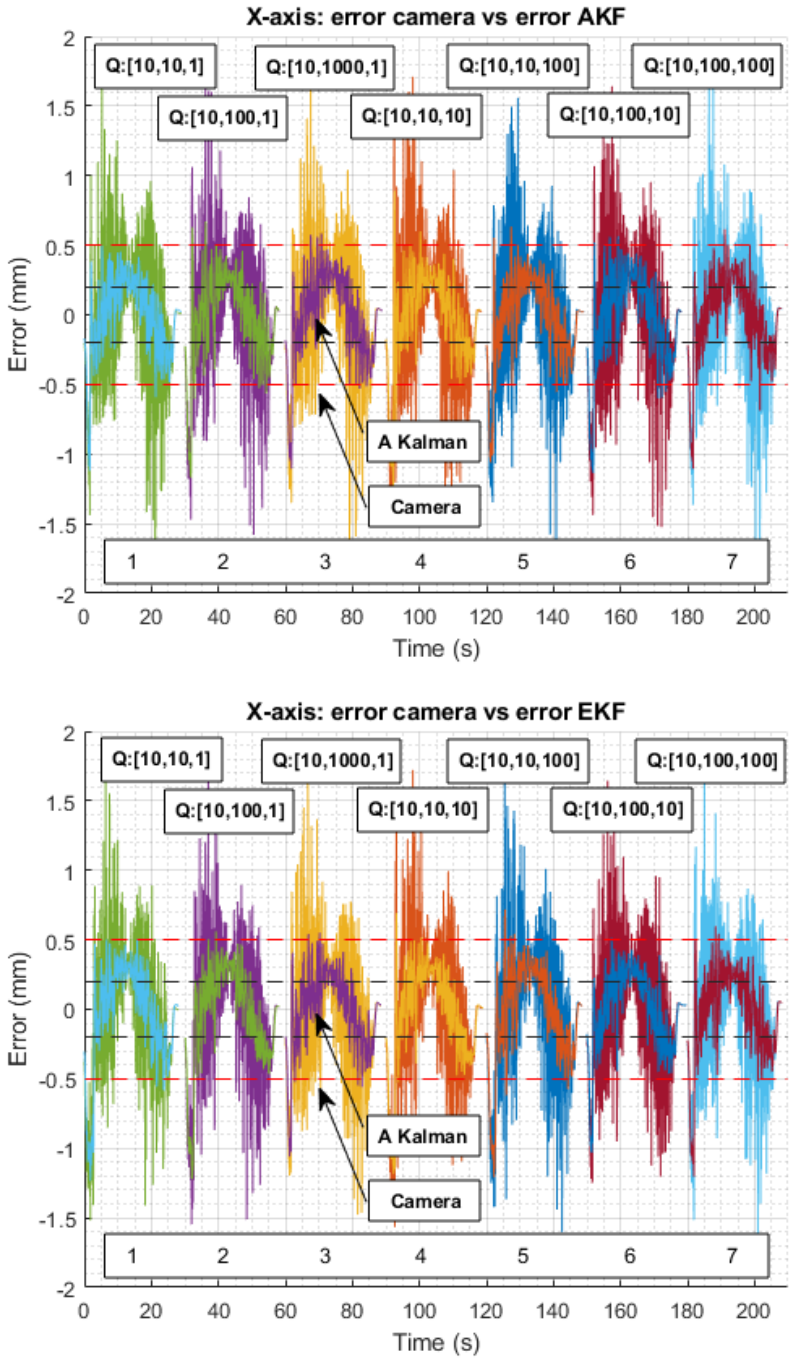


Figure 3.40: Tuning AKF e EKF

Table 3.7: Errors for tuning of AKF

Error values for Q tuning (mm)							
	1	2	3	4	5	6	7
Mean	0.54	0.48	0.52	0.51	0.50	0.52	0.50
Max	1.88	1.78	1.74	2.02	1.90	1.67	1.86
SD	0.26	0.29	0.27	0.27	0.26	0.27	0.29

Table 3.8: Errors for tuning of EKF

Error values for Q tuning							
	1	2	3	4	5	6	7
Mean	0.53	0.52	0.52	0.53	0.53	0.51	0.51
Max	2.03	1.93	1.83	1.89	2.00	1.89	1.77
SD	0.30	0.29	0.29	0.30	0.29	0.29	0.27

In conclusion, Table 3.9 shows a comparison of the best results obtained after the tuning of the disturbance covariance matrix. Though these are not definitive results, since the optimization on the controller parameters is missing at this stage, they give a glimpse of the behavior of the filtering effect on the accuracy.
As it is evident, even though the BKF, with the selected parameters, almost does not filter, differently to AKF and EKF, still the resulting average error is of the same intensity (around 0.5 mm) and also the maximum value and standard deviation are comparable.

This is not clearly explainable, since in the face of big changes in filtering effect, the changes in the measured error are too low, almost undetectable considering the repetitive accuracy of the experiment. Nonetheless, some considerations can be made, exploiting the acquiring data, and in particular some possible factors can be excluded from the list of the possible phenomena affecting the correction performance.

Table 3.9: Comparison of errors for tuning KFs

Best filters' error values (mm)			
	BKF	AKF	EKF
Mean	0.49	0.52	0.51
Max	1.69	1.67	1.77
SD	0.25	0.27	0.27

For instance, if at the beginning the lagging in the sampling time, due to the additional computational cost in the EKF, was suspected to be a cause of the detrimental performances of the latter, this issue does not appear anymore once that the update of the transfer function parameters has been rescheduled at a lower frequency. In fact, with the current code the sampling times of the KFs are equivalent and steady at around 25 ms, as shown in Fig. 3.41.

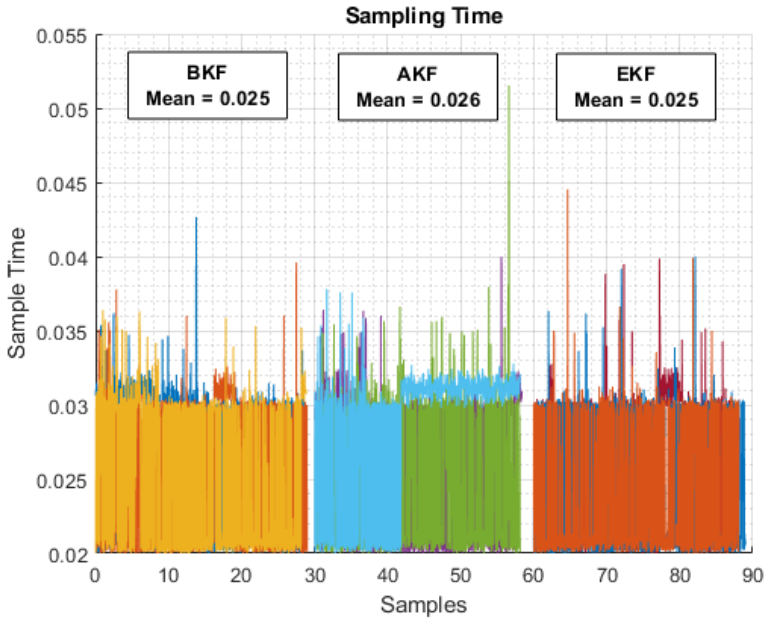
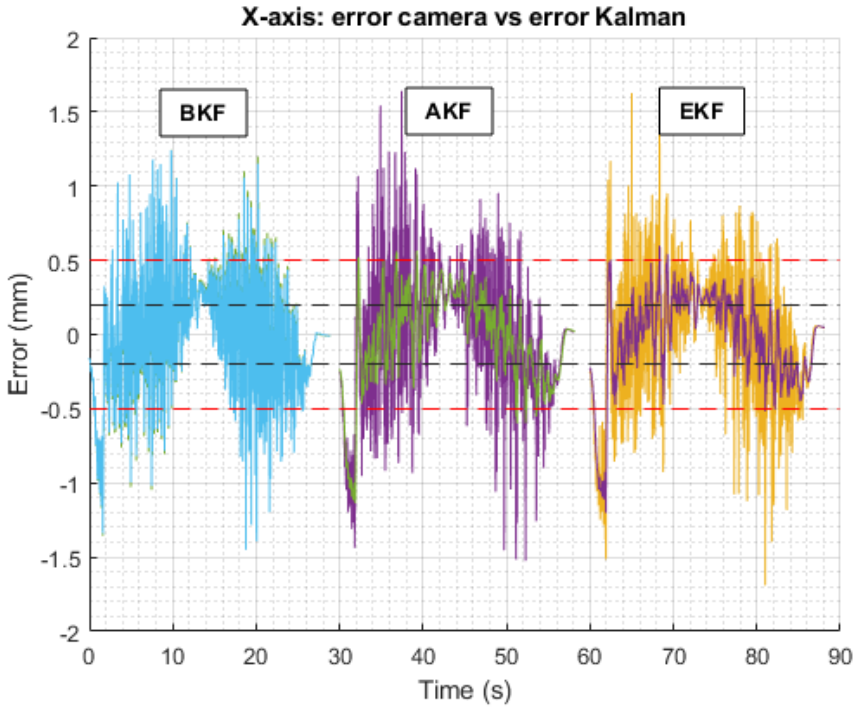


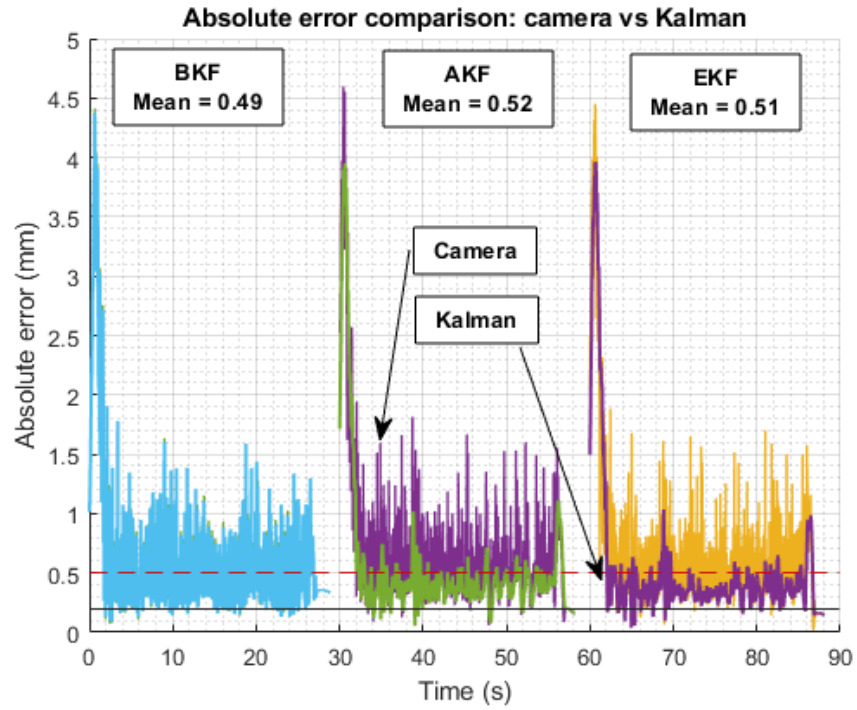
Figure 3.41: Sampling time

Different, instead, is the case of the camera noise. Though the filtered values enter the control, the measurements taken as an output from the camera are still affected by noise and possible disturbances.
It would be inexact to refer to the filtered error as the output of the experiment, since it could bring to wrong assessments and consideration about the real behavior of the robot. At the same time, however, the cameras are suspected of poor accuracy in the high precision range required by the system.

As presented in Fig. 3.42, the three different KFs lead to approximately the same mean error, against a strong reduction of the estimated one. For this reason, when discussing the results in chapter 4, also the measurements taken by the laser tracker are considered. The laser tracker measurements, however, are not deprived of disturbances, and, in particular, a direct comparison over time is not easy to perform, having the two systems different sampling rate and the internal clock do not match.



(a) X-axis error



(b) Absolute error

Figure 3.42: Kalmans' error comparison

3.6. Dynamic Control

3.6.1. Linear control

After having successfully tested the static control and implemented the Kalman filter for the camera measurements, the dynamic control is developed, which aims at minimizing the pose error of the robot along a given trajectory in space.
The controller performance can be computed as mean or median error along the whole path, as well as the maximum absolute error and the standard deviation.

In order to conduct the experiments, a Karel program has been written, giving instruction to the robot arm about the trajectory to execute. In particular, the same path is repeated twice: the first time without control, which serves as a benchmark, while the second one with the control enabled.
The controller as been developed on a circular trajectory in the x-y plane, with slope along the z-axis, so to test the response for the different Cartesian coordinates. Along the path, instead, the orientation had to be kept constant.

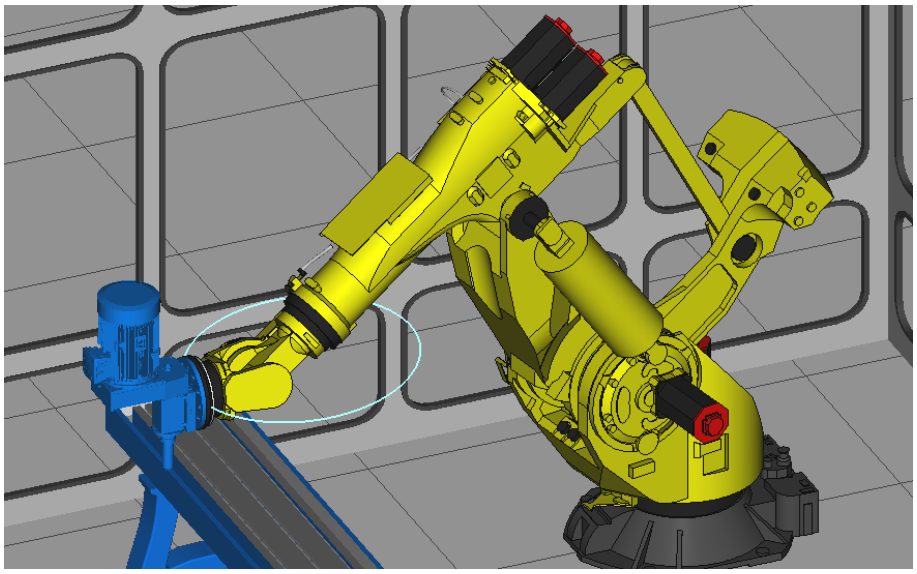


Figure 3.43: Testing trajectory

Table 3.10: Inclined circular path points coordinates

Inclined circular path						
P	1	2	2	4	5	5
X (mm)	450	500	0	-500	0	500
Y (mm)	0	0	-500	0	500	0
Z (mm)	0	100	150	200	150	100

Another variable considered for the testing is the working speed of the robot. In particular, the tests have been performed with two working speed: 60 mm/s, and the double, 120 mm/s. These speed pairs will be kept fixed for all the performed tests.

Firstly, tests have been carried out with a proportional control, to better understand the behavior of the robot and the differences with the static case. Indeed, now the control action and the robot dynamic response around the target pose is influenced by the superimposed motion of the trajectory.

Initial tests have been done with same values of proportional gains for all the channels of the Cartesian coordinates, with gradually increasing values of control gains. In order to compare the results, the unfiltered cameras' data have been used, so to avoid any wrong conclusion due to the application of the Kalman filter. Moreover, the first and last 10 percent of the trajectory have been neglected.
To visualize the error distribution, the Matlab boxplot function has been used, which displays the median value, the 25th percentile, the 75th percentile, maximum and minimum and outliers of a certain set of values.

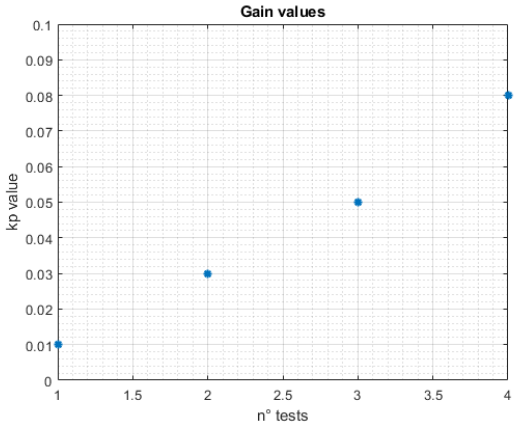


Figure 3.44: Gains used for all channels

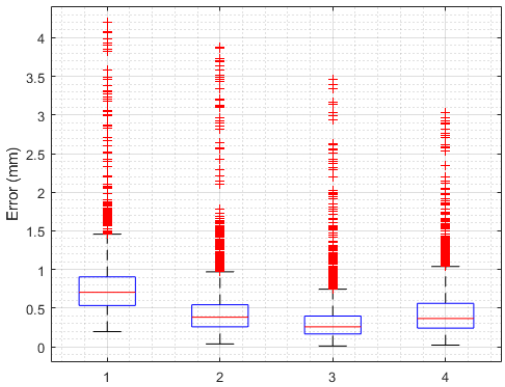


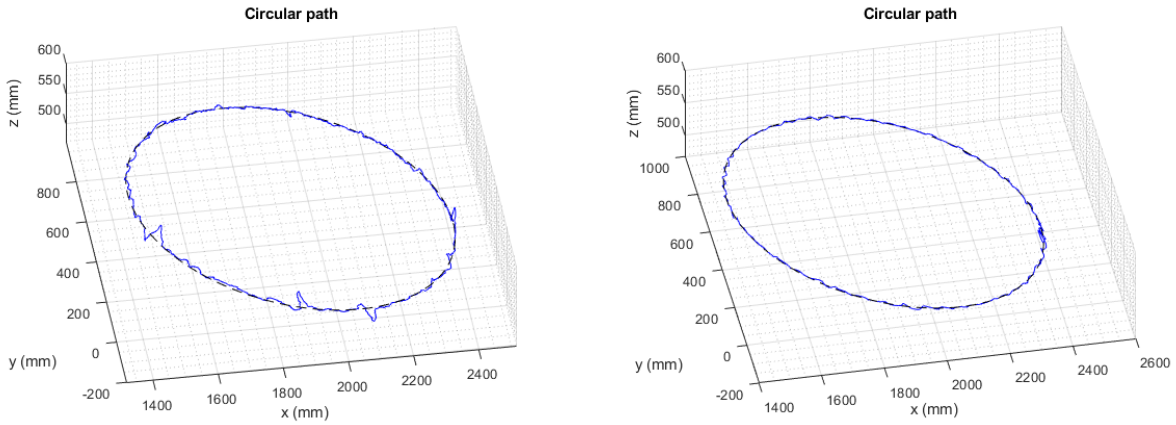
Figure 3.45: Data error distribution

Increasing the values of gains reduces the mean error as well as the maximum one, however the result is not very smooth.
In particular, there are a couple of sections where the error sharply increases: they correspond to the sectors where the wrist of the robotic arm has to change direction of motion, which can lead to unwanted excessive movements.

To avoid these phenomena, it has been found that the control gains on the three angular positions has to be kept small. Indeed, too high values of control gains would result in an excessive motion of the wrist, resulting in a loss of accuracy.

Moreover, the orientation error at the tool end effector is lower and more limited if compared to the linear error, which can reach quite high values, so overall it is beneficial to use reduced control on the orientation for improving the results of the linear axis. Additionally, also the control value on the z-axis should be lowered, to avoid unwanted oscillations.

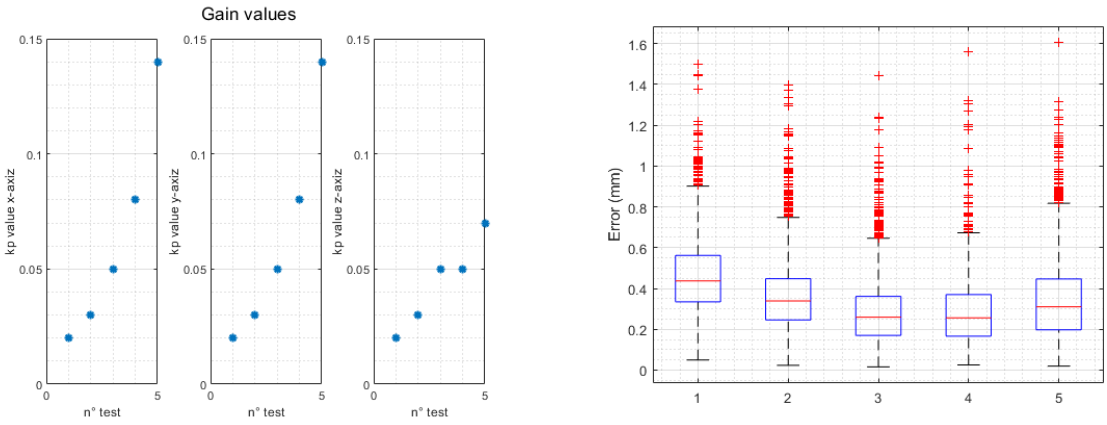
Here is the result and comparison between the two control strategies (in both cases the error is magnified so to better see the results):



(a) Same control gain (b) Smaller control gain on Euler angles

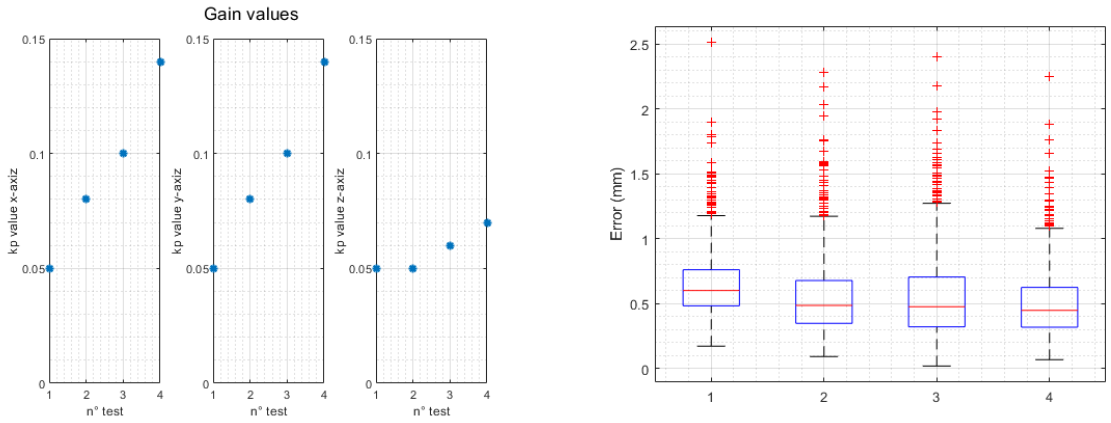
Figure 3.46: Circular path comparison between different control strategies

The test performed with lower values of control on the orientation show the same trend seen in the previous tests. At a certain value of gain, the error reaches a minimum before starting to increase again.
At higher speed, the overall error is larger, as expected, and the optimal control gains that allow to minimize the error are bigger than for lower speeds.



(a) Gains used for control (b) Data error distribution

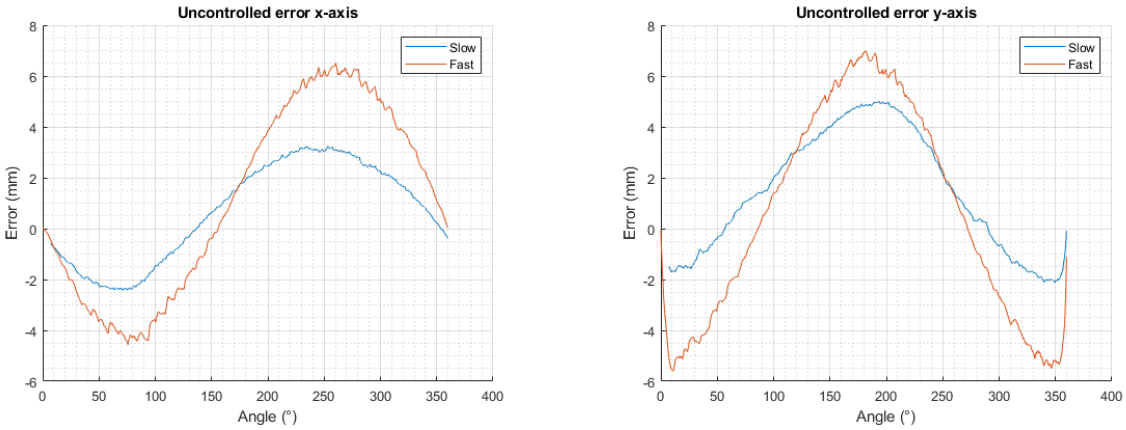
Figure 3.47: Linear axis control at low execution speed



(a) Gains used for control (b) Data error distribution

Figure 3.48: Linear axis control at high execution speed

An interesting aspect is the error shape, which can be noticed to depend on the particular path used, while its magnitude depends on the execution speed.



(a) Error on x-axis (b) Error on y-axis

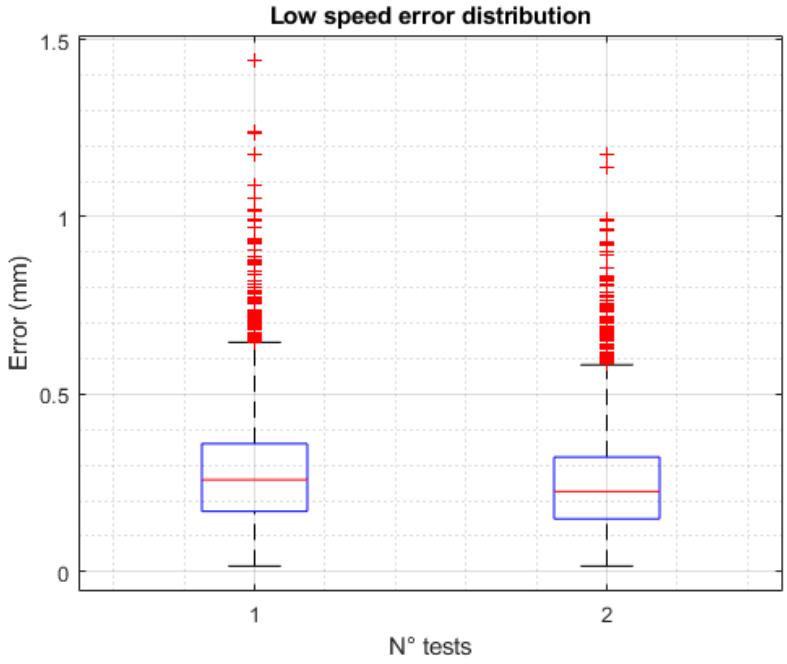
Figure 3.49: Uncontrolled error magnitude with respect to the execution speed

3.6.2. Fuzzy logic for dynamic control

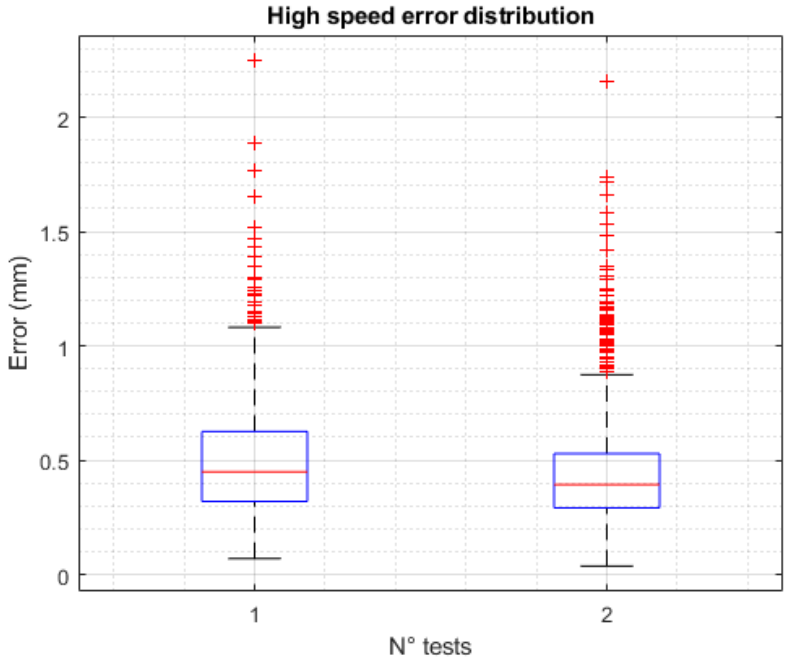
For instance, the fuzzy logic control, already developed and tested for the static case, can be applied to the dynamic path compensation. As already seen with the static control, the fuzzy logic could allow to better control the dynamic behavior of the robot, using stronger gains with higher errors and, at the same time, approach the reference pose without overshoot or excessive strength at lower error magnitude. It also allows more variety in tuning, being able to change more parameters.

The control has therefore been tested with the same working conditions as for the previous cases: same circular path, and the same pair of working speeds.

At both low and high speed, the fuzzy logic shows an improvement in terms of mean and maximum error, improving the overall response behavior.



(a) Low speed tests



(b) High speed tests

Figure 3.50: Error distributions at different speeds for proportional fuzzy logic

3.6.3. Feedforward control

Since the robot arm has a very high repeatability, once the path and the speed has been fixed, the error profile along the trajectory is relatively constant. This characteristic can be exploited by implementing a feedforward control action that can compensate the predictable part of the error, while the feedback control is kept for the unpredictable movements and oscillations.

This method can be used for some cases where very high precision may be needed, but loses the general characteristic of the control, since the feedforward action is specific to each path. However, it is interesting to see the potential improvements on the final accuracy.
To do so, it is first needed to compute the error predictable component, extrapolated by performing the dynamic path without control. This has to be done for all the wanted Cartesian coordinates, so to create a feedforward input for each of the channels interested.

In the considered path, the feedforward action has been implemented for the x and y-axis, since they are the ones with bigger error magnitude and an easily predictable shape. For this purpose, the z axis, and the orientation error has not been considered, since on the testing trajectory they are quite limited and can reach already good results with just the feedback loop.

The feedforward action has been implemented by interpolating the predictable error components with a time dependent polynomial shape.
Thus, while the trajectory is performed, the feedforward input is computed by evaluating the polynomial shape at the current instant and applying the necessary correction. Eventually, the feedforward control has been tested on the low speed trajectory, with a fuzzy feedback control on the residual error, resulting in a further decrease of the error magnitude (see Fig. 3.54).

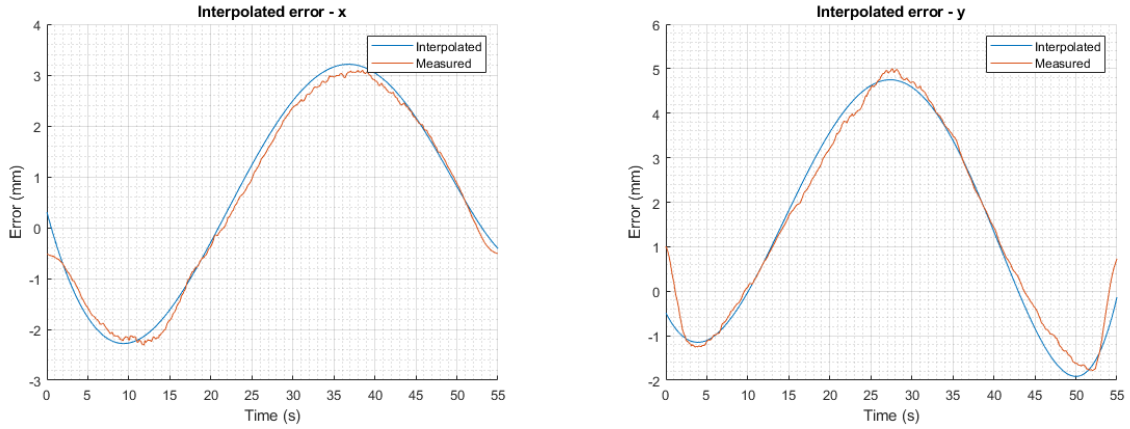


Figure 3.51: Interpolated error for x and y axes

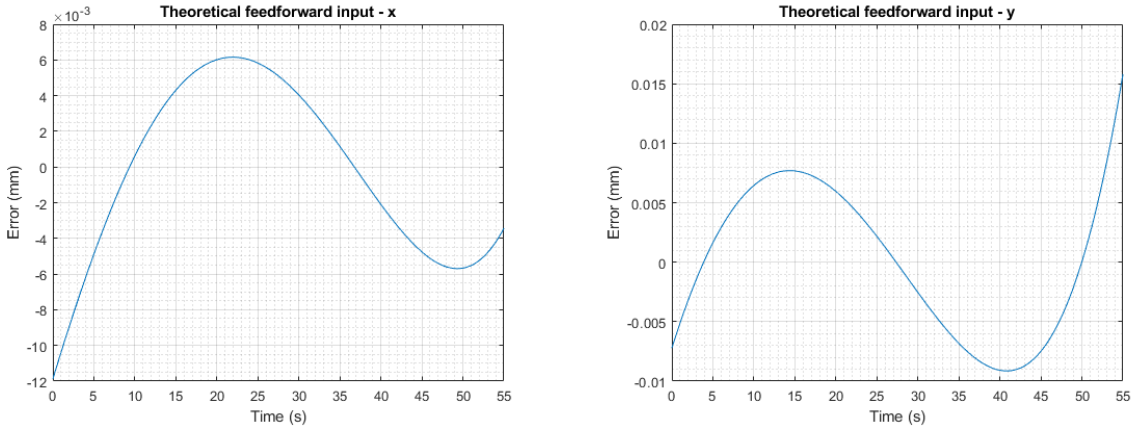


Figure 3.52: Theoretical feedforward input for x and y axes

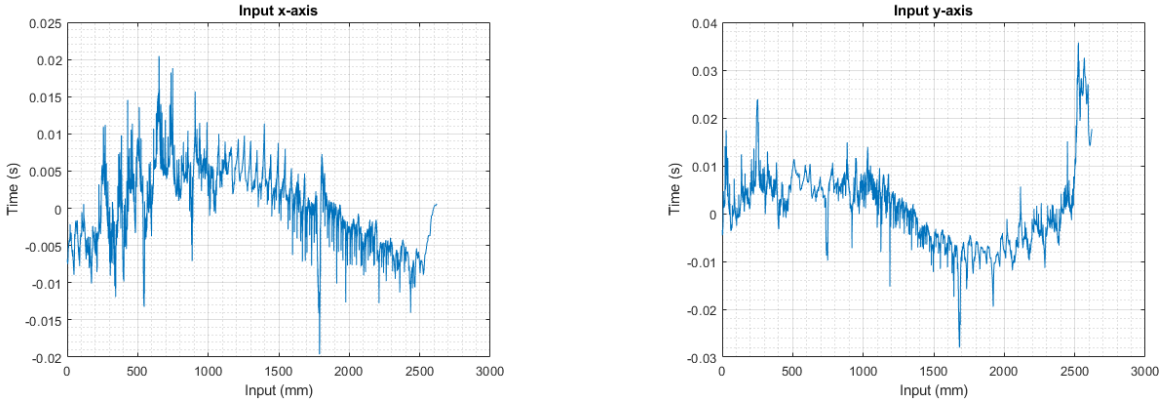


Figure 3.53: Input sent to the x and y axes

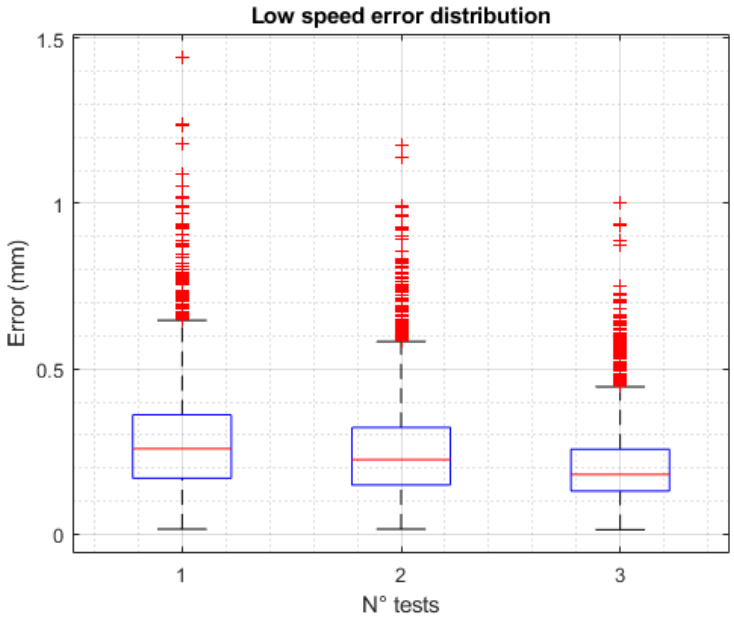


Figure 3.54: Error distribution for low speed tests: proportional, fuzzy, feedforward

4 | Presentation of the Results

4.1. Outputs

The experimental tests showed an improvement in the working precision of the IR both for the static and dynamic case.
For what concerns the static case, the error has been reduced noticeably, being kept within 0.05 mm from the target pose. The compensation speed, as well, has been a performance factor considered, with the fuzzy logic that allowed to keep a fast response without compromising the stability. The static control has been tested and analyzed in different poses.
For the dynamic case, instead, the control action has allowed to reduce the error along the path from several millimeters to a few tenths of millimeters, with different working speeds. Also, in this case, different path trajectories have been inspected and analyzed.

4.1.1. Static control

The static case has been tested in a neutral position, far from the limits of the robot capabilities, i.e., at mid-height and without the arm fully extended or contracted. The robot precision can be easily improved with a proportional control on the error magnitude, allowing to keep the position error of the tool end effector below 0.05 mm.
Hence, the performance difference between the different controller can be assessed mainly in terms of control speed, other than the overall precision.

With fuzzy logic, it has been possible to keep a fast response for big error magnitudes, while having a precise and controlled approach to the target pose.
At the same time, with static control, it is possible to have almost no error at steady state, no matter how big the initial error is, due to the integral action provided by the DPM function, which allows convergence of the position after a certain amount of time.

The typical system controlled response to an input disturbance is shown in figure 4.2. As the tool end effector is perturbed from the target position, the control gains get bigger and the error is quickly reduced.

For instance, the control logic is able to compensate circa 60 percent of the error in 0.5 seconds, while the final target is reached with lower control gains, allowing a more controller response and a final error well below 0.05 mm.
The transient response, instead, can vary depending on the considered coordinate or pose. The z-axis, especially, has a bigger tendency to oscillate than the x and y axes, while the orientation usually have a low error to begin with.

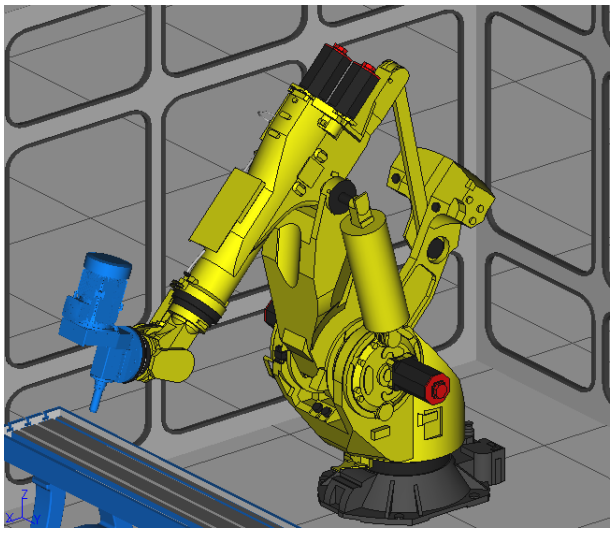


Figure 4.1: Pose used for the static test; Joints (deg) = [1.0, 12.0, -46.0, -1.0, 65.0, 1.0]

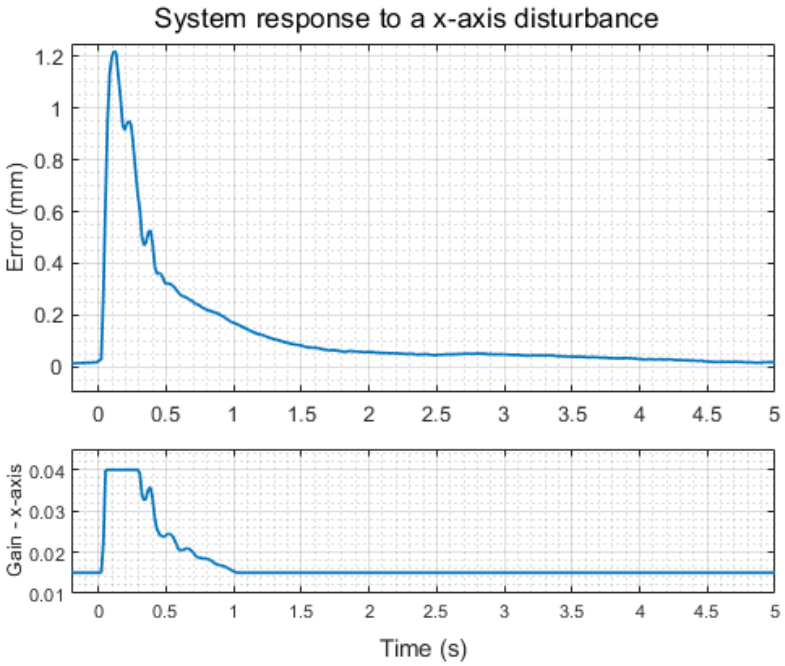
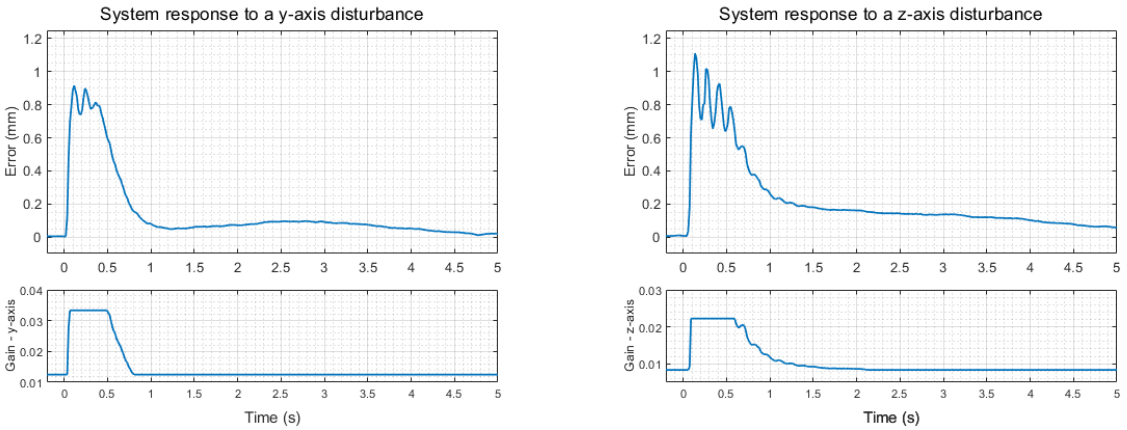


Figure 4.2: System response to an x-axis disturbance



(a) y axis disturbance response

(b) z axis disturbance response

Figure 4.3: System response to a y and z axes disturbances

Static multi-pose

Moreover, the static control has been also performed in several positions, different to the neutral one. As already mentioned in chapter 3.4, the static fuzzy logic has been tested in some of the 14 positions used in the parameter identification phase, and, additionally, also random positions within the working space of the robot have been considered.

The following Fig. 4.3 shows the spatial error compensation of disturbances sent along each linear axis, evaluated in the six random positions selected. The overall response in the different positions is quite similar, with a stable behavior and fast response at high magnitudes.
The results confirm the ability of the developed fuzzy control logic to be used in different poses within the robot working space, while maintaining similar performances for what concerns precision and compensation speed along the different axis.

Table 4.1: Static multi-pose testing points

Pos.	Joint (deg)
1	[50.0, 28.0, 30.0, 67.0, -56.0, -53.0]
2	[35.0, 25.0, 35.0, 51.0, -48.0, -40.0]
3	[18.0, 10.0, 19.0, 45.0, -25.0, -41.0]
4	[8.0, -3.0, 41.0, 12.0, -42.0, -9.0]
5	[3.0, -5.0, 31.0, 6.0, -31.0, -5.0]
6	[4.0, -10.0, 8.0, 29.0, -9.0, -29.0]

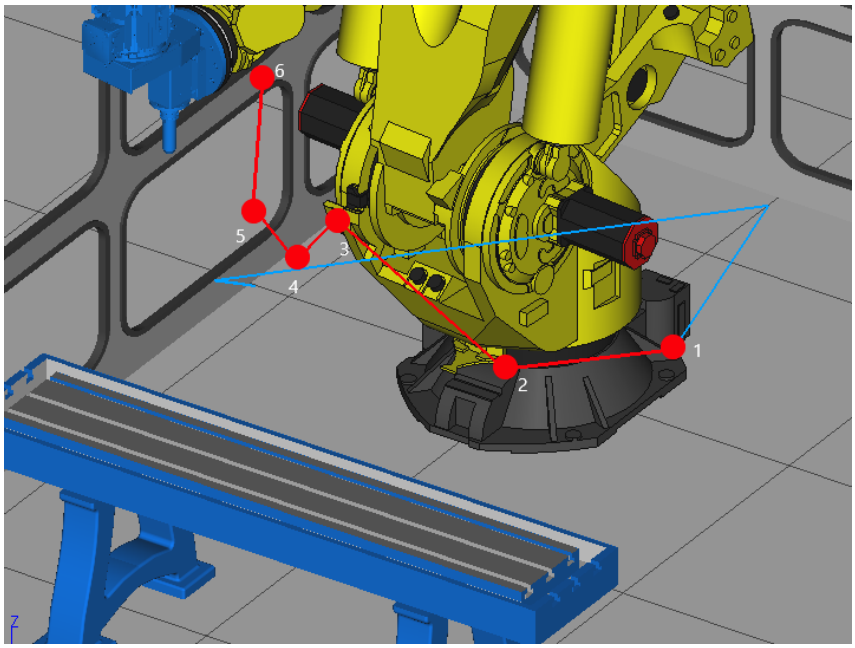
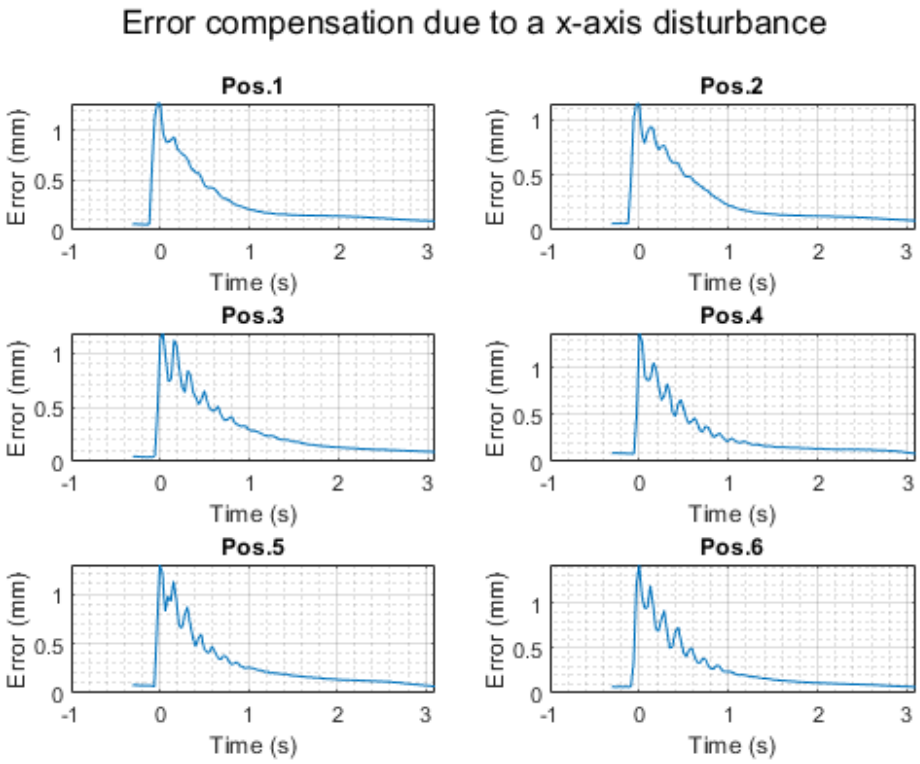
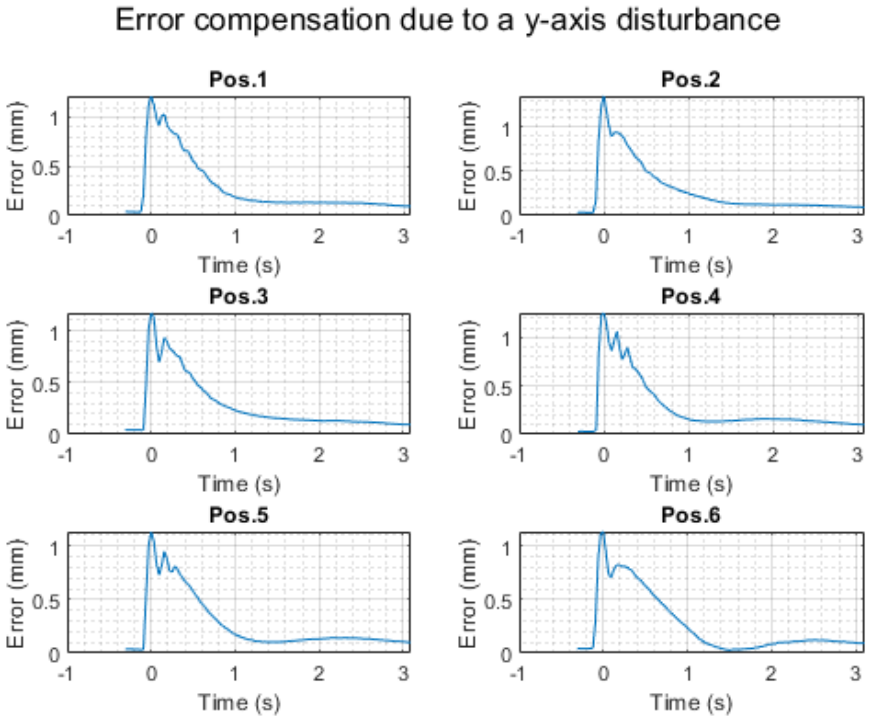


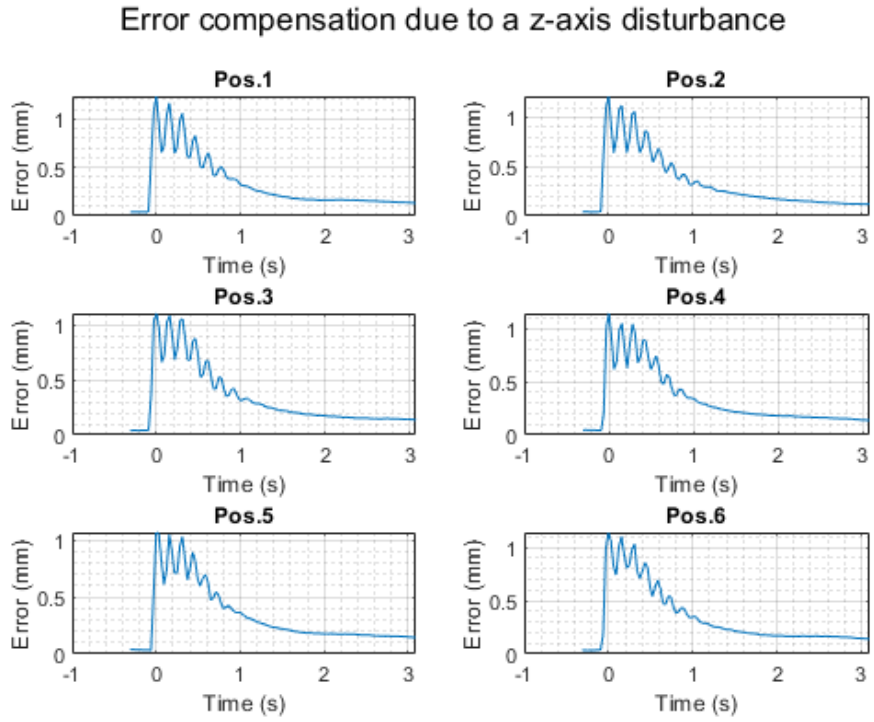
Figure 4.4: Points used for testing the static control



(a) x-axis disturbance



(b) y-axis disturbance



(c) z-axis disturbance

Figure 4.3: System response to each axis disturbance

4.1.2. Dynamic control

Differently from the static case, the dynamic one required more optimization, and the evaluation of the performances along different working conditions and configurations. Therefore, as explained in section 3.6, it has been firstly tested along a circular trajectory and at different speeds.

The performance parameter is the absolute position error along the whole trajectory, which has to be minimized. As in the static case, firstly the linear control with proportional gain has been tested, followed by the fuzzy logic control.
For the comparison, only the data coming directly from the camera measurements have been used, not considering the filtered values which may not be trusted as a real representation of the behavior.

Low speed control

For the low speed trajectory, three types of control strategies have been tested: linear proportional control on the error, fuzzy logic control and feedforward control with fuzzy logic feedback loop. Additionally, also the type of Kalman filter has been varied, to compare the performances of the controllers on different filtered data.

With regard to the control strategies, the best results achieved are the following, also presented in Table 4.2.

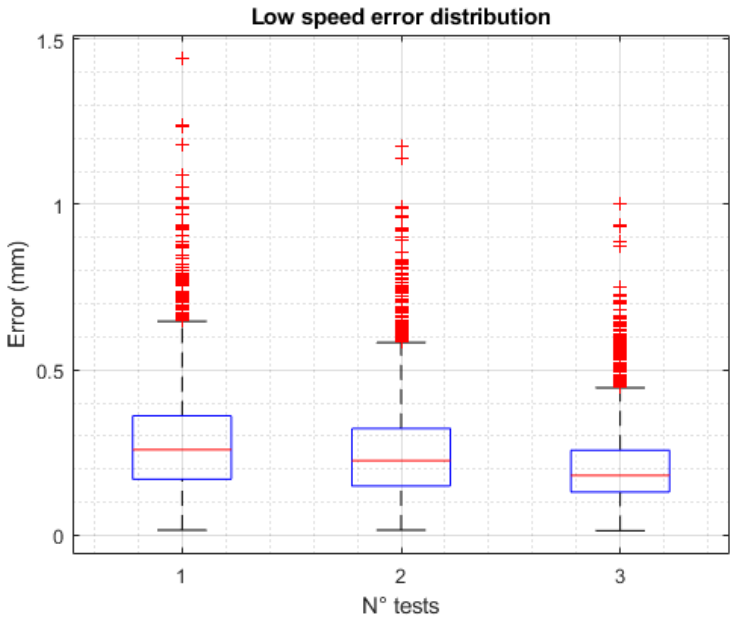


Figure 4.4: Dynamic control low speed error distribution: linear, fuzzy and with feedforward action

The fuzzy logic allows reducing the median error by additional 0.03 mm, while the feedforward control can further reduce it by 0.07 mm if compared to the linear control case. Additionally, also the maximum error values have been reduced, with the feedforward control showing the best results, followed by the fuzzy logic control.

Table 4.2: Dynamic path control at low speed

Error (mm)	1. Proportional control	2. Fuzzy logic control	3. Feedforward control
Median	0.256	0.226	0.181
Maximum	0.673	0.583	0.446
Max outlier	1.562	1.176	1.001

Compared to the uncontrolled trajectory, the computed mean error along a 10 percent arc path is reduced by more than 85 percent, with peaks of more than 90 percent (see Fig. 4.5). The errors shown in this figure are computed by the control logic with respect to time.

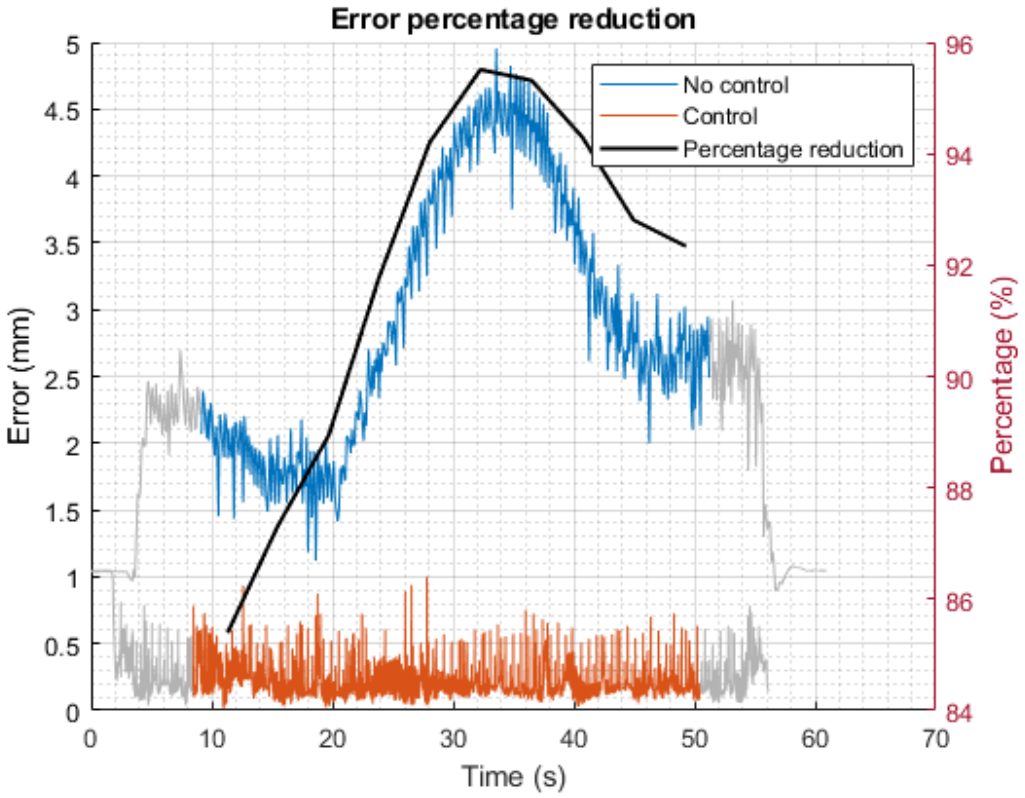


Figure 4.5: Error reduction at low speed

The final spatial error between the performed trajectory and the target path, instead, is different, and obtained by computing the minimum distance to the target for each point of the trajectory.
Calculating the error regardless of time can be more meaningful for the final results, while for control purposes the time dependent component is the needed one.

In this case, the spatial error has a median value in the range of 0.1 mm, with the highest values not higher than 0.5 mm.

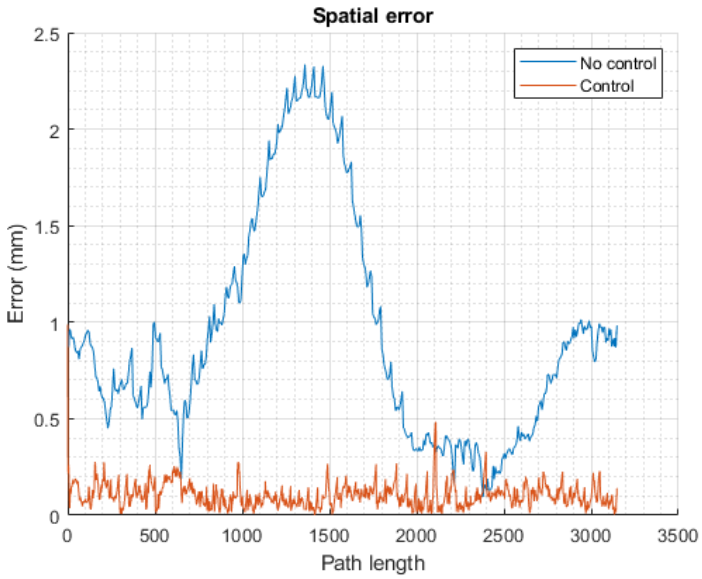


Figure 4.6: Spatial error with respect to the path length: low speed comparison

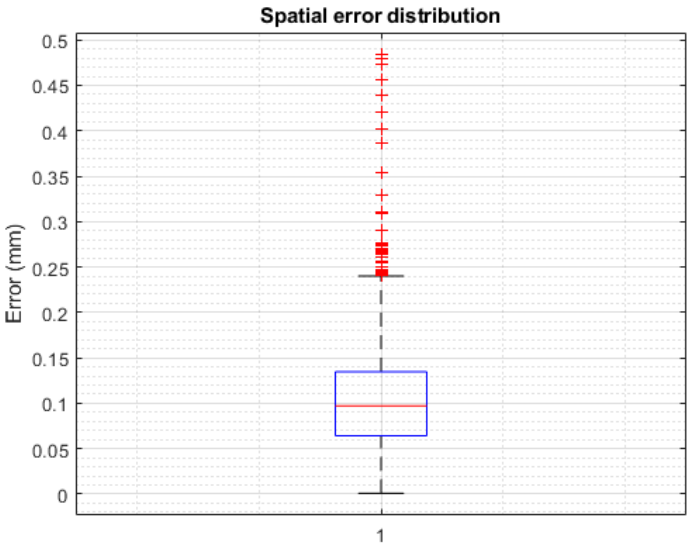


Figure 4.7: Spatial error distribution at low speed

High speed control

In the high speed case, the fuzzy logic once again allows reaching better results if compared to the simple linear proportional control, showing an improvement of 0.05 millimeters as reported in Table 4.3.

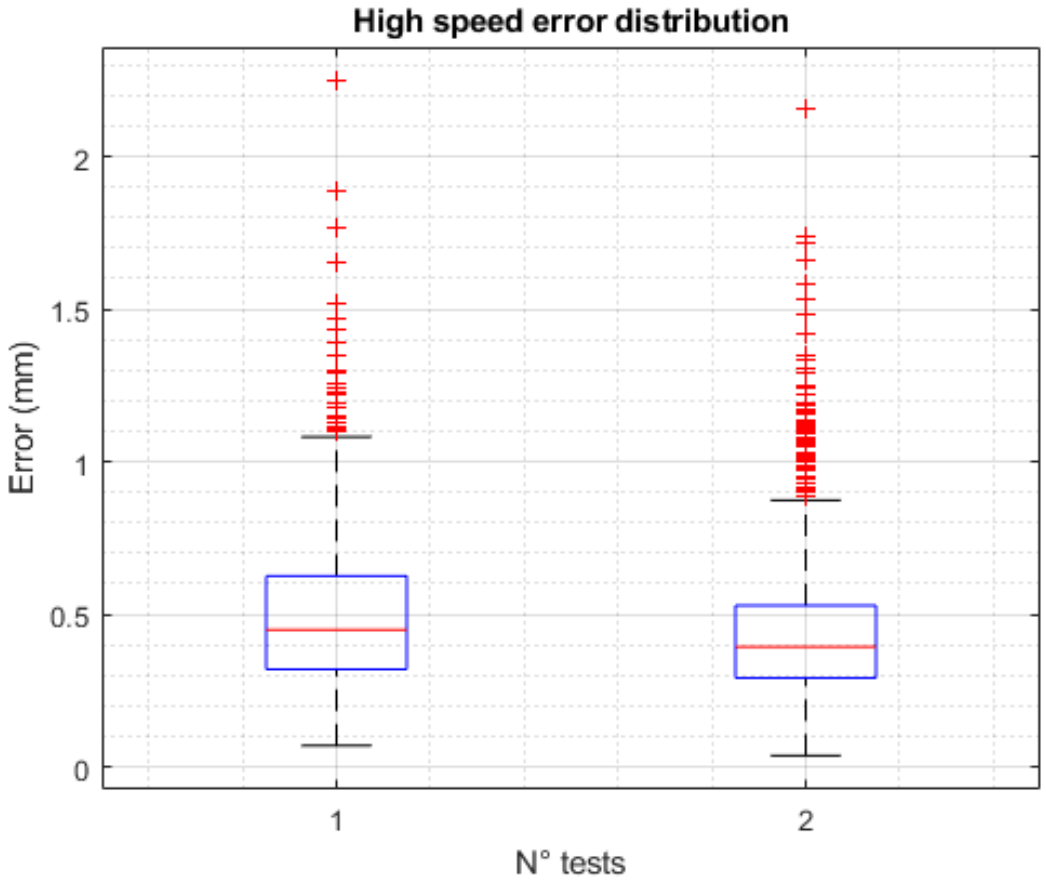


Figure 4.8: Error reduction at high speed: linear and fuzzy

Table 4.3: Dynamic path control at high speed

Error (mm)	1. Proportional control	2. Fuzzy logic control
Median	0.449	0.392
Maximum	1.080	0.874
Max outlier	2.248	2.160

As for the low speed case, the high speed tests showed a reduction of the error with respect to time of 90 percent or more (see Fig. 4.8).

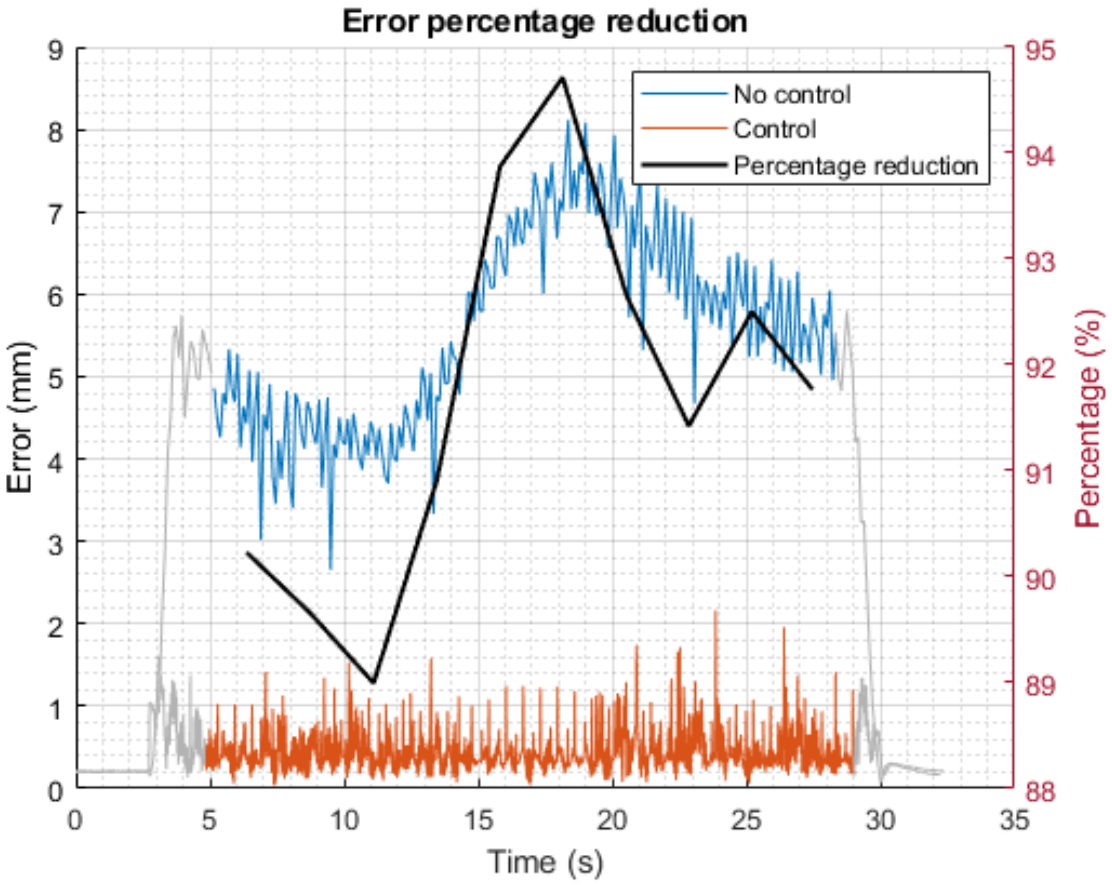


Figure 4.9: Error reduction at high speed

Similar behavior can also be seen for the spatial error computed along the path length. The final result is a median error of 0.2 mm, with the maximum as high as 0.7 mm.

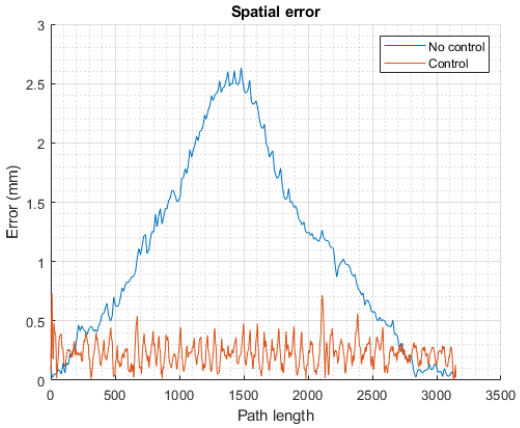


Figure 4.10: Spatial error with respect to the path length: high speed comparison

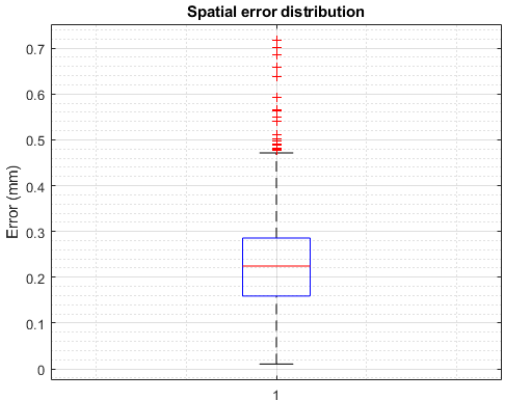


Figure 4.11: Spatial error distribution at high speed

Circular path radius

As described in section 3.6, the main testing trajectory is a circle with a 1 meter diameter defined on the x-y plane and inclined along the z-axis.
By plotting the internal data coming from the robot, it has been discovered that the target path, set by the software of the robot controller, is not perfectly circular in the x-y plane, but the radius varies along the circumference.
A similar test, with a flat trajectory with a constant value of z-axis position, doesn't show the same behavior but a constant x-y circumference radius, as illustrated by Fig. 4.12.

This behavior could be explained by the robot internal control trying to keep the tool speed constant in space, thus the z-axis motion requires the radius to be reduced so to balance the extra velocity component. The trajectory passes more precisely at 90°, 180° and 270° since are the positions in which the path points are set, while in between each one of them the trajectory is generated by the robot controller. In this work, this behavior has not been considered, as the reference pose has been kept equal to the one computed by the robot internal control, but could be interesting to investigate in future studies.

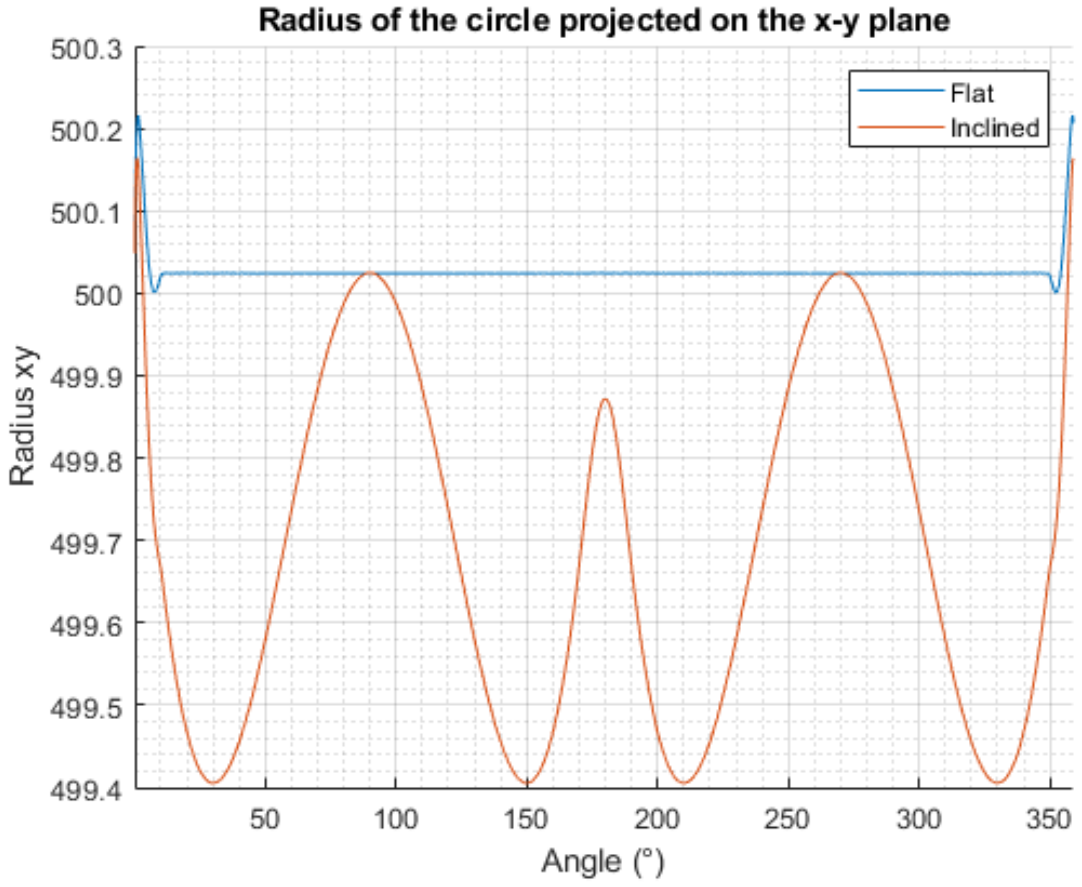


Figure 4.12: Circle radius with respect to the angular position

Addition path from ISO normative

To test the control logic also on different working paths, an additional case is taken, derived from the suggested path of the normative IS0 9283:1998(E) [12]. The concerned test path consists in a trajectory composed by curved and straight lines, circles and right angles, which allows evaluating the performances of the controller over different conditions.

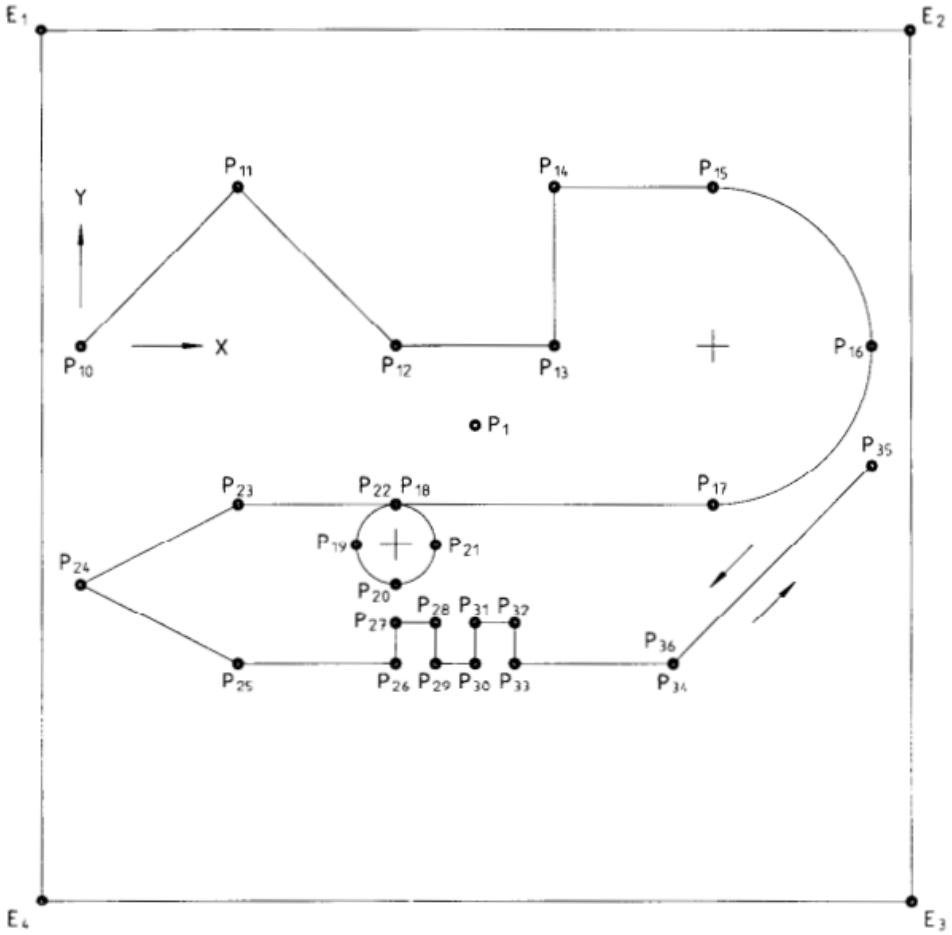


Figure 4.13: IS0 9283:1998(E) - Optional test path

Table 4.4: IS0 9283:1998(E) - Optional test path adjusted points coordinates

Plane 1200 x 1200 [mm]													
P	10	11	12	13	14	15	16	17	18	19	20	21	22
X	0	210	420	630	630	840	1080	840	420	360	420	480	420
Y	0	210	0	0	210	210	-30	-270	-270	- 330	-390	-330	-270

23	24	25	26	27	28	29	30	31	32	33	34	35	36
210	0	210	420	420	450	450	480	480	510	510	780	1080	780
-270	-390	-510	-510	-480	-480	-510	-510	-480	-480	-510	-510	-210	-510

The trajectory plane, which points coordinates are given in Table 4.4, is the 50% bigger than the one proposed in the norm, to account for the size of the robot under consideration and to enclose a wider variety of configurations. Moreover, in this specific work, it has been performed on an inclined plane of 45° with respect to the robot z-axis, so to test all the linear components of motion.
NB: The performed ISO path has a slight difference in the P_{35} point, being placed slightly moved along the y-axis, due to an error during the trajectory definition in Karel, which however does not impact the overall testing results.

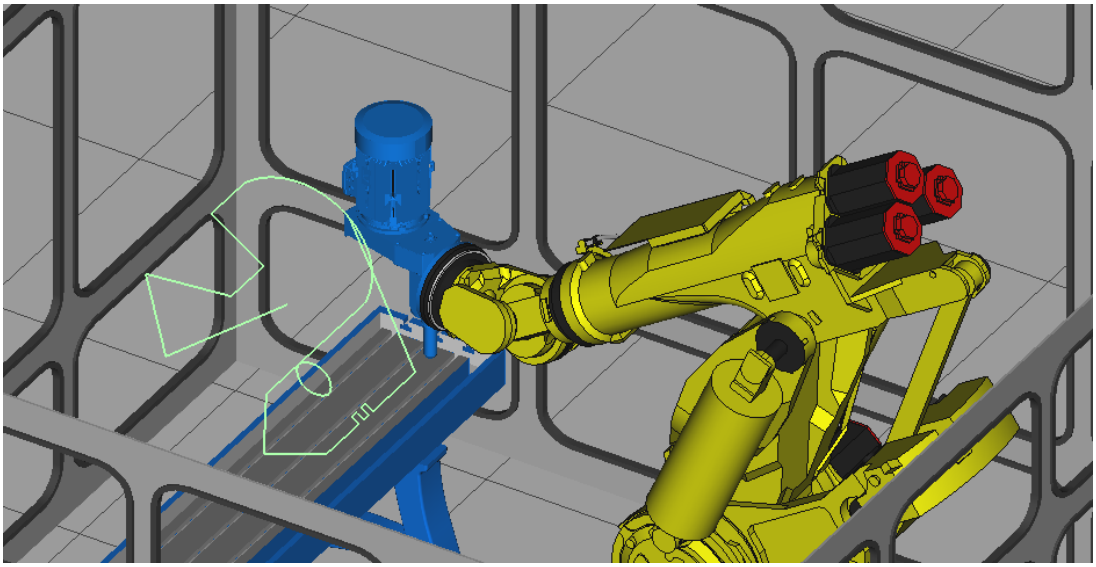
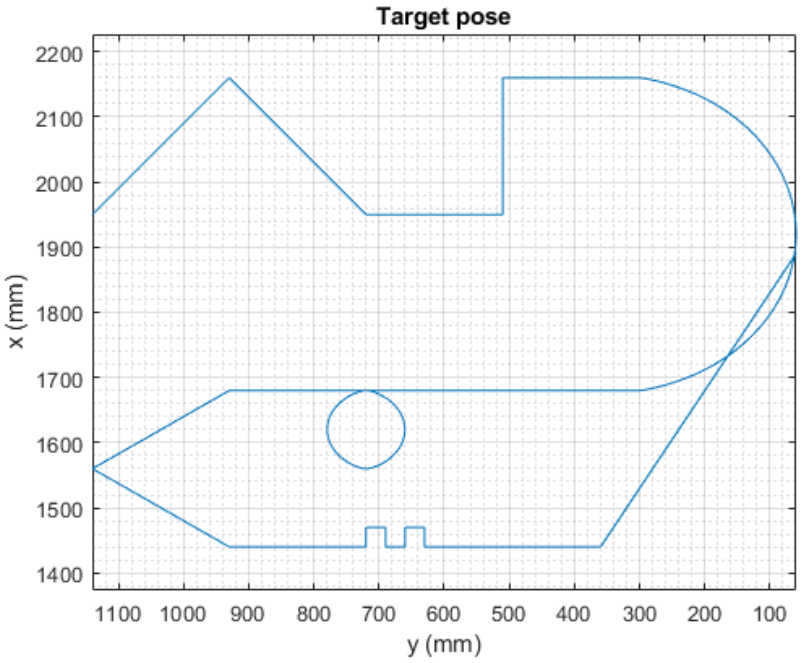


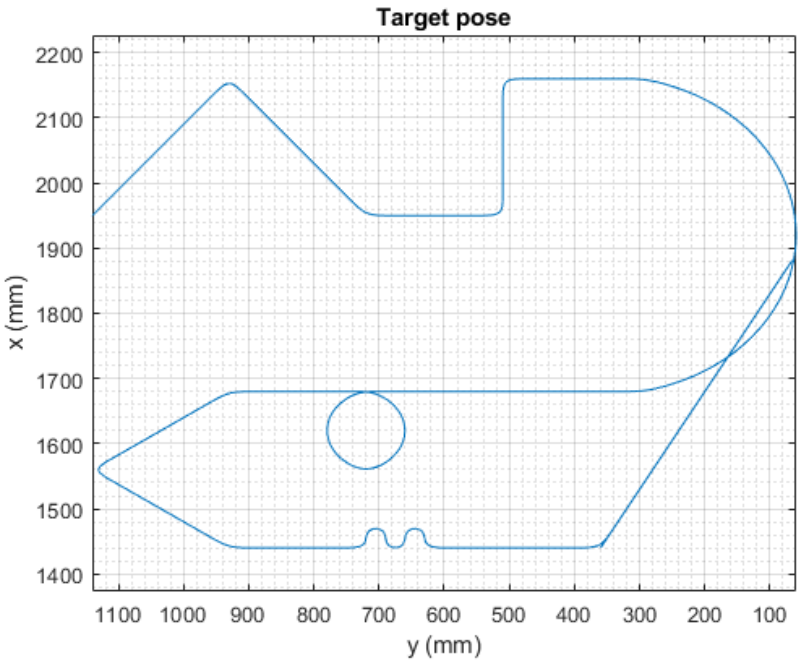
Figure 4.14: IS0 9283:1998(E) - Optional test path, Roboguide image

The control logic and testing method is kept identical to the circular path tests. It has been used the fuzzy logic control, developed and tuned with the circular path, and the trajectory has been performed with and without control, and with different Kalman filter options, so to compare the results for both slow and fast execution speed.
An additional variable considered during these tests is the *CNT* (continuous) value, used for the trajectory definition in the Karel program. It is a variable that can change between 1 and 100, and states how precise the robot should pass through a given point. It is in contraposition to the *FINE* option, previously presented in section 2.2, which instead focuses on reaching the targeted position.
The higher the *CNT* value, the less precise the movement is, but the execution speed can be maintained higher. To do so, however, the actual trajectory of the robot loose adherence to the theoretical one, by rounding the corners of the path, so to compensate for the higher execution speed.

Two *CNT* values have been used, 1 and 50, and the resulting paths are shown in Fig. 4.15.



(a) *CNT*: 1

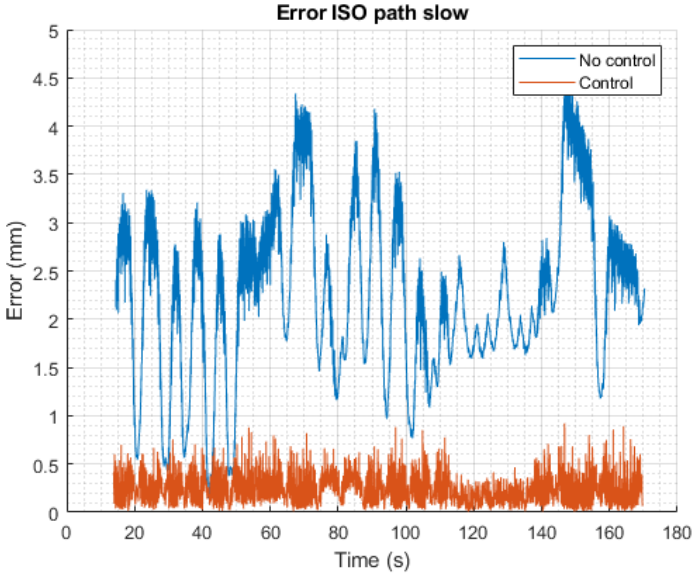


(b) *CNT*: 50

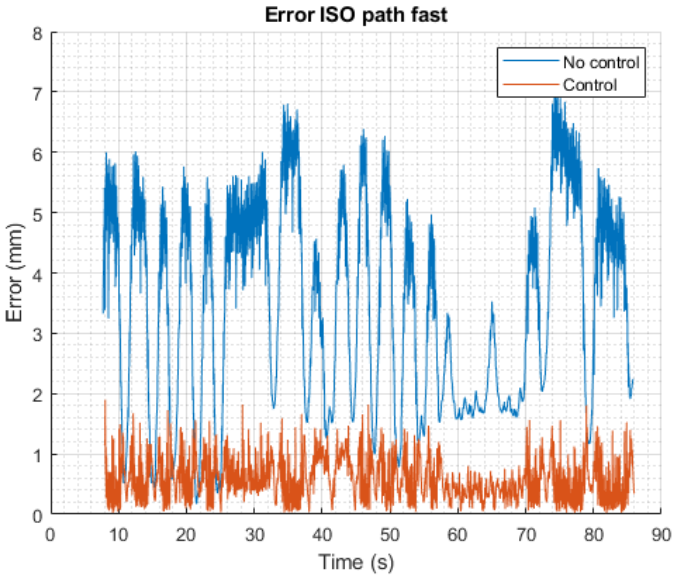
Figure 4.15: Actual trajectory of the ISO path at different *CNT* values

CNT 1

As derived from some first tests without controller action, the uncontrolled error with respect to the target pose (in blue) varies with time, increases up to several millimeters, especially in correspondence to the corners and curves.
The control action, based on the fuzzy logic tested with the circular path, allows reducing it and stabilizing the error at low magnitude (curves in orange), at both high and low speed (see Fig. 4.16).



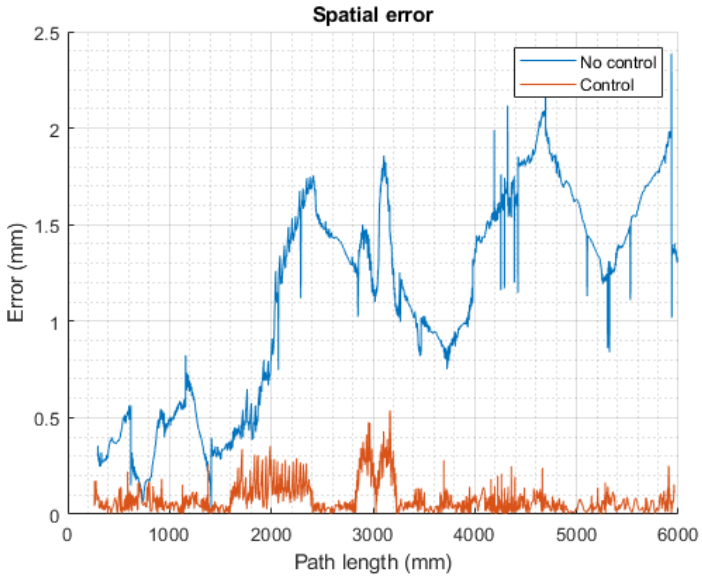
(a) Low speed



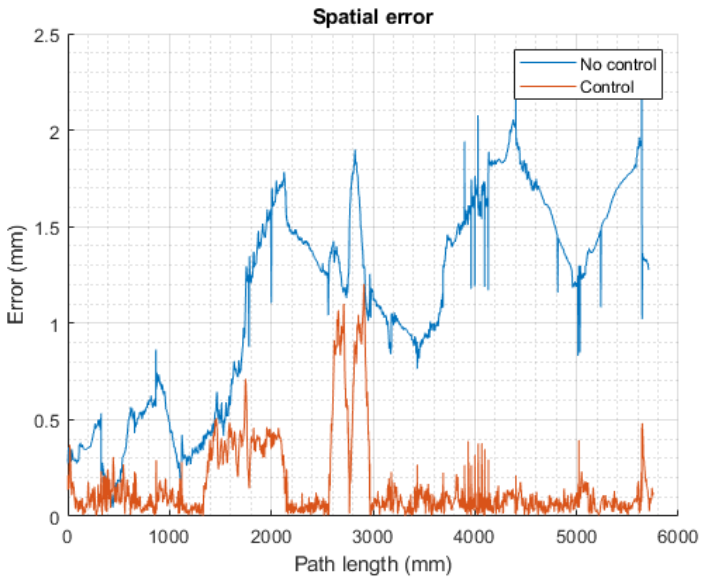
(b) High speed

Figure 4.16: Error reduction over time for ISO path

As well as for the circular path, the spatial error can be computed along the path length. In the plots of Fig. 4.17 it is appreciable how the error is significantly lower compared to the one over time, since this error formulation does not account for delay in positioning but only focuses on the trajectory of the path.
In particular, it drops over 2 millimeters for the uncontrolled test, and it is fairly reduced also for the controlled one.



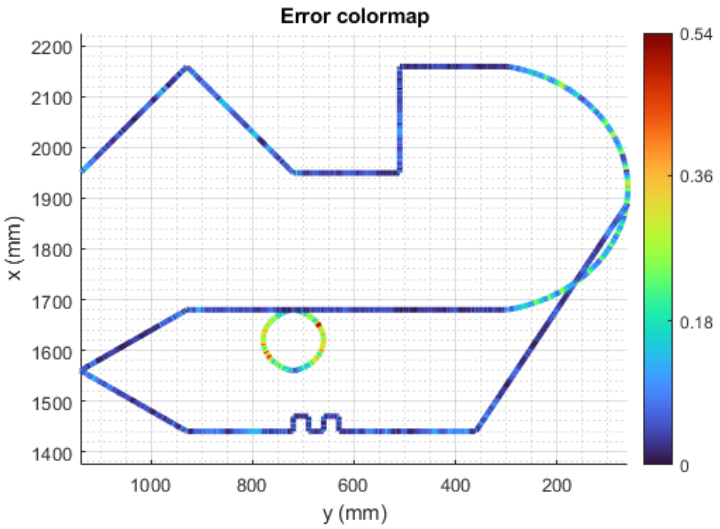
(a) Low speed



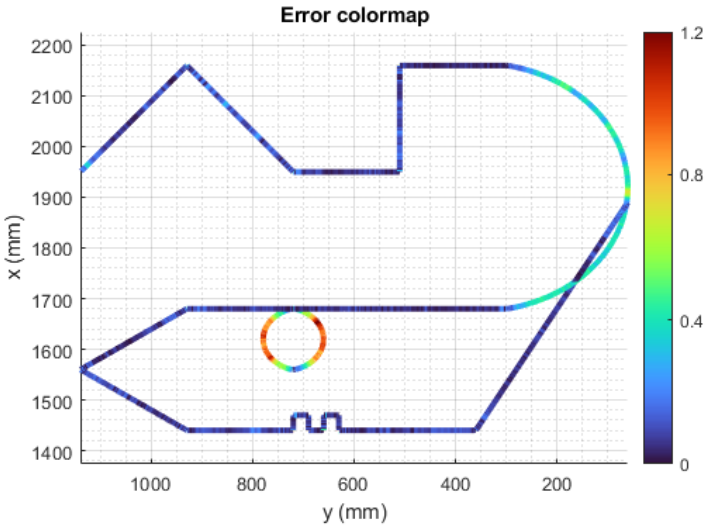
(b) High speed

Figure 4.17: Spatial error reduction over path length for ISO path

Furthermore, it can be noticed how the maximum spatial error is concentrated in two parts: the semicircular trajectory and the smaller full circle, as also highlighted by the color map plots in Fig. 4.18.
Especially in the latter, it reaches its maximum values of 0.54 mm and 1.2 mm for the slow and high speed tests respectively.



(a) Low speed



(b) High speed

Figure 4.18: Color map of spatial error for test under control

CNT 50

The tests carried out with the *CNT* value equal to 50 shows an error, with respect to the target pose presented in Fig. 4.15b, similar to the trajectory for *CNT* equal to 1.

Indeed, when comparing Fig. 4.16a with Fig. 4.19, and Fig. 4.17a with Fig. 4.20, the same behavior is evident. However, the final spatial error is higher in some points, in particular in correspondence to the sharp turns, while the execution time is diminished, due to the more rounded corners that allow for more speed at the expense of lower precision.

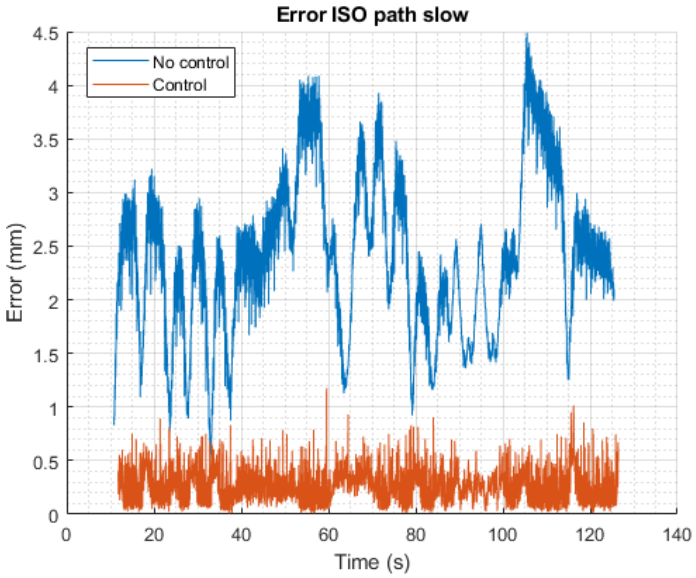


Figure 4.19: Error reduction at low speed with CNT = 50

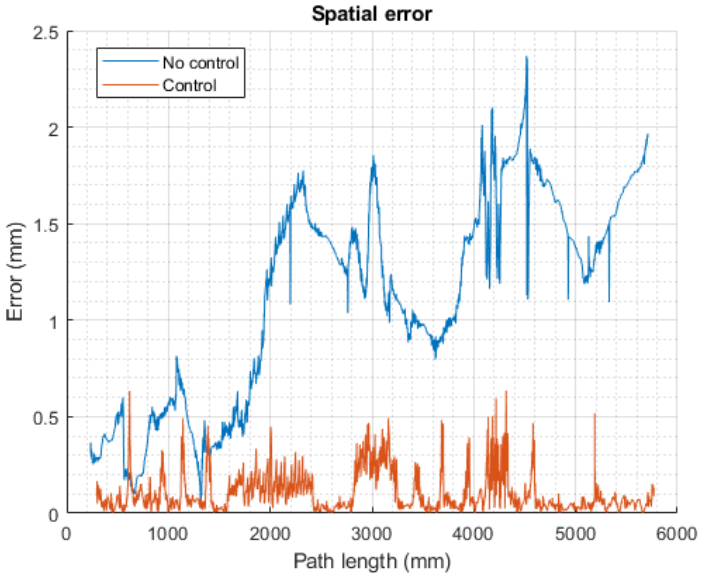


Figure 4.20: Spatial error reduction at low speed for CNT = 50

Orientation error

In the previous sections, the results related to the linear error have been analyzed and shown. However, the control logic also allowed to compensate the orientation error.
As mentioned in section 3.6, the orientation error at the toll center point is usually quite small if compared to the linear one. Additionally, a strong control of the orientation error could lead to decrease of precision for both linear coordinates and orientation.

For completeness, also the orientation error is plotted (see Fig. 4.21), showing the comparison between the case without control strategy and the one with active control.
As it can be noticed in the plots of the left column, the orientation error without control, along the ISO path at high speed (120 mm/s), is in the range of +/- 0.2 degrees. The control action, present in the plots of the right column, helps in aligning the tool at the correct orientation, keeping it within 0.1 degrees, or even less, from the target pose.

In the case of orientation error, the improvement is less marked if compared with the linear coordinates, since the starting point is already accurate. However, the control action is confirmed to help with the final result, without generating unusual behavior that could lead to bigger errors.

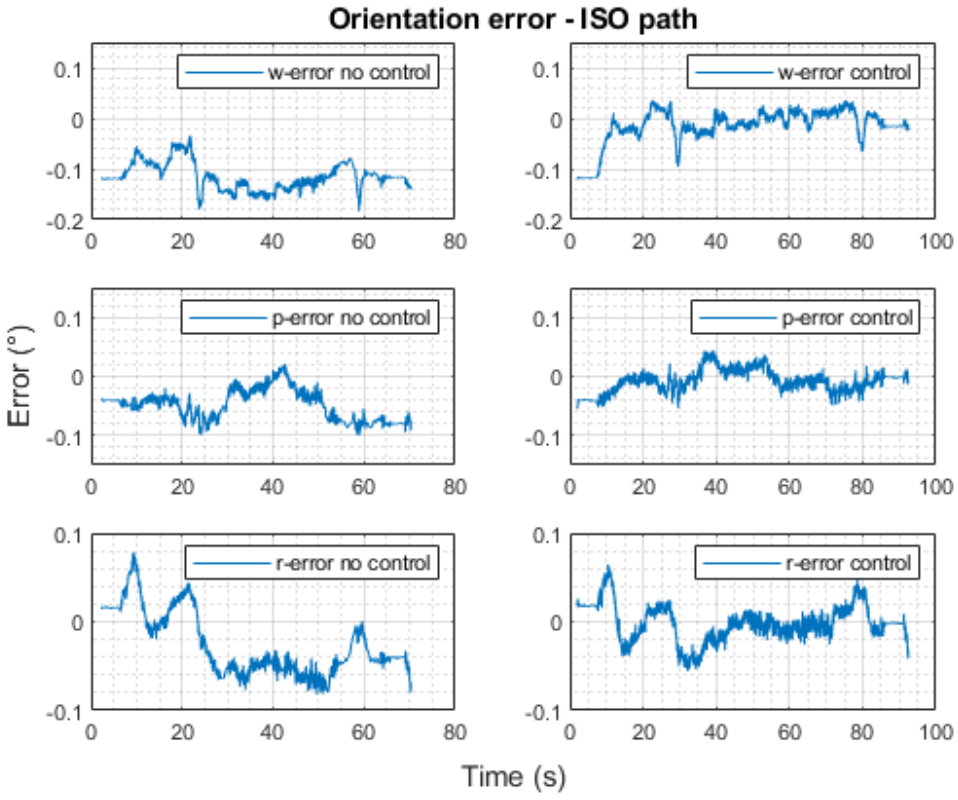
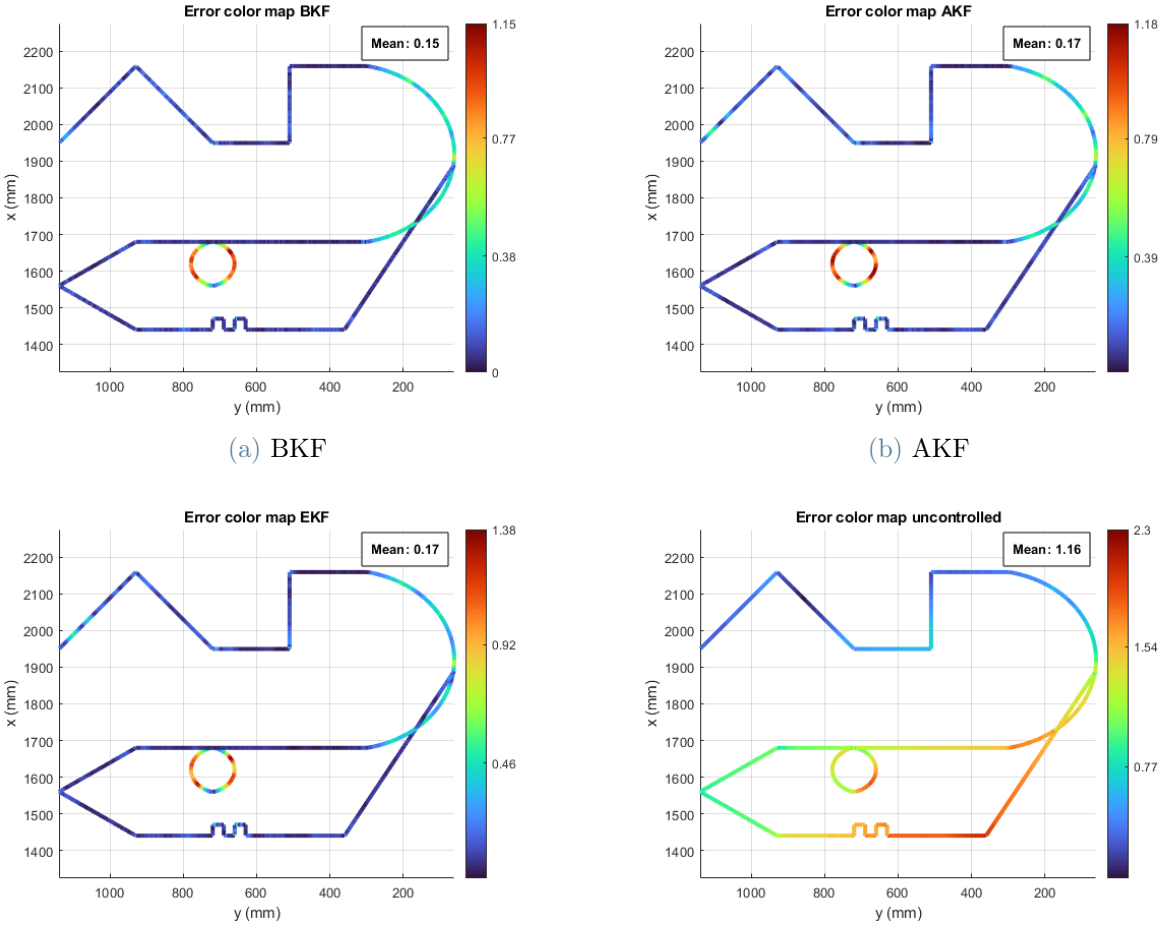


Figure 4.21: Orientation error for the high speed ISO path

4.1.3. Comparison of Kalman filters

The results reported in chapter 4 up to this section are obtained through the use of the BKF, even if measured data were available for all the three Kalman types (BKF, AKF, and EKF). This choice is due to the lack of impacting differences in the resulting error, as previously shown during tuning post-processing in section 3.5.5.
However, some findings can be extracted from the analysis of the Kalman behavior, reason why a comparison is performed in this section.

The error mean values from all the controlled cases are comparable, with a reduction up to 87% of the mean error and up to 50% of the maximum error, as shown in Table 4.5. On the other hand, both the AKF and the EKF seem to be detrimental for the maximum error and the standard deviation. The critical error areas are the same, in particular the small circle, that requires continuous changes of direction at high rate.



(a) BKF (b) AKF

(c) EKF (d) Uncontrolled

Figure 4.22: Kalman's comparison for ISO path at high speed

NB: The plots show average values over multiple trials. The first and latter 10% of measures are cut for reducing outliers presence. For the plots showing the circular path and the slow speed ISO path color maps, see appendix B.

Table 4.5: Comparison of positional errors for KFs

Averaged statistics for error values (mm)				
	BKF	AKF	EKF	No control
Mean	0.15	0.17	0.17	1.16
Max	1.16	1.24	1.43	2.31
SD	0.18	0.20	0.20	0.53

The keys of the lack of improvement of the more complex Kalman filters exploited, is probably to research in two causes: cameras' noise and filtering delay. The noise might prevent the recognition of smaller values for the error, also provoking higher outliers to be taken into account in the estimation of the mean values.
On the other hand, the small delay due to filtration effect might affect the control responsiveness and, thus, worsen the performances. Indeed, as shown in Fig. 4.23, it is more present, due to higher filtration, in the AKF and EKF.

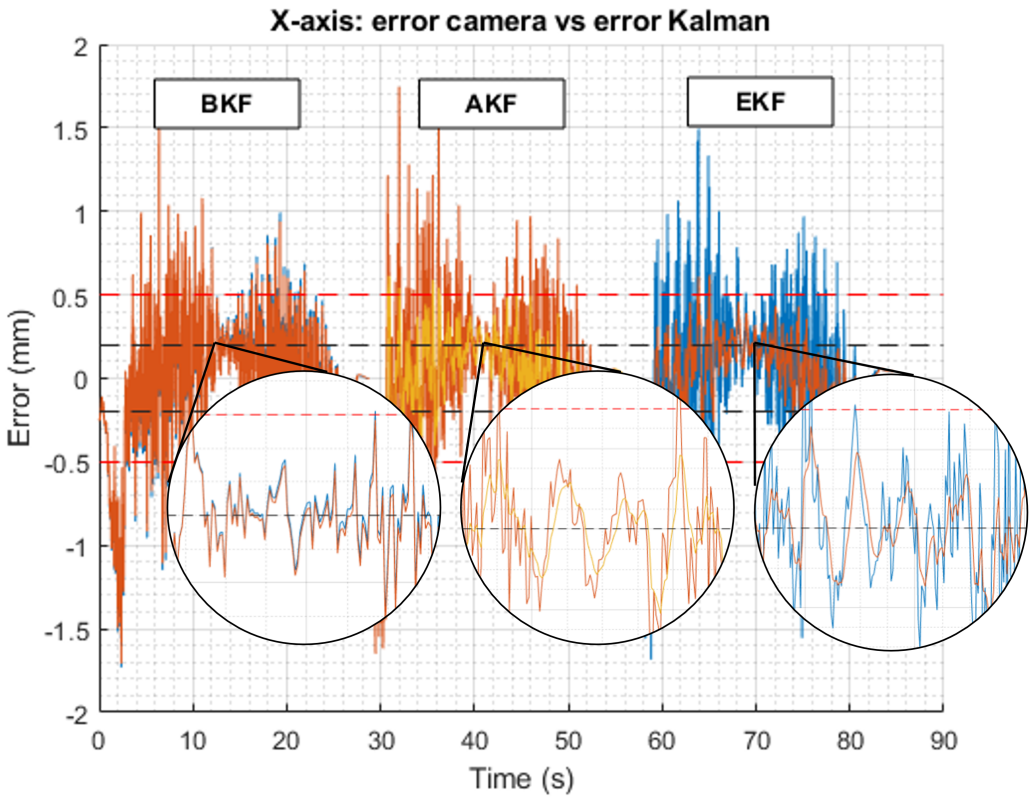


Figure 4.23: Zoom on Kalman filtering effect

Laser measurements comparison

The camera tracking system, communicating in real-time with the robot through *FaRoC*, has been used for obtaining the actual pose of the end effector and, thus, for performing all the online control functions. On the contrary, the laser tracker system had not integrated in the available packages the real-time feature, which is expensive to purchase. An in-house code is under development, but will be available only to future works.

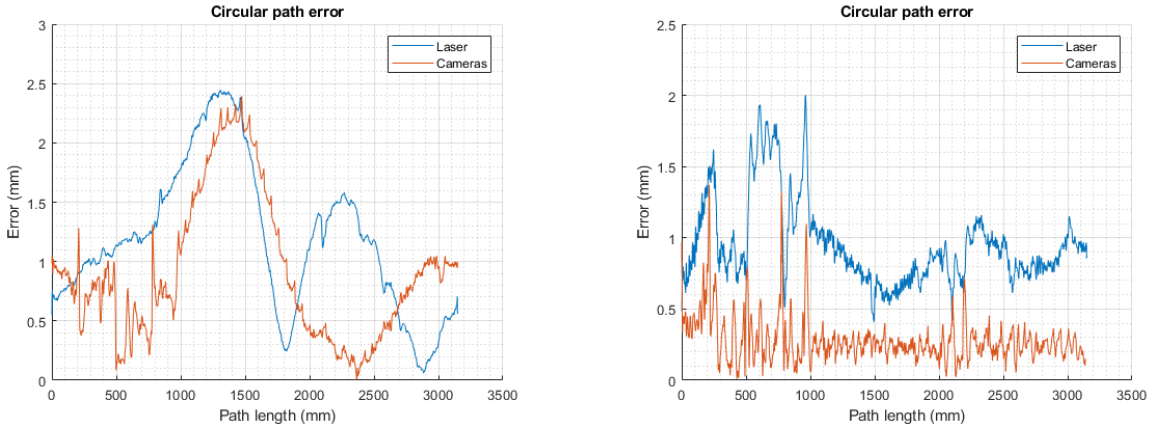
Nonetheless, to assess the precision of the control logic based on the camera measurements, a laser tracker has been used to measure the end effector position, and then to make a comparison during the post-processing activity. Therefore, in this thesis work, the laser measurements are not used for control purposes, but only as an external reference.

Furthermore, it is important to denote that, as stated before, a direct comparison of the two measurement systems over time is not easy to perform. The sampling rates are different, 150 Hz for the cameras and 200 Hz for the laser, and most importantly it has been not possible to match the internal clocks due to inner settings.

Also for these reasons, the laser tracker and camera measurements are clearly different, for both magnitude and general trends, which can be more accentuated in some parts than in others. The difference in values affects the final controlled error, which is minimized based on the data coming from the cameras (in orange in Fig. 4.24 and Fig. 4.25), while it remains quite large considering the laser measurements (in blue).
Moreover, a different dynamical behavior can be seen, as the laser tracker tend to show a smoother trajectory, while in the data coming from the cameras some random spikes not registered by the laser, due to noise or error in the pose detection. This is particularly evident in Fig. 4.24a, while in the controlled runs it is overcome by the continuous corrections applied.

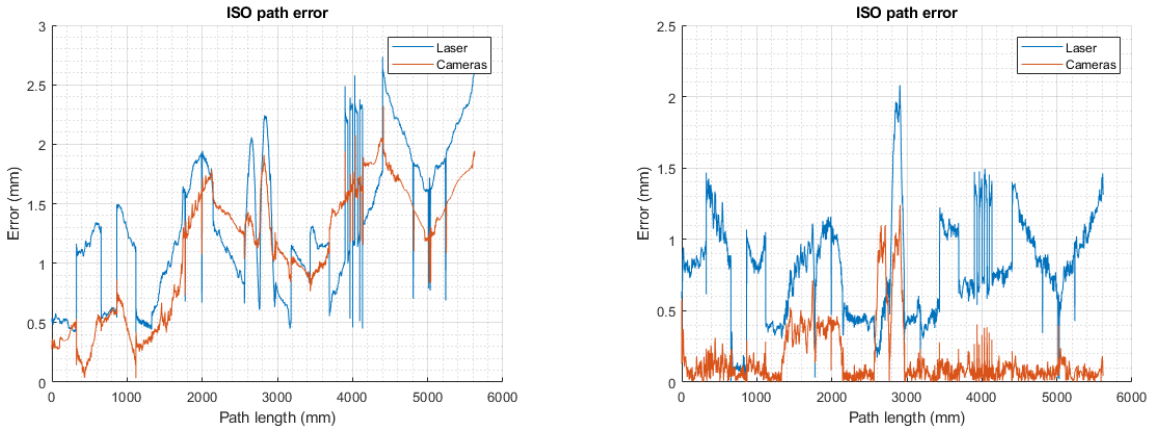
Another explanation for the generalized offset trend between the two measurements can be the different calibration and some possible errors that can arise during this procedure. However, for what concerns the aim of this thesis work, the results are satisfactory, based on the cameras' measurements, as the developed control logic is independent of the measuring tools.

In the end, it could be possible that different instruments may lead to different final results and require change in the control parameters due to the changes in the quality of the data, but this aspect is left for further study.



(a) Uncontrolled (b) Controlled

Figure 4.24: Tracking system comparison: circular path spatial error at high speed



(a) Uncontrolled (b) Controlled

Figure 4.25: Tracking system comparison: ISO path spatial error at high speed

4.2. Findings

The research carried out on the robot system has allowed to get a better understanding of the IR dynamic behavior and the optimal strategy for its control. Eventually, both the static and dynamic control managed to improve the overall accuracy of the robot pose, with different working conditions.

For the static case, the final pose error can reach very low magnitudes, even below 0.05 mm, regardless of the position considered. The main limitation in this factor is the real precision of the measuring instrument, which will determine the final precision of the robot as well.
The system response to disturbances has been studied, which suggested the use of a fuzzy logic control due to the different behavior at high and low error magnitude, thus allowing to keep a fast response without compromising the steady state behavior.

The dynamic tests showed a significant improvement, in the order of millimeters, with the addition of the control action.
The main difficulty is represented by the high frequency vibration along the path, which are unpredictable and constitutes the major source of error.
As in the static case, the fuzzy logic helps in the improvements of the performance, allowing to keep high gain for higher error magnitudes without resulting in excessive correction when close to the target.
The control logic has proved to be consistent even with different paths and geometries, as shown by the ISO trajectory tests.

In the case of the dynamic control, also the possibility of implementing a feedforward action could help in a further increment of precision, at the cost of losing generality of the control logic, which becomes path dependent, requiring it to be computed for each specific trajectory.

5 | Conclusions and future developments

5.1. Summary of Findings

The work carried out on the FANUC M-900iB/700 allowed to better understand the dynamics and the behavior of an IR. Different tests have been performed with the aim of studying the controllability of the robot in various situations and positions. The study can be split in two main cases: static control and dynamic control.

Regarding the system dynamics, it has a complex behavior, influenced by many factors, such has the joint motors, the pose, the coordinate considered and more. However, the joint motors can be reasonably described by a first order TF, while the arm structure can be approximated using a second order TF. Furthermore, the system exhibits delays coming from the internal control of the robot governing the movement of joint motors, and that can be estimated to be around 50 ms in terms of the system's response.

For what concerns static control, it has been acknowledged that linear control strategies may be effective but are limited in enhancing the performance beyond a certain threshold, mainly due to low frequency oscillations at steady state. To address this limitation, implementing fuzzy logic control proves beneficial, leading to improvements for both speed and accuracy at steady state. High error magnitudes are quickly compensated with significant control gains, while, for a smooth and precise response, low gains are the suitable ones. This control logic has been tested and demonstrates efficient operation in several positions within the robot's workspace, and in different pose configurations and orientation.

In contrast, dynamic control represents a more challenging scenario compared to static control. Uncontrolled errors can be within a few millimeters, characterized by low frequency constant behavior perturbed by unpredictable oscillations and disturbances. Moreover, the robot's behavior is speed dependent, with higher speeds leading to larger errors. Similar to the static case, linear control reaches a performance limit, and the fuzzy logic aids in achieving better results by facilitating improved gain tuning.

For dynamic control, a Kalman filter has been designed and tested to enhance the estimation of the system's state from noisy measurements. Furthermore, additional versions, incorporating disturbances as expanded state variables, have been implemented, along with an extended model addressing non-linearities in the system model, particularly in relation to the position of the end effector.
However, tests have revealed no significant improvement, thus a better understanding of camera measurements and error computation is needed. Indeed, utilizing the same camera measurements, that in the controller are filtered by the Kalman filter, results in a degradation of the ability to assess and evaluate performances. This issue arises because, under a certain threshold, to distinguish whether error outliers are due to noise or poor control becomes challenging.
Moreover, the control logic has been applied to different paths and geometries, consistently enhancing results in each case. Circular trajectories and small-radius curves pose primary challenges, where the error is more difficult to compensate due to the higher rate of direction changes.

5.2. Outcomes

The achieved results for both static and dynamic scenarios demonstrate a considerable reduction in the error of the end effector, both in terms of mean and maximum values.

In static control, the control outcome remains unaffected by the initial error magnitude, which can be within the range of a few millimeters. Consequently, performance evaluation involves applying step inputs through the control input channel and measuring the response using the camera tracking system.
The fuzzy logic exhibits an impressing ability to compensate for errors by more than 50 percent in 0.5 seconds, when perturbed by 1 millimeter disturbances. Furthermore, within the 0.3-0.2 mm threshold, the control switches to lower gains, enabling accuracy without oscillations, and approaching the reference with an error as minimal as 0.05 millimeters. It is a remarkable result, considering that the repeatability limit of the robot is ± 0.1 mm, therefore with this value the control successfully achieves its goal.

Dynamic control also shows improvement with respect to the case without control. The trajectory error cannot be reduced to zero as in static control, since more factors influence the robot path accuracy, such as velocity, trajectory, curvature radius, and external conditions, but can be greatly reduced by the control action. In particular, two paths have been designed and tested: a circular path on the horizontal plane with a 1 meter diameter, and a polygonal path derived from the normative IS0 9283:1998(E) [12].

In addition, the first one has also been inclined on the vertical axis, to stress at the same time the control of the z-axis coordinate. The *"ISO"* path, instead, has been expanded, with respect to the ISO proposed one, to a working plane of 1200x1200 mm. Moreover, the tests have been conducted at low (60 mm/s) and high (120 mm/s) speeds.

For what concerns the circular path, the control action allows keeping spatial error below 0.5 mm for the maximum values and below 0.25 mm for more than 90 percent of the trajectory at low speed. At high speed, the error remains within 0.75 mm, with most of the time below 0.5 mm.
In the ISO path, results vary along the path according to the specific geometry encountered. Overall, the error has been kept consistently below 0.2 mm at low speed in the straight sections, reaching a maximum of 0.5 mm in circular parts. Similar behavior is seen also at higher speed, with the maximum error reaching 1 mm in circular sections, while being kept at less than 0.3 mm in the straights parts and corners.

The implementation of Kalman filters, while promising, is still in the process of being fully mastered. On a positive note, the successful and seamless integration of filtered measures into the control logic occurred without significantly compromising the sampling time. Additionally, through proper tuning, it has been possible to optimize the filtration magnitude and achieve low error values.
On the other hand, as discussed in section 4.1.3, no remarkable improvements are exhibited in the expanded versions. The uncertainty surrounding the causes, possibly to research on reduced responsiveness due to delays or in data background noise, has hindered further exploration of this behavior. The laser tracker measurements, aimed to be used as reference values to perform a comparison with the cameras, were not up to the task because possibly subjected to calibration errors or external disturbances.

In summary, the control logic demonstrates effectiveness in delivering reliable outcomes, ensuring a consistent pattern of error values in numerous tests conducted under multiple paths and working conditions. Especially in terms of absolute accuracy, the precision of the robot system undergoes a substantial enhancement and proves highly resilient against minor dynamic disturbances, as those associated with gradual changes in direction.
Nevertheless, it is important to acknowledge a slight reduction in repeatability over test repetitions compared to the uncontrolled scenario. This decrease is attributed to the added influence of the controller's sampling time and the camera's repeatability, highlighting the need for careful consideration of these factors in the overall evaluation of the system's performance.

5.3. Discussion

The implementation of feedback control for the IR tool end effector has proven to be an effective and relevant improvement for what concerns the final precision. This control operates independently of the measurement system, relying solely on external position data for comparison with the desired pose.
However, the ultimate accuracy achieved depends also on the reliability of the chosen measuring tool, as the proposed control logic can minimize the computed error with respect to the obtained data. For instance, discrepancies arose between the cameras and laser measurements. Eventually, the error computed based on the camera measurements has been compensated, while according to the laser tracker the results were still imprecise.

At the moment, this presents the limits of the system. It is known that the precision of cameras is optimal within their viewing cones, but diminishes with increasing distance, potentially posing challenges in larger working spaces. Laser trackers, on the other hand, encounter visibility issues in complex path geometries, which can prevent the communication with the target. Nonetheless, the deployment of multiple cameras, as in this work's setup, mitigates these issues. Furthermore, advancements in camera tracking systems have enabled the attainment of accuracy surpassing the robot's repeatability, which is obtained in this work in static scenarios and at slow speed, i.e., for lower dynamics. This can already be a robust foundation for high-precision absolute positioning processes, particularly relevant in the aerospace industry where tolerances are stringent.

In any case, while defects along straight paths or curves with large radii are overcome, certain limitations persist regardless of the tracking system. Rapid changes in direction leading to overshooting and outliers at corner points, as well as disturbances triggering the natural frequency of the robot, remain challenges. Notably, also the application of expanded Kalman filters imposes restrictions on the dynamics and bandwidth of the external controller, reacting to very high-frequency changes with small manipulated variables.

5.4. Suggestions for Future Research

The aforementioned challenges underline the ongoing complexities in achieving optimal control in dynamic scenarios, and might be the object of further study.
For instance, the conducted tests involved the end effector being free to move in space, devoid of external loads and without performing manufacturing tasks. Therefore, it could be interesting to investigate the final accuracy under working conditions, such as cutting, grinding, machining or similar operations, and assessing the resulting quality of the manufactured piece.

Within the scope of this thesis, a Kalman filter has been developed to estimate the system's state based on camera measurements. However, understanding its contribution to the control action proved to be complex, as the final results exhibited minimal changes between tests with high and low filter magnitude. Investigating the camera measurements and the applied offset computation mechanism imperative for gaining a deeper comprehension of the tracking system's behavior.
Furthermore, introducing a laser tracker's communication protocol could provide reference measurements and valuable insights to compare the controllability with different qualities of data, offering a comparative analysis with cameras in terms of final outcomes in manufacturing processes.
Additionally, another aspect that can be improved is the optimization of the algorithms and scripts employed in the control logic for computing the output, ultimately leading to a reduction in sampling time. This optimization effort, in particular, promises enhanced efficiency and responsiveness in the control system.

5.5. Conclusion

The work carried out in this thesis aimed at overcoming one of the main challenges and limitations of today's use of industrial robots. Indeed, the pose control of an IR based on external measuring system, as camera tracking systems, can represent a relevant improvement for the end effector pose and path accuracy, a crucial aspect in industrial production.

By adopting direct position control, along with fuzzy-logic and Kalman filters, for improving the efficiency and versatility of industrial robotic systems, the developed control logic has demonstrated to achieve significant results in different robot poses and trajectories, possibly allowing better manufacturing results and increase the range of feasible applications for industrial robots, thanks to its improved capabilities.

In conclusion, the insights and understanding gained in this study of industrial robot control provide a foundation for future research, encouraging further exploration of external tracking system in direct position control, and they suggest new opportunities for future advancements in the realm of automation and precision engineering that will shape the industry of tomorrow.

Bibliography

[1] International Federation of Robotics. Market presentation World Robotics 2023. Extended version. Retrieved: 2023-11-27, 2023. URL https://ifr.org/img/worldrobotics/2023_WR_extended_version.pdf.

[2] Eckart Uhlmann and J. Krüger. *Industrieroboter*, pages 1655–1663. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. ISBN 978-3-662-54805-9. doi: 10.1007/978-3-662-54805-9_126. URL https://doi.org/10.1007/978-3-662-54805-9_126.

[3] Christian Möller. *Entwicklung eines hochgenauen Bearbeitungsroboters durch den Einsatz zusätzlicher Messtechnik*, volume 4. Fraunhofer Verlag, 2019. ISBN 978-3-8396-1600-0.

[4] R. Möller. Machining of CFRP with Water-Jets in Aerospace Industry. Lecture in: ICMAC 2011 – International Conference on Manufacturing of Advanced Composites, Belfast, 2011.

[5] Julian Blumberg, Zhoulong Li, Lemopi Isodore Besong, Mitchel Polte, Johannes Buhl, Eckart Uhlmann, and Markus Bambach. Deformation error compensation of industrial robots in single point incremental forming by means of data-driven stiffness model. *2021 26th International Conference on Automation and Computing (ICAC)*, pages 1–6, 2021. doi: 10.23919/ICAC50006.2021.9594138.

[6] Juqing Yang, Dayong Wang, Baixing Fan, Dengfeng Dong, and Weihu Zhou. Online absolute pose compensation and steering control of industrial robot based on six degrees of freedom laser measurement. *Optical Engineering*, 56:034111, 03 2017. doi: 10.1117/1.OE.56.3.034111.

[7] VDI 2860. *VDI 2860: Montage- und Handhabungstechnik*, 1990.

[8] ISO 8373:2021. *Robotics – Vocabulary*, 2021.

[9] A V Novikov, A A Barmin, A A Mudrova, and U V Yasinskaya. State and prospects of industrial robotics in shipbuilding. *Journal of Physics: Conference Series*, 1333 (5):052029, oct 2019. doi: 10.1088/1742-6596/1333/5/052029. URL `https://dx.doi.org/10.1088/1742-6596/1333/5/052029`.

[10] B. Rooks. Robot welding in shipbuilding. *Industrial Robot*, 24(6):413–417, 1997. doi: 10.1108/01439919710192527. URL `https://doi.org/10.1108/01439919710192527`.

[11] Christian Möller, Hans Schmidt, Nihar Shah, and Jörg Wollnack. Enhanced Absolute Accuracy of an Industrial Milling Robot Using Stereo Camera System. *Procedia Technology*, 26, 12 2016. doi: 10.1016/j.protcy.2016.08.050.

[12] ISO 9283:1998. *Manipulating industrial robots – Performance criteria and related test methods*, 4 1998.

[13] Eckart Uhlmann, Mitchel Polte, and Julian Blumberg. Enhanced Accuracy Specification of Heavy-Duty Industrial Robots and the Influence of Secondary Encoders. *Proceedings of the Machining Innovations Conference for Aerospace Industry (MIC) 2022*, 11 2022. doi: 10.2139/ssrn.4259370.

[14] Stefan Hesse, Viktorio Malisa, Ana Almansa, Brirgit Graf, Erwin Wagner, Markus Trenker, Roland Ambrosch, Wilfried Kubinger, and Christof Hieger. *Robotik-Montage-Handhabung*. Carl Hansa Verlag München, 01 2010. ISBN 978-3-446-41969-8.

[15] A. Ehm and Shaker Verlag. *Einsatz von Industrierobotern für die Bohrbearbeitung an automobilen Strukturbauteilen unter Berücksichtigung des thermischen Verlagerungsverhaltens und der Prozessinteraktion*. Schriftenreihe des PTW: "Innovation Fertigungstechnik". Shaker Verlag, 2016. ISBN 9783844042627.

[16] Eberhard Abele, Wilfried Polley, Alexander Ehm, and Maximilian Troue. Spanende Bearbeitung mit Industrierobotern - Thermische Einflüsse auf die Bearbeitungsgenauigkeit, 2013. URL `https://api.semanticscholar.org/CorpusID:92057555`.

[17] J. Tsai, E. Wong, J. Tao, H. D. McGee, and H. Akeel. Secondary position feedback control of a robot, 2013. U.S. Patent.

[18] C. Möller, H. Schmidt, P. Koch, and C. et al. Böhlmann. Real Time Pose Control of an Industrial Robotic System for Machining of Large Scale Components in Aerospace Industry Using Laser Tracker System. *SAE Int. J. Aerosp.*, 2017. doi: 10.4271/2017-01-2165.

[19] Hexagon AB. Leica Absolute Tracker AT960, 2023. URL `https://hexagon.com/en/products/leica-absolute-tracker-at960`. Retrieved: 2023-08-27.

[20] Jürgen Hefele and Claus Brenner. Robot Pose Correction using Photogrammetric Tracking. *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 170–178, 02 2001. doi: 10.1117/12.417194.

[21] Sepehr Gharaaty, Tingting Shu, Ahmed Joubair, Wen Xie, and Ilian Bonev. Online pose correction of an industrial robot using an optical coordinate measure machine system. *International Journal of Advanced Robotic Systems*, 15:172988141878791, 07 2018. doi: 10.1177/1729881418787915.

[22] Liao Wu and Hongliang Ren. Finding the Kinematic Base Frame of a Robot by Hand-Eye Calibration Using 3D Position Data. *IEEE Transactions on Automation Science and Engineering*, 14:1–11, 2017. doi: 10.1109/TASE.2016.2517674.

[23] *FANUC Robot series R-30iB/R-30iB Mate/R-30iB Plus/R-30iB Mate Plus/R-30iB Compact Plus/R-30iB Mini Plus CONTROLLER External Vision Interface, B-84244EN/02*. FANUC America Corporation, 3900 W. Hamlin Road, Rochester Hills, Michigan, 6 2022. Operator's Manual.

[24] MABI Robotic. MAX-150-2.0-P, 2023. URL `https://www.mabi-robotic.com/robotic/robotic-arm/product.asp?IDProdukt=26&userlang=EN`. Retrieved: 2023-10-09.

[25] Hexagon AB. Leica T-Mac, 2023. URL `https://hexagon.com/products/leica-t-mac`. Retrieved: 2023-08-27.

[26] Xiaojia Shi, Fumin Zhang, Xinghua Qu, and Bailing Liu. An online real-time path compensation system for industrial robots based on laser tracker. *International Journal of Advanced Robotic Systems*, 2016. doi: 10.1177/1729881416663366.

[27] Fanuc, Ltd. Robot M-900iB/700, 2023. URL `https://www.fanuc.eu/it/en/robots/robot-filter-page/m-900-series/m-900ib-700`. Retrieved: 2023-08-27.

[28] Julian Blumberg, Mitchel Polte, and Eckart Uhlmann. Estimation of External Force-Torque Vector Based on Double Encoders of Industrial Robots Using a Hybrid Gaussian Process Regression and Joint Stiffness Model. *Journal of Machine Engineering*, 06 2023. doi: 10.36897/jme/167359.

[29] Fanuc, Ltd. Controller R-30iB Plus, 2023. URL `https://www.fanuc.eu/it/en/robots/accessories/robot-controller-and-connectivity`. Retrieved: 2023-08-27.

[30] *SYSTEM R-30iA, R-30iB, and R-30iB Plus Controller, KAREL Reference Manual, MARRC75KR07091E Rev J.* FANUC America Corporation, 3900 W. Hamlin Road, Rochester Hills, Michigan, 2017. Reference Manual.

[31] Wael Hojak. Entwicklung einer echtzeitfähigen datenschnittstelle sowie einer darauf aufbauenden kamerabasierten echtzeitregelung für industrieroboter. Master's thesis, Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, 3 2023. EN: "Development of a real-time capable data interface and a camera-based real-time control for industrial robots based on it".

[32] *FANUC Robot series: R-30iB and R-30iB Plus controller, Dynamic Path Modification, MARUBDPMO02141E.* FANUC America Corporation, 3900 W. Hamlin Road, Rochester Hills, Michigan, 11 2021. User's Guide.

[33] Wael Hojak and Julian Markus Alexander Blumberg. FaRoC. `https://git.tu-berlin.de/waelhojak/FaRoC`, 2023.

[34] Agajan Torayev, Giovanna Martínez-Arellano, Jack C Chaplin, David Sanderson, and Svetan Ratchev. Towards Modular and Plug-and-Produce Manufacturing Apps. *Procedia CIRP*, 107:1257–1262, 2022.

[35] G.A. vd. Hoorn. Fanuc DPM Mouse Demo. `https://github.com/gavanderhoorn/fanuc_dpm_mouse_demo`, 2020.

[36] Fanuc, Ltd. Simulation Software Roboguide, 2023. URL `https://www.fanuc.eu/it/en/robots/accessories/roboguide`. Retrieved: 2023-09-12.

[37] Advanced Realtime Tracking GmbH & Co. KG. ARTTRACK5, 2023. URL `https://ar-tracking.com/en/product-program/arttrack5`. Retrieved: 2023-08-28.

[38] Advanced Realtime Tracking GmbH & Co. KG. DTrack, 2023. URL `https://ar-tracking.com/en/product-program/dtrack`. Retrieved: 2023-08-28.

[39] Rolf Isermann and Marco Münchhof. *Identification of Dynamic Systems.* Springer Berlin, Heidelberg, 11 2010. ISBN 978-3-540-78879-9. doi: 10.1007/978-3-540-78879-9.

[40] Ying Bai Bai and Dali Wang. Fundamentals of Fuzzy Logic Control – Fuzzy Sets, Fuzzy Rules and Defuzzifications. *Advances in Industrial Control*, 01 2007. doi: 10.1007/978-1-84628-469-4_2.

[41] Lotfi A. Zadeh. Fuzzy sets. *Information and control*, 8.3, 1965. doi: 10.1016/S0019-9958(65)90241-X.

[42] MathWorks. Foundations of Fuzzy Logic, 2023. URL `https://it.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html`.

[43] Li Bo, Li Yufei, Tian Wei, and Liao Wenhe. Pose accuracy improvement in robotic machining by visually-guided method and experimental investigation. *Robotics and Autonomous Systems*, 2023. doi: 10.1016/j.robot.2023.104416.

[44] N.G. Adar, A. Egrisogut Tiryaki, and R. Kozan. Real Time Visual Servoing of a 6-DOF Robotic Arm using Fuzzy-PID Controller. *Special issue of the International Conference on Computational and Experimental Science and Engineering*, 128, 2015. doi: 10.12693/APhysPolA.128.B-348.

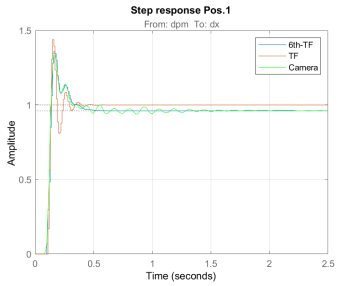
[45] Dino Dominic Ligutan, Levin Jaeron S. Cruz, Michael Carlo D. P. Del Rosario, Jho Nathan Singh Kudhal, Alexander C. Abad, and Elmer P. Dadios. Design and Implementation of a Fuzzy Logic-based Joint Controller on a 6-DOF Robot Arm with Machine Vision Feedback. *Computing Conference*, 2017. doi: 10.1109/SAI.2017.8252111.

[46] Greg Welch and Gary Bishop. An introduction to the kalman filter. *Proc. Siggraph Course*, 8, 01 2006.

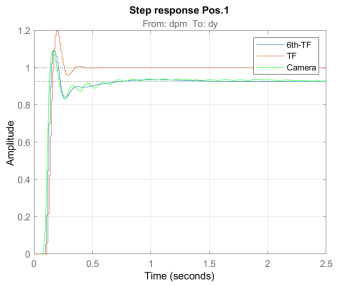
[47] Thomas Powell. Automated Tuning of an Extended Kalman Filter Using the Downhill Simplex Algorithm. *Journal of Guidance Control and Dynamics*, 25, 09 2002. doi: 10.2514/2.4983.

A | Appendix A

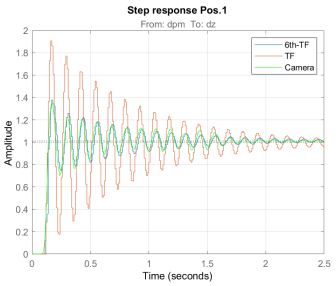
Graphics of the step responses on the x, y, and z axes for all the positions.



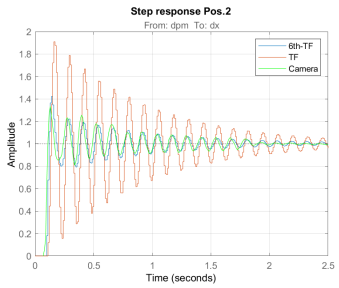
(a) Pos.1 x-axis



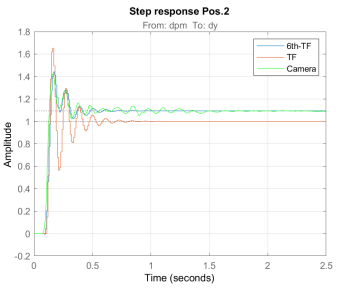
(b) Pos.1 y-axis



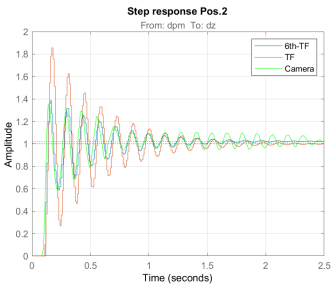
(c) Pos.1 z-axis



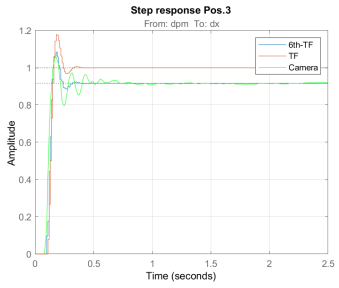
(d) Pos.2 x-axis



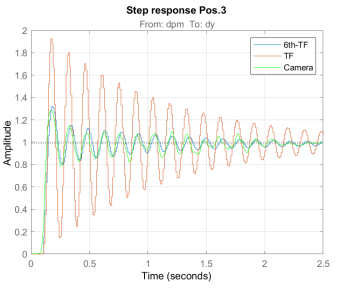
(e) Pos.2 y-axis



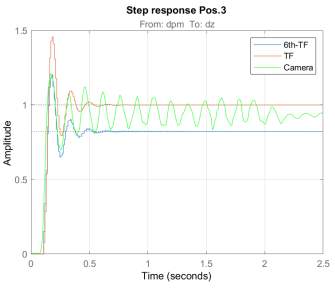
(f) Pos.2 z-axis



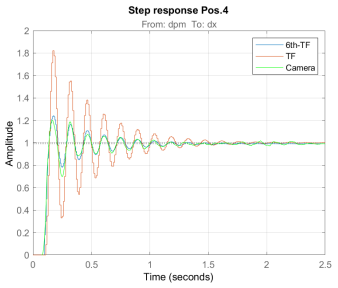
(g) Pos.3 x-axis



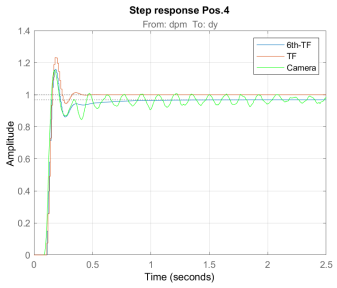
(h) Pos.3 y-axis



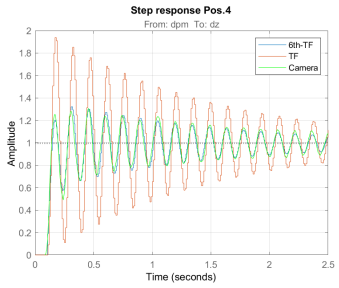
(i) Pos.3 z-axis



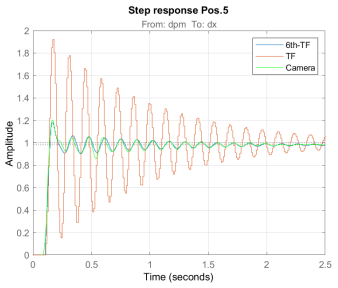
(j) Pos.4 x-axis



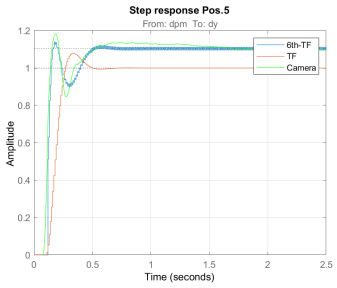
(k) Pos.4 y-axis



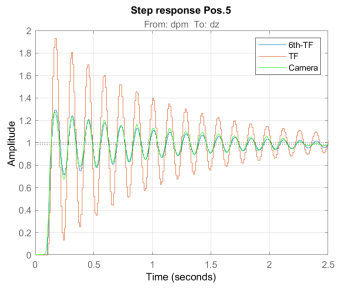
(l) Pos.4 z-axis



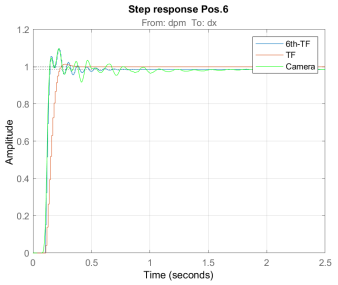
(m) Pos.5 x-axis



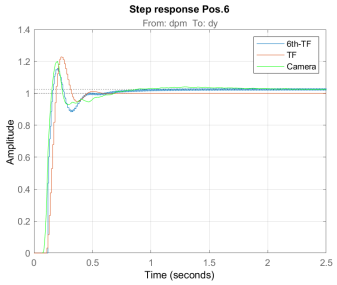
(n) Pos.5 y-axis



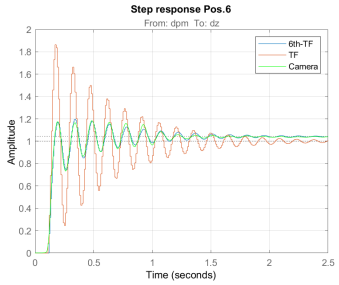
(o) Pos.5 z-axis



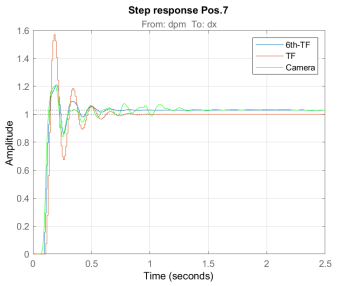
(p) Pos.6 x-axis



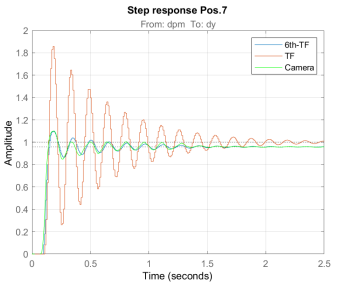
(q) Pos.6 y-axis



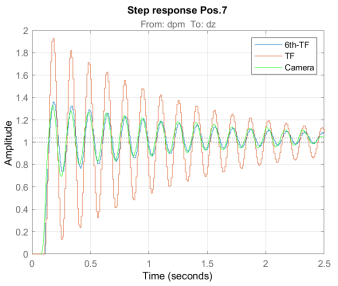
(r) Pos.6 z-axis



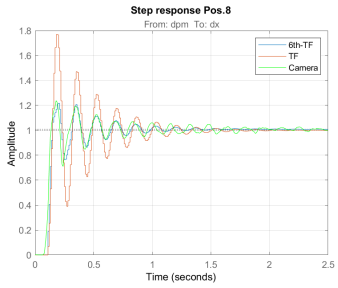
(s) Pos.7 x-axis



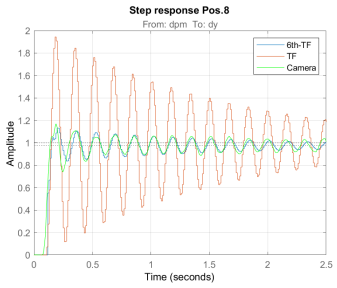
(t) Pos.7 y-axis



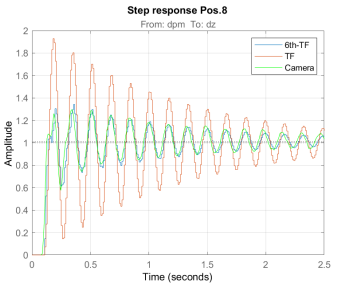
(u) Pos.7 z-axis



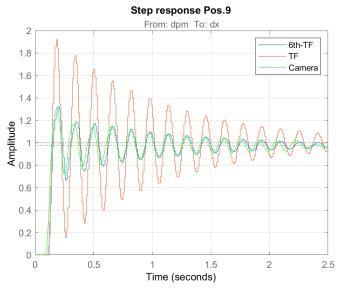
(a) Pos.8 x-axis



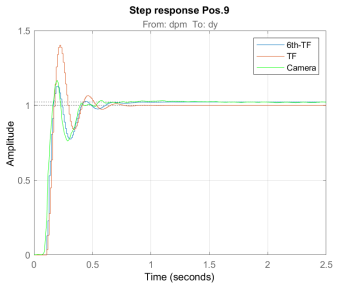
(b) Pos.8 y-axis



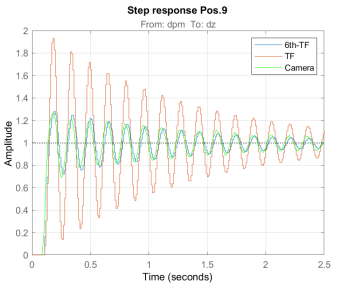
(c) Pos.8 z-axis



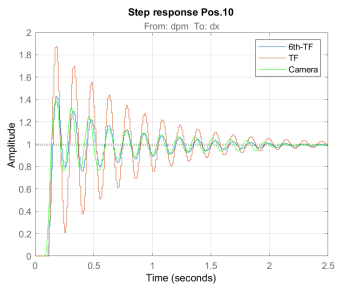
(d) Pos.9 x-axis



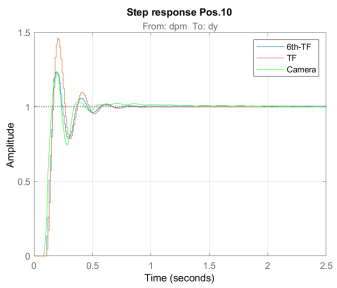
(e) Pos.9 y-axis



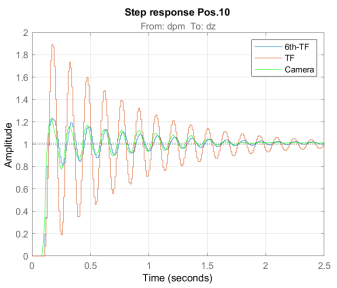
(f) Pos.9 z-axis



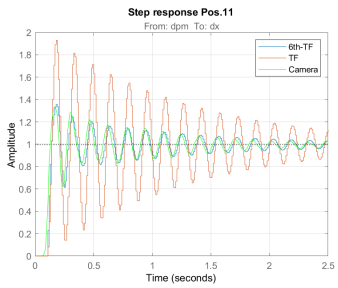
(g) Pos.10 x-axis



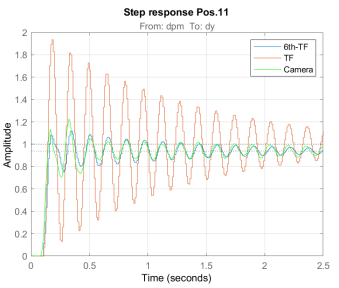
(h) Pos.10 y-axis



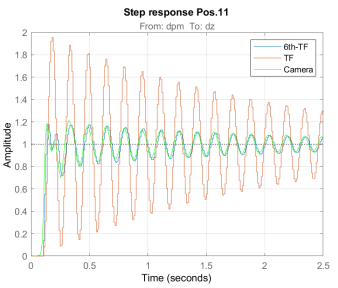
(i) Pos.10 z-axis



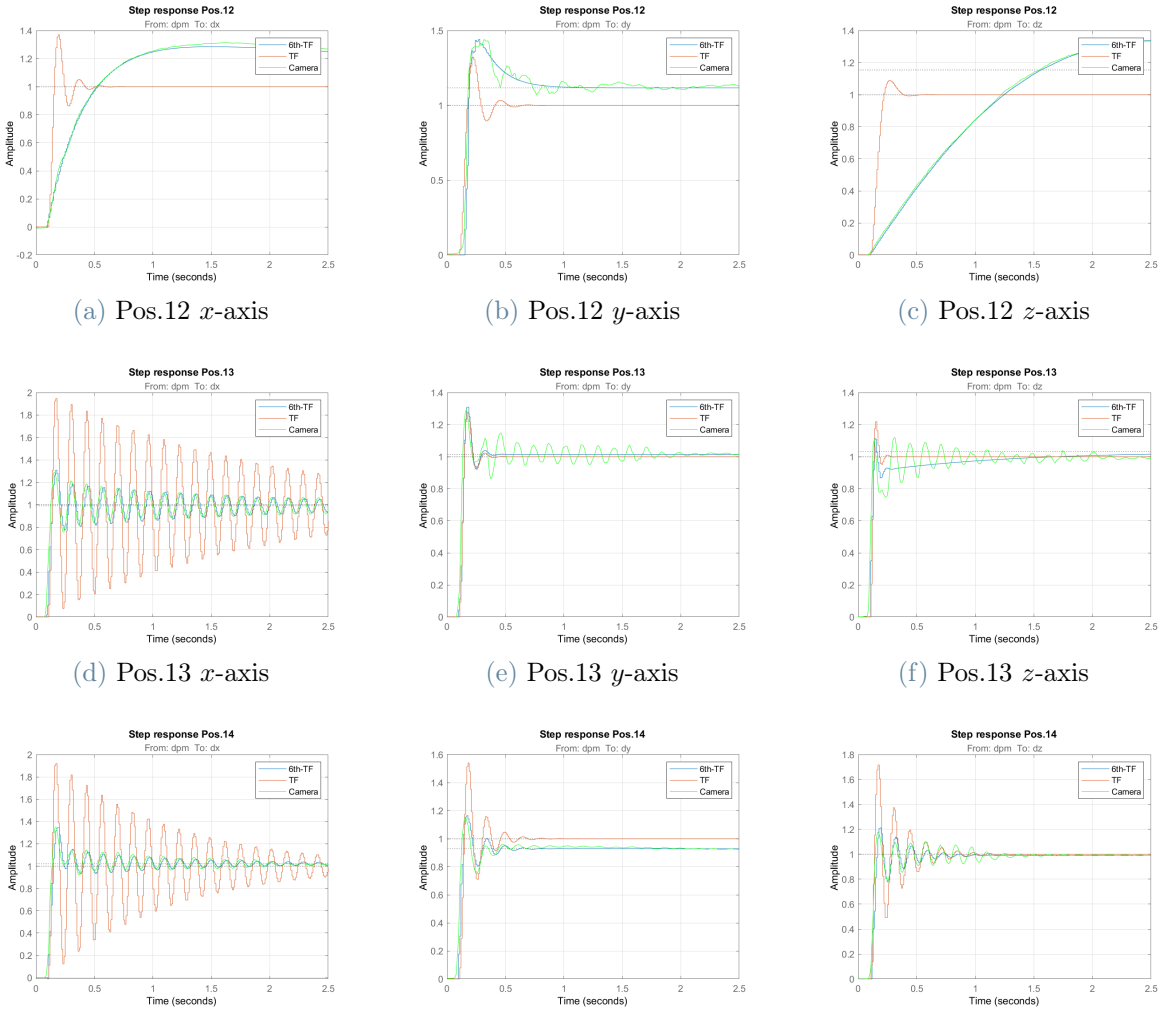
(j) Pos.11 x-axis



(k) Pos.11 y-axis



(l) Pos.11 z-axis



(a) Pos.12 x-axis (b) Pos.12 y-axis (c) Pos.12 z-axis

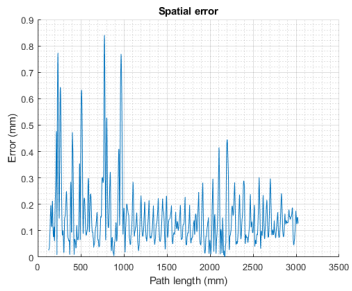
(d) Pos.13 x-axis (e) Pos.13 y-axis (f) Pos.13 z-axis

(g) Pos.14 x-axis (h) Pos.14 y-axis (i) Pos.14 z-axis

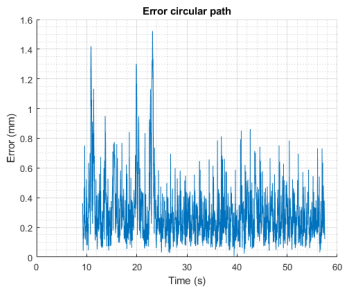
Figure A.1: Step responses on the x, y, and z axes for all the positions

B | Appendix B

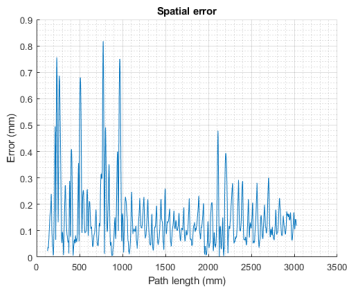
Graphics of the measured errors for different Kalman filters and different values of CNT.



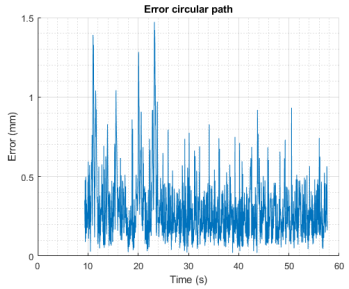
(a) Circle slow AKF



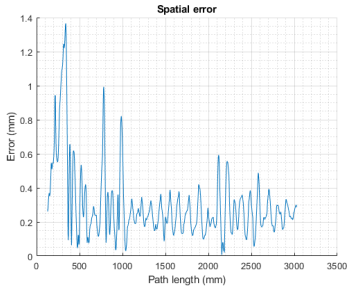
(b) Circle slow AKF



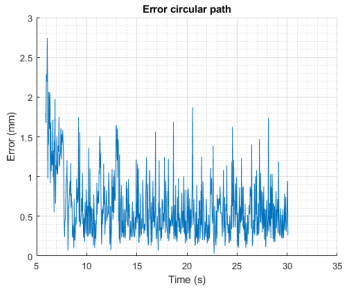
(c) Circle slow EKF



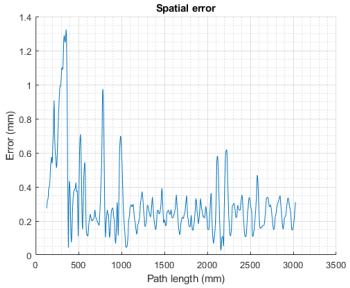
(d) Circle slow EKF



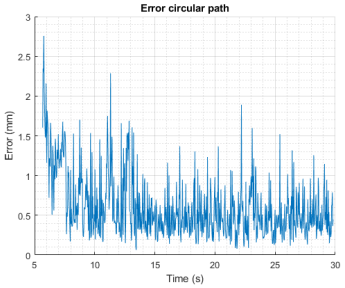
(e) Circle fast AKF



(f) Circle fast AKF

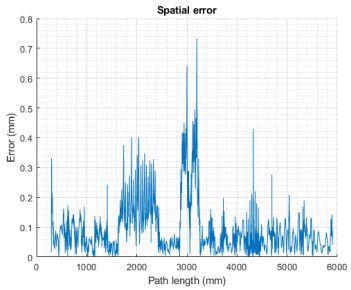


(g) Circle fast EKF

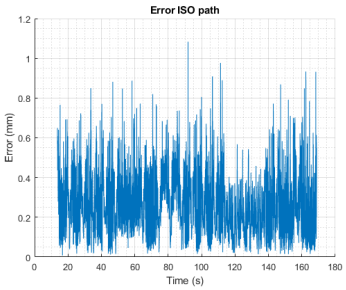


(h) Circle fast EKF

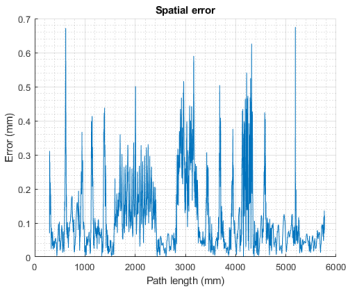
Figure B.1: Kalman's comparison for circular path



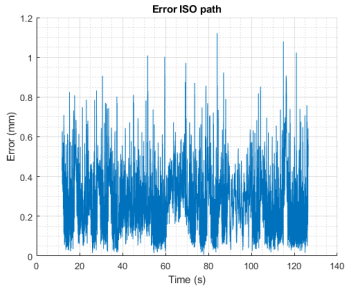
(i) ISO slow AKF



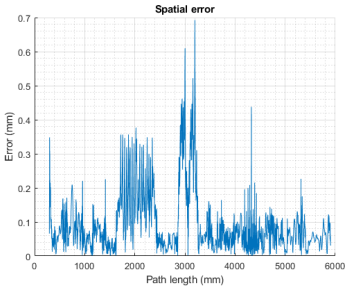
(j) ISO slow AKF



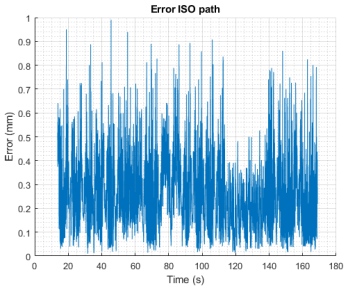
(k) ISO slow AKF cnt 50



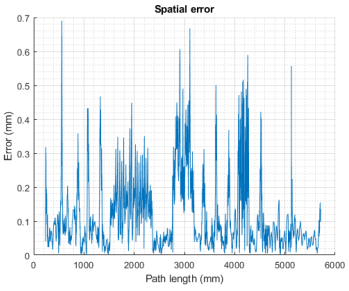
(l) ISO slow AKF cnt 50



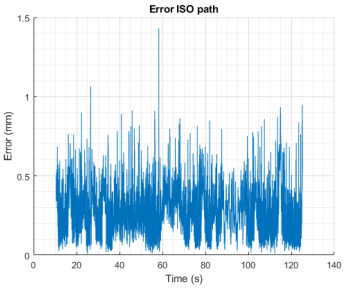
(m) ISO slow EKF



(n) ISO slow EKF

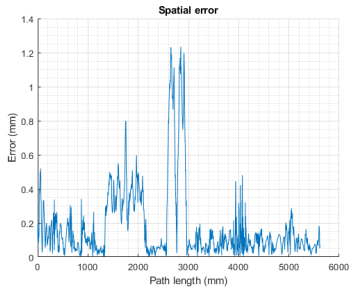


(o) ISO slow EKF cnt 50

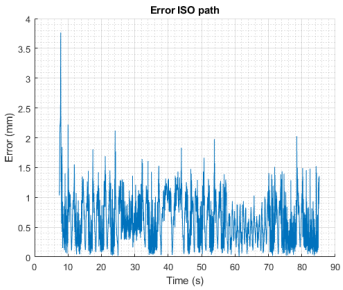


(p) ISO slow EKF cnt 50

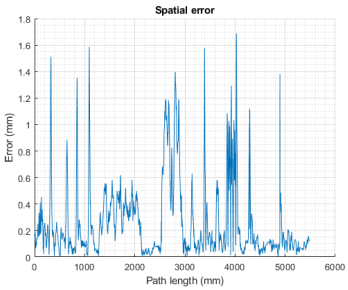
Figure B.1: Kalman's comparison for ISO at low speed



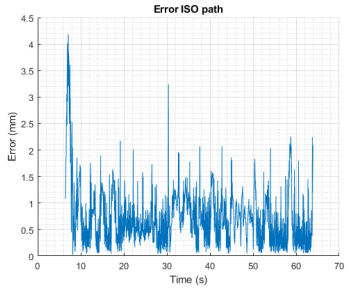
(q) ISO fast AKF



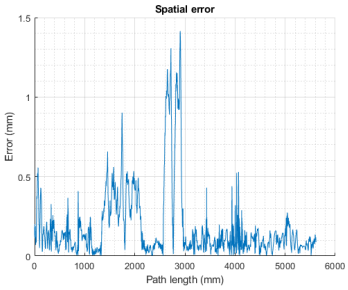
(r) ISO fast AKF



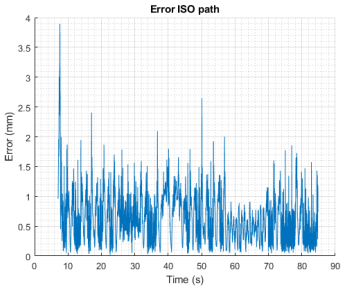
(s) ISO fast AKF cnt 50



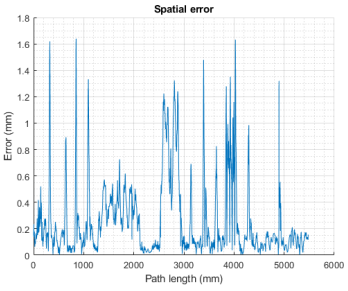
(t) ISO fast AKF cnt 50



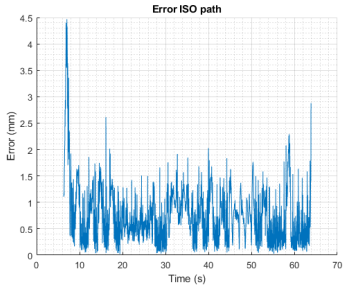
(u) ISO fast EKF



(v) ISO fast EKF

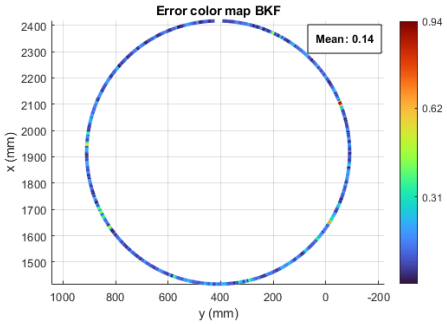


(w) ISO fast EKF cnt 50

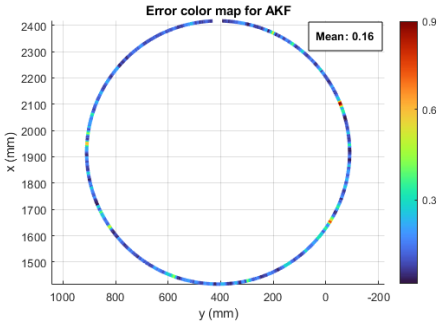


(x) ISO fast EKF cnt 50

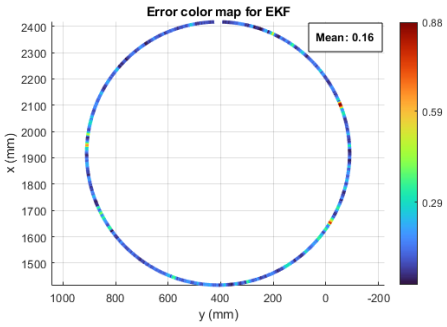
Figure B.1: Kalman's comparison for ISO at high speed



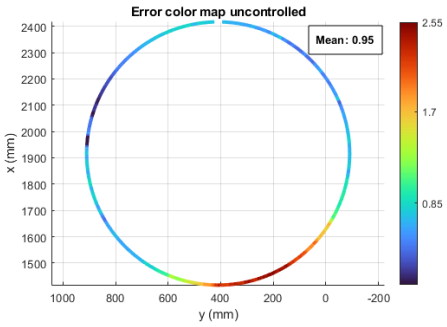
(a) BKF



(b) AKF

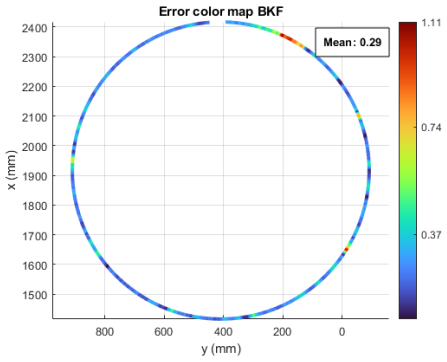


(c) EKF

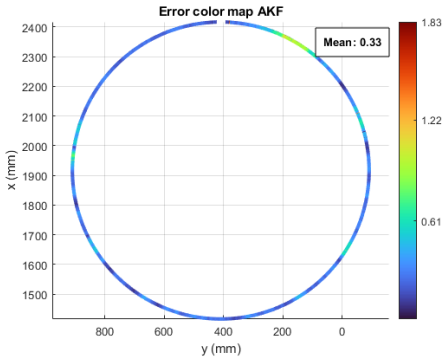


(d) Uncontrolled

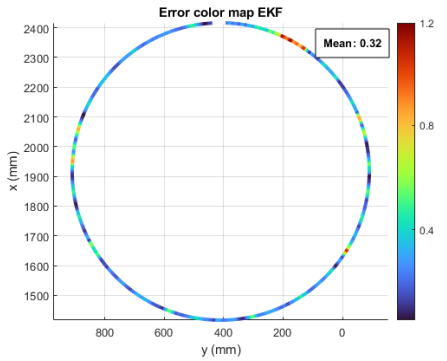
Figure B.2: Color map Kalman's comparison for circular path at slow speed



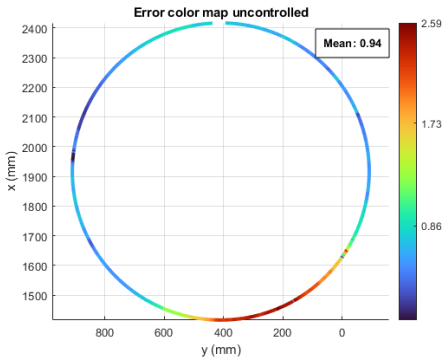
(e) BKF



(f) AKF

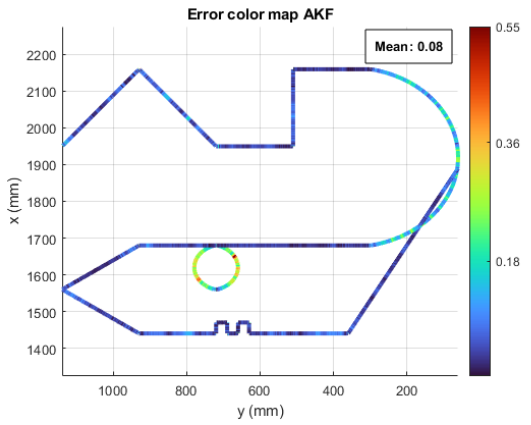


(g) EKF

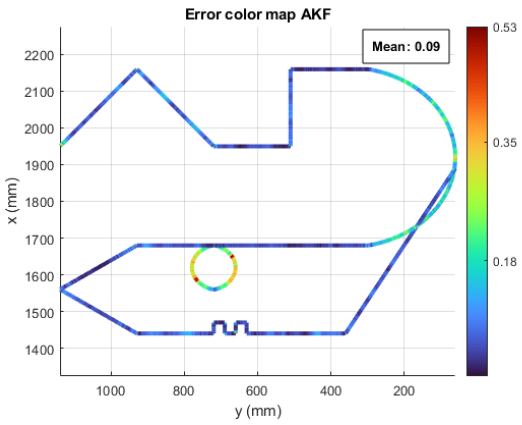


(h) Uncontrolled

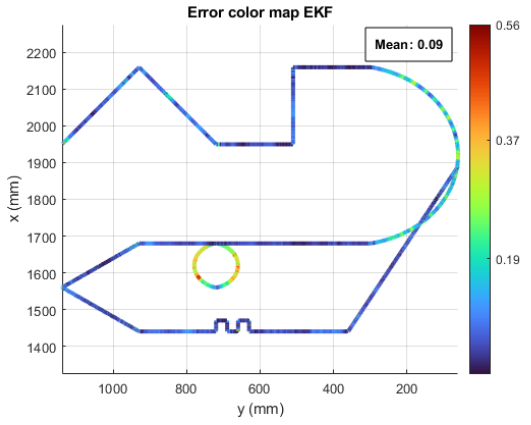
Figure B.2: Color map Kalman's comparison for circular path at high speed



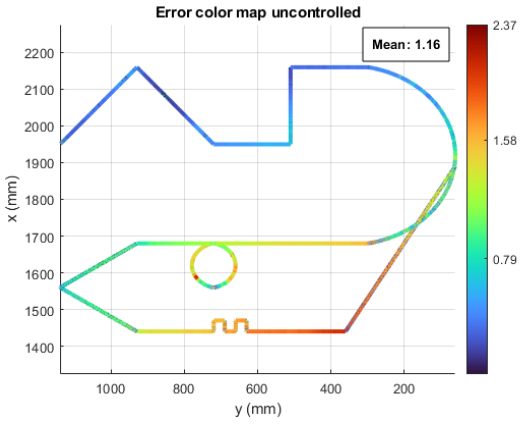
(i) BKF



(j) AKF



(k) EKF



(l) Uncontrolled

Figure B.2: Color map Kalman's comparison for ISO path at slow speed