Dual Artificial Potential Field (d-APF) Based Control for Safe and Efficient Robotic Operation on Industrial Collaborative Scenarios

Diego Rodríguez Guerra

October 2023
Version: First Version

University of the Basque Country (UPV/EHU)

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Department of Automatic Control and Systems Engineering
Virtual Sensorization Research Group

Documentation

Dual Artificial Potential Field (d-APF) Based Control for Safe and Efficient Robotic Operation on Industrial Collaborative Scenarios

Diego Rodríguez Guerra

1. Supervisor Itziar Cabanes Axpe
Department of Automatic Control and Systems Engineering
University of the Basque Country

2. Supervisor Gorka Sorrosal Yarritu
Department of Intelligent Control (KAU)
IKERLAN S. COOP

October 2023

Diego Rodríguez Guerra
Dual Artificial Potential Field (d-APF) Based Control for Safe and Efficient Robotic Operation on Industrial Collaborative Scenarios
Documentation, October 2023
Supervisors: Itziar Cabanes Axpe and Gorka Sorrosal Yarritu

University of the Basque Country (UPV/EHU)
Virtual Sensorization Research Group
Department of Automatic Control and Systems Engineering
Plaza Ingeniero Torres Quevedo, 1
48013 , Bilbao (Bizkaia)

Disclaimer
You can edit this page to suit your needs. For instance, here we have a no copyright statement, a colophon and some other information. This page is based on the corresponding page of Ken Arroyo Ohori's thesis, with minimal changes.

Colophon
This document was typeset with the help of KOMA-Script and LaTeX using the kaobook class.

The source code of this book is available at:
https://github.com/fmarotta/kaobook

(You are welcome to contribute!)

Publisher
First printed in July 2023 by publishers

Abstract

This Ph.D. proposes two main contributions for the safe and efficient collaborative manufacturing on modern industrial environments. On the one hand, a novel d-APF controller is proposed to avoid simultaneously singularities and collisions while operating in a shared environment. This contribution leans on the second novelty proposed in this Ph.D., a decoupled kinematic model for non-spherical wrist cobot. Without the last contribution or any similar singularity characterization, the d-APF controller cannot be implemented due to the lack of a set of characterized singular configurations.

With the aim of explaining the controller, the initial chapters of this document explain the theoretical fundamentals to implement the novel kinematic model, the reference controller and the proposed one. The main advantage of the decoupled kinematic model proposed relies in the wrist spherification technique to obtain a quasi-spherical wrist, which is a non-spherical wrist that kinematically behaves as if it was spherical. Based on this new kinematic model, a set of closed solutions for the inverse kinematics and the singularities are obtained. Indeed, the joint dependent singularity characterization obtained is the key to implement the proposed d-APF controller, enabling the measure of the closeness to the singularity to compute a repulsive response and a limiting velocity index for each singular joint. In addition to the controller and the kinematic model, the reference or benchmarking controller is also implemented, the DLS-APF controller. This controller is not only utilized as a reference controller that integrates the singularity handling throught a DLS kinematic model, but also sets the basics for the collision avoidance components of the d-APF controller.

Then, the implementation of the model and both controllers is regarded for a UR10e robot and the SUPSI robot. In the case of the UR10e robot, the manipulator behavior for both controllers has been tested for both, a combined Gazebo-MoveIt simulated environment and real demonstrator environment. Complementary, the SUPSI robot has adopted a more supportive role to verify and extend the conclusions obtained from the first robot experimentation to other non-spherical wrist cobots with similar structure. Therefore, only simulation behavioral tests have been executed for this robot. This implementation for both robots relies on a generic software control architecture developed in this Ph.D. that creates a RTT Controller Manager and a hardware interface bridge to integrate ROS Control capabilities into Orocos components. On top of that, a specific RTT ROS Controller is implemented for each robot and each studied controller (the DLS-APF and the d-APF ones), achieving a hardware agnostic real time capable ROS controller.

Subsequently, several tests on the implemented controller have been implemented to test the real time and collaborative application suitability of the proposal. The results shows an improvement of a rough 17% in the computational time to compute a singularity free collision avoidance response with respect to the reference controller. Moreover, the real time encapsulation of the ROS Control capabilities allows sending commands to the robot at frequency rates above 500Hz, enabling the response of the robot to obstacles in very short times. It means that, depending on the application requirements, the determinism of a safe response can be assured favouring the real time capabilities of the robotic control system. Finally, some trajectory execution tests have been executed with and without obstacles, ending with positive results where a reduction of about the 40% of the required execution time have been appreciated compared to a fully manually developed strategy. Therefore, the proposed approach after a thoughtfull analysis has been determined as suitable to improve current manufacturing efficiency on collaborative scenarios (roughly 11% of reduction in time) without disregarding the safety aspects due to the simultaneous collision and singularity avoidance that allows evading collisions in a smoother manner.

Resumen

El presente trabajo de tesis propone dos contribuciones principales para la fabricación colaborativa segura y eficiente en entornos industriales modernos. Por un lado, se propone un novedoso controlador para evitar simultáneamente singularidades y colisiones mientras se opera en un entorno compartido. Esta contribución se apoya en la segunda novedad propuesta, un modelo cinemático desacoplado para robots colaborativos de muñeca no esférica. Sin esta última contribución o una caracterización similar de las singularidades del manipulador, el controlador d-APF propuesto no puede implementarse debido a la falta de un conjunto cerrado de configuraciones singulares caracterizadas.

Con el objetivo de explicar el controlador, en los capítulos iniciales de este documento se explican los fundamentos teóricos para implementar el novedoso modelo cinemático, el controlador de referencia y el propuesto. La principal ventaja del nuevo planteamiento para el modelo cinemático desacoplado radica en la técnica de la esferificación de muñeca para obtener una muñeca quasi-esférica. O lo que es lo mismo, una muñeca no esférica que cinemáticamente se comporta como si lo fuera. A partir de este nuevo modelo cinemático, se obtiene un conjunto de soluciones cerradas para la cinemática inversa y las singularidades. De hecho, la caracterización de la singularidad dependiente de la articulación obtenida es la clave para implementar el controlador d-APF propuesto, permitiendo la medida de la cercanía a la singularidad para calcular una respuesta repulsiva y un índice de velocidad límite para cada articulación. Además del controlador y el modelo cinemático, también se ha detallado el controlador de referencia o controlador DLS-APF. Este controlador no sólo se utiliza como referencia sino que integra el manejo de la singularidad a través del modelo cinemático DLS y también establece las bases para los componentes de evasión de colisiones del controlador propuesto. Para validar la solución propuesta se considera la implementación del modelo y ambos controladores para un robot UR10e y el robot SUPSI. En el caso del robot UR10e, se ha evaluado el comportamiento del manipulador para ambos controladores tanto en un entorno combinado de Gazebo y MoveIt en simulacion, como en un demostrador dentro de un entorno controlado real. Para apoyar y extender los resultados obtenidos en el anterior robot a otros robots de estructuras de muñeca no esférica semejantes, se han probado tan sólo en simulación ambos controladores para el robot SUPSI. Esta implementación en ambos robots se basa en una arquitectura de control de software genérica desarrollada en esta tesis doctoral que crea un RTT Controller Manager y un hardware interface bridge para integrar las capacidades de ROS Control en los componentes de Orocos. Además de eso, un RTT ROS Controller específico es implementado para cada robot y caso de estudio (los controladores DLS-APF - Damped Least Square y Artificial Potential Field - y d-APF - dual Artificial Potential Field).

Posteriormente, para comprobar la idoneidad de la propuesta en tiempo real y en aplicaciones colaborativas se han realizado ensayos en simulación y en entorno real. Los resultados muestran una mejora aproximada del 17% en el tiempo computacional para calcular una respuesta de evasión de colisiones libre de singularidades, con respecto al controlador de referencia. Además, la encapsulación de las capacidades de ROS Control en tiempo real permite enviar comandos con frecuencias superiores a los 500Hz, posibilitando la respuesta a los obstáculos en periodos cortos de tiempo. Esto quiere decir que, dependiendo de los requisitos de la aplicación, se puede garantizar el determinismo y capacidades de tiempo real del sistema de control del robot. Finalmente, se han efectuado algunas pruebas de ejecución de trayectorias con y sin obstáculos, finalizando con resultados positivos donde se han apreciado hasta un 40% de reducción en los tiempos requeridos para ejecutar de forma semi-automatizada frente al proceso de fabricación totalmente manual. Asímismo, en comparación con otros controladores para la gestión de singularidades y obstáculos, la mejora en eficiencia se corresponde con un 11% de ahorro en tiempos para ejecutar el proceso de fabricación gracias al control propuesto en la presente tesis doctoral. Esto es indicativo de que se consigue mantener la seguridad del operario en todo momento, alcanzando un proceso más fluido de fabricación debido a la evasión simultánea de colisiones y singularidades.

Acknowledgement

I would like to give a big shout-out to everyone that had helped me overcomming all the difficulties found during the Ph.D. bringing me their support. Firstly, I would like to thank my family (specially to my dad Miguel, my mum Isabel, and older brother Jorge) and loved ones (in special my girlfriend Ana) for their daily support and love, which had helped me keeping motivated everytime during these three years and a half of Ph.D. thesis.

I cannot forget also to thank Ikerlan S. Coop for the financial support given to develop my Ph.D. under their guidance. Specially, I would like to thank Gorka Sorrosal, my supervisor from there who has suffered me the most. Luckily, I have learned lot of things from him and I consider him not only as a mentor but a friend, too. From Ikerlan too, I would like to thank Karlos Calleja, my back up supervisor, whose had also helped in my apprenticeship. For me, it also has been a real pleasure to have the chance of working alongside and under the guidance of Itziar Cabanes, my supervisor from the UPV/EHU (Euskal Herriko Unibersitatea) whom I also like to thank all her efforts, the technical guidance and emotional support given. I truly believe that without the guidance of them, the results of this research would have been different.

In addition, I really appreciate the welcome received by the SUPSI and their people during my Ph.D. research stay. In this period, not only good results from the research are obtained, but I also made a lot of friends. I would like to give a special shout-out to Alessio Mosca, my personal mentor and friend there, whom without his help, this work could not be possible.

Lastly, I cannot forget either my closest friends from Álava/Araba and La Rioja who had suffered my many complains and my very worst down in the dumps moments. In special, I really appreciate the help and emotional support of Ignacio Trojaola who is not only a very good friend, but a good mentor in the Ph.D. and life too.

Contents

List of Figures

List of Tables

Motivation, State of the Art, Hypothesis and Methodology

1 Introduction

The current industrial manufacturing environments still present several production tasks and operations that are not automated yet, leading to non-ergonomic manufacturing environments without high demanding quality standards and productivity. On the one hand, some of those tasks are really expensive to automate on a single technological upgrade. On the other hand, other types of processes require a very specific knowledge and expertise of their operations to fully automate the tasks with the desired degree of automation (only achievable by the most experienced operators with the required skills). These two reasons make this technological upgrade unaffordable in several senses, impeding the development of fully autonomous manufacturing systems (also known as dark or lights-out factories).

In this panorama, collaborative robots or cobots have emerged to face the limitations of the current automation and autonomous systems by allowing shared workspaces and tasks with human operators. Thus, operator safety became the first issue to guarantee when a human worker shared the workspace with a robotic manipulator in a secure fence-free environment. In these scenarios where humans and cobots share not only the workspace but also the working task or operation, also known as industrial collaborative scenarios, every aspect of the manufacturing application matters and should be regarded. For instance, if the cobot carries a heavy or sharp load, the load could hurt the operator in case of a collision. Therefore, to avoid non-desired contact situations that might compromise the safety of the operator, not only the robot should be intrinsically safe (collaborative), but the application itself too.

Moreover, non-desired collisions lead to continuous safety stops during collaborative manufacturing processes. This safety behavior, which is indeed required, waste several manufacturing time due to the continuous rearm of the robot arm for the operator. Therefore, avoiding non-desired contacts, or collisions, can also increase the performance of the manufacturing processes by dodging non-required continuous safety stops, making collision avoidance in collaborative scenarios a widely studied topic. However, it is relevant to remember that to achieve a control technique that allows the real time response to a human operator, the control algorithms selected should avoid heavy demanding computational workloads with continuous re-planning of the robot trajectory to seek for a new safe (collision-free) path. Among these dynamic real time response control algorithms, the Artificial Potential Field (APF) based controller arises as a suitable tool to implement safe and efficient collaborative manufacturing processes.

Nevertheless, these dynamic response control algorithms present the major drawback of disregarding the configurations where the robot kinematics is ill-conditioned while avoiding the collision. So whenever an operator is being avoided, there is a chance of activating the safety emergency systems of the robot due to excessive torque control signals. In this situation, the robot is stopped, and the risks for the operator

might increase. Similarly, the efficiency of the manufacturing process is decreased due to the required rearm after the safety stop.

Therefore, this first chapter of this Ph.D. thesis in collaborative robotics pretends to clarify the challenges to beat and potential improvements for introducing cobots in current manually executed manufacturing processes. Moreover, it is also explained the structure of the thesis into which the content of each of the chapters has been divided, as well as the subject matter of each of the chapters and the topics they will address.

1.1 Motivation

In the introduction of this chapter, cobots are presented as a suitable tool to address some of the limitations of current manual manufacturing tasks and operations [1]. On the one hand, thanks to the semi-automated processes approach, adequate successive technological partial upgrades can be executed instead of fully automating an industrial manufacturing process at once. On the other hand, their integration allows the automation of repetitive operations to keep manufacturing quality standards while the operator brings flexibility through its experience and cognitive processes [2, 3]. However, the aforementioned advantages of shared manufacturing environments must deal with some areas for improvement to properly implement these scenarios at industrial processes. These disadvantages are linked to two main aspects: the safety of the operator and adequate time management [4].

Therefore, whenever a human worker is integrated into an automated manufacturing system surrounded by robotic systems without any protection fence, its safety must be guaranteed [5]. This aspect is partially achieved through industrial collaborative robots, which are indeed intrinsically safe designed robotic manipulators [6]. Nevertheless, just employing collaborative robots is not enough to guarantee operators' safety. Other aspects should be regarded, such as which tool is installed in the manipulator or which load it should operate with [7]. I.e., shearing or dangerous tools or loads (e.g. a welding actuator) require additional safety measures to make the whole application collaborative [8]. Otherwise, they cannot be considered collaborative applications and would require the installation of safety fences or any other similar technology to avoid the risks of sharing spaces [9, 10].

Moreover, regarding adequate time management, the first issue that comes to mind consists of the limited working speeds and forces of cobots [11]. Thus, whenever a collaborative application is designed to be coordinated with other autonomous manufacturing processes, it should be borne in mind that the cycle time of the collaborative subprocesses will be higher [12]. Additionally, the primary safety function of cobots to ensure the safety stops, which generates elevated time wasted due to safety reasons. So the safety bottlenecks should be regarded too [13, 14]. Likewise, robot movements and the operator itinerary must be scheduled collaboratively to avoid excessive production bottlenecks. Thus proper management of the task sequence is required. However, the randomness in human actions makes it hard to model operator behavior into control loops or automation systems [15–17].

In this manner, achieving a plausible industrial collaborative scenario where the production is safe while operations are adequately distributed to avoid excessive time wasted due to safety stops requires the combination of collaborative robots with advanced automation or control systems. Therefore, this work aims to develop an advanced control system that allows safe and collaborative manufacturing without disregarding the timing aspects that could generate relevant production bottlenecks. Thus, real industrial collaborative scenarios can be implemented on existing manufacturing shop floors that could synchronize themselves with the autonomous manufacturing tasks of a complete industrial production process.

1.2 Thesis structure

The current Ph.D. thesis is organized into nine different chapters. Each of these chapter addresses the following topics:

Chapter 1, as seen along this chapter, is dedicated to display the need for the Ph.D., exposing a preview on the State of the Art and the Motivation of the doctorate. It also addressed briefly the content of each chapter and what can be expected to be found inside each chapter.

Chapter 2 dives deeper into the state of current industrial collaborative scenario through a detailed State of the Art on the topic. This study leads then to present the hypothesis and objectives that are addressed in the research.

Chapter 3 describes the main aspects about the selected research methodology, data collection, and analysis techniques that better suits the objective of the thesis. Moreover, it is also presented the different scenarios and validation environments for both, simulated and real laboratories tests.

Chapter 4 begins with a little preamble of how and why classic kinematic models of non-spherical wrist cobots are not optimal for modern collision avoidance reactive algorithms. In order to address the presented limitations, a new approach to model non-spherical wrist cobots is described in this chapter. The proposed kinematic model allows the computation of a simplified forward and inverse kinematics for this type of robots for both position and velocity. Therefore, a singularity theoretical study and the implications of the usage of the proposed model is also included in this chapter.

Chapter 5 describes the reference controller and the modification done with respect to original version of the control algorithm from [18]. This reference controller introduces the Damped Least Square (DLS) kineamtic model to handle the singularity while avoiding obstacles, and a new library to handle more efficiently the dynamics of the robot. These two features complements the behavior of the obstacle avoidance based on an Artificial Potential Field (APF) based controller.

Chapter 6 deals with the design of the novel controller for simultaneous collision and singularity avoidance, the dual Artificial Potential Field (d-APF) controller proposed in this Ph.D. The aspects detailed in this chapter are related to the specific design of the controller to apply the

characterized closed set of singularities found by the application of the methodology displayed in Chapter 4.

Chapter 7 exposes the considerations and key points of the implementation in both controllers, from Chapter 5 and Chapter 6, for a UR10e robot and the available deliberative manipulator at SUPSI (Scuola Universitaria Professionale della Svizzera Italiana, where took place the research stay period). The implementation aspects displayed in this chapter are related to a simulation environment based on MoveIt and Gazebo for both robots, and also to a real robot implementation in the case of the UR10e.

Chapter 8 displays the different results for several tests executed in simulation and for the real robots. These tests address different aspects as timing performance of the controller, required time to compute a new singularity and collision free control signal, and the performance of tracking a trajectory for regular scenarios and the ones where the collision is avoided. All the results are presented for both robot models. A conclusion on the obtained results is also presented in this chapter.

Chapter 9 ends this document showing the conclussion obtained of the research as well as the developed research outcomes. Finally, it also indicates possible lines for future research on the field.

State of the Art and Objectives 2

This chapter aims to set the basics for the research objectives and goals. To do so, it starts with a detailed State of the Art where the problematic of non-spherical wrist cobots when working with reactive collision avoidance strategies for industrial collaborative scenarios is shown. Then the hypothesis to improve the current State of the Art through the developments of this Ph.D. is presented, leading to specific objectives to address in order to accomplish the proposed goals.

2.1 State of the Art

2.1.1 Introduction

Current industrial manufacturing environments are characterized for an increasing demand on flexible and autonomous processes that can adapt to both unexpected or different manufacturing scenarios, and failures in the machines to keep producing [1]. On the one hand, autonomous manufacturing environments have shown a great performance and improvement in manufacturing quality thanks to automatically adapt to manufacturing standards (i.e. Iterative Learning Controllers - ILC [19]), or adapt to failures to keep production unstopped (i.e. Fault Tolerant Controllers - FTC [20, 21]). On the contrary, achieving the desired level of flexibility to adapt the manufacturing production cycle to unexpected situations is still limited [22].

To overcome these limitations, modern manufacturing environment requires of the combined efforts of human operators with robots to keep up with the flexibility without disregarding manufacturing quality [23]. However, due to the dangers and risks of traditional robot manipulators this type of relation is hardly achievable because the safety cannot be guaranteed for the operators. This panorama, leads to a novel type of robot manipulators that are intrinsically safe designed to allow the human operator share spaces and collaborate with robots, the collaborative robots or cobots [24].

Moreover, the new type of robotic manipulators comes along with a new industrial manufacturing paradigm, the semi-automation of processes [25]. The semi-automation of processes consists of combining human skills and robotic capabilities in manufacturing by partially automating tasks of a collaborative work cell, so the human will develop some operations and the remaining will be done by the robot [2]. This working philosophy also benefits current manufacturing industrial environments where some tasks cannot be yet fully automated in a single step due to economic or technical reasons [26]. In this manner, instead of automating directly an industrial task once, their are planned successive technical upgrades until the desired level of automation is achieved [27]. Thus, not only the technical complexity is reduced on each automation gap covered but also the required financial support to do them are reduced [1].

Despite the benefits the semi-automation and cobots bring to industrial manufacturing environment, there are still aspects to be regarded to achieved the desired smooth flexible industrial manufacturing [2]. Firstly, their intrinsically safe design generates production bottlenecks due to excessive non-critical safety stops due to unexpected collisions between human and operators [28]. In addition, a collaborative application cannot be considered safe if any component involves any risk for the human operator. In other words, if the cobot carries heavy, sharp or pointy objects that might endanger the safety of the operator, the application cannot be considered a collaborative one, so additional safety measures must be installed [7, 9]. Lastly, it is also relevant to bear in mind how the task are distributed among the human operator and the robot in order not to disturb between each other and create additional risky situations [29, 30]. An optimal distribution between human and robots of the operations can also increase the productivity avoiding collisions through a right scheduling technique [26]. The methods proposed in the literature to face this issues are further addressed in Subsection 2.1.2, where five challenges are proposed by this work in [31] to enable real industrial collaborative scenarios.

From all the different types of approaches and challenges to enable the industrial collaborative scenarios of [31], the collision avoidance techniques respond to the necessity of guaranteeing operator safety when they can move freely on a shared fence-free environment [32]. These techniques do not only implies increasing the safety level for operators due to the avoidance of potential risky situations, but they can also improve manufacturing efficiency on these scenarios [33]. Because of this dual benefit from avoiding collision, it has become one of the most commonly explored topic of HRC [18]. However, there are two main aspects to bear in mind while developing advanced control algorithms: having a reliable environment modelling and utilizing a light weighted control algorithm.

Having a reliable environment modelling is required to know information about the robot surroundings in order to interact properly with each of the detected elements [34]. In collaborative robotics, these interactions are not just limited to picking and placing objects or recognizing their relative positioning, recognizing the human position or intention is also required to allow a safe human-robot interaction [35, 36]. Therefore, 3D recognition systems or visions systems gain relevance in this environments to allow the safe interaction between the robot and each element of its surrounding. From among them, it is remarkable that the ones that are the most commonly used are the vision based algorihtms due to the quantity and great versatility of information that they give [28].

In this way, the vision algorithms should work together with advanced light weighted control algorithms for allowing safe human-robot shared manufacturing spaces. Some of the most commonly used control algorithms such as the Artificial Potential Field (APF) based controllers strongly relies in the kinematic model of the robot. However, some of the common commercial collaborative manipulators present a non-spherical wrist structure [37–40]. This type of robot structure makes more complicated to find optimal solutions to the Inverse Kinematic (IK) of these robotics manipulators, impeding the computation of a closed set of solutions for the IK nor the singular configurations. Therefore, the

performance of some types of advanced control algorithms for robotics can be reduced due to these new robotic structures that brings with some types of collaborative robots [41].

The following subsections will cover in detail each of the mentioned aspects during the introduction. Subsection 2.1.2 addresses the current challenges in industrial collaborative scenarios to allow the safe human interaction at all levels of a manufacturing environment. In addition, Subsection 2.1.3 exposes the different vision based techniques to recognize and segment the human in these shared environments from different types of pictures formats. Then, several control algorithms these vision based techniques can be coordinated to are going to be reviewed in Subsection 2.1.4. In addition, this section will cover up also the most commonly techniques to avoid colliding with the operator. Then, the main limitations of the kinematics model due to a non-spherical wrist robot structure are exposed in Subsection 2.1.5. Lastly, a brief conclussion is included that reviews the main points of the exposed information in this State of the Art.

2.1.2 Challenges of current collaborative scenarios

As stated in the introduction, the collaborative robotics was born to face the production flexibility limitations of traditional manufacturing plants [22]. The reason behind the success of the aforementioned improvement for autonomous systems corresponds to the fact that they are generally applied on already working industrial automations [20]. However, there are still several examples of manufacturing processes that cannot be fully automated [26], so no autonomy can be applied. This processes generally coincide with intricate manual manufacturing processes where their fully automation is complex due to technical or economic reasons [2, 25].

Thanks to cobots some fully manual non-automated processes can be partially automated. The usage of cobots on industrial manufacturing processes combines the advantages of both: automated and hand-made processes. On the one hand, as the cobot is an intrinsically safe designed robot that allows sharing spaces with human workers, novel techniques to achieve an autonomous manufacturing can be applied without risking human safety. On the other hand, the fact that an operator is continuously helped by a robot assistant (as a partner), allows the desired level of flexibility through the added value of human cognitive processes to manufacturing cycles [1, 42, 43]. In this manner, the proposed combination of autonomous and manually-driven operations achieves the desired level of flexibility while maintaining the automation implementation and operation costs under the desired profitable thresholds [44].

The cornerstone to allow these shared manufacturing environment (also known as industrial collaborative scenarios) between workers and robots on current industrial plants is the intrinsically safe design of industrial cobots. Nevertheless as shown in Figure 2.1, the mechano-electrical design of the robot from the first level of the pyramid is just the first step to achieve the industrial collaborative scenarios on industrial processes. The industrial collaborative scenarios also demands for safety during any type of shared operation, safe work cell task schedulling, and safe coordination with other autonomous or automated industrial systems and

machines. In this manner, the enabling challenges to achieve the industrial collaborative scenarios can be divided into four levels of achievement according to their closeness to the robot design and their integration with modern industrial processes (Task Design Level, Operation Level, Work cell Level, and Industrial Process Level, from Figure 2.1)[1].

1: A more detailed explanation about this levels and different application examples can be found in the first formal contribution of this Ph.D. thesis [31].

The top and bottom levels, Task Design Level and the Industrial Process Level, correspond to aspects related to safe design of the cobot and its coordination with other components or machines of a modern industrial manufacturing plant, respectively. While the safe design of cobots and the coordination with other parts of the industrial processes through different communication protocols are quite well achieved, there are a number of challenges in the middle levels that need to be solved for the optimal implementation of these scenarios on current industrial manufacturing plants [45–47].

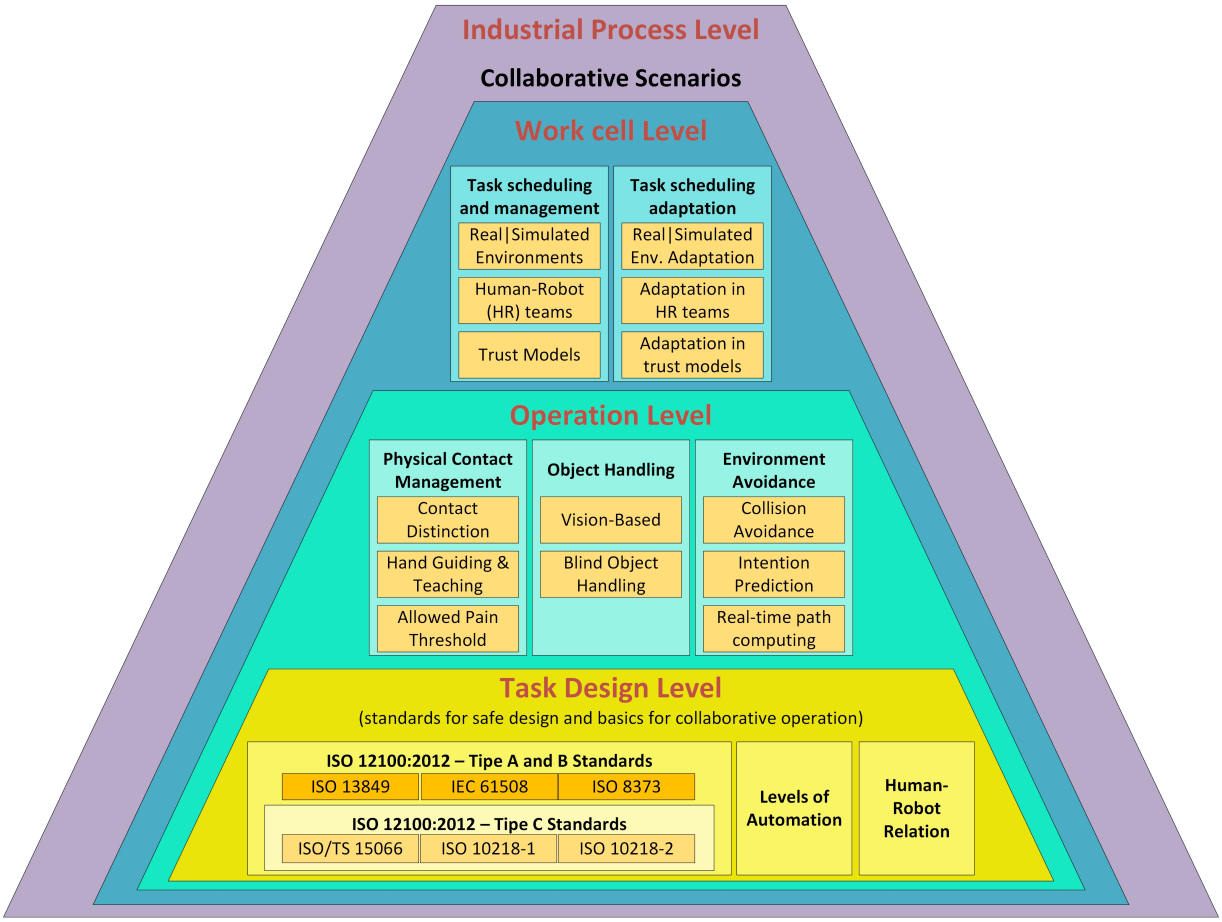

Figure 2.1: Different integration levels and challenges of industrial collaborative scenarios.

These challenges which address five different aspects, also depicted on Figure 2.1, correspond to the Physical contact management [48–50], the Object handling techniques [51–54], the Environment Avoidance [18, 35, 55], the Task scheduling and management [29, 56, 57], and the Task scheduling optimization [30, 58]. They have been born to deal with the necessity of guaranteeing the whole application safety aspects at every moment to allow real fence-free shared workspaces [31]. The last referenced work, the first contribution of this work, responds to

the non existing guidelines of how to implement safety on every level of a collaborative scenario, easing the integration of future collaborative working cells. In other words, allowing a real fence-free industrial environment between robots and workers, aspects as "which tool is the robot carrying?", "is the tool sharp?", "does the tool present a hot end?", or "does the load the robot is carrying is edgy or heavy?" should be also regarded. In fact, if all these elements cannot guarantee a safe application, additionally safety measures are required, reducing the movement freedom and flexibility of industrial shared scenarios [7, 9]. This could lead from fully developing collaborative applications (directly bounded to the Human-Robot Collaboration, HRC [59]) for industrial collaborative scenarios to develop partially fenced working areas where Human-Robot Cooperative (HRCp) [2], or Human-Robot Coexistence (HRCx) [60] working modes are required instead [61].

The different levels of collaborations (HRC, HRCp, and HRCx) takes into account other two dimensions to define a collaborative working team [61]. As shown in Figure 2.2, there are two additional level regarding the HRC interaction levels which are referred to the isolation (the robot working alone) and the synchronization which is somewhere in between the HRCp and the HRCx. However, the relevant fact is that to describe a collabotive applications and its implications the HRC safety implications and the HRC team composition should also beared in mind [45, 61]. In this manner, the HRC safety implications are bounded to how the robot interacts directly with the human operator and determines whether it should stop or reduce its velocity according to the ISO-TS 15066 [2, 5, 62]. In addition it is also relevant for the application how many members interacts in the collaborative application, since more than one robot or human is harder to synchronize and manage safely properly. Hence, an adequate combination of these factors will end up in a suitable collaborative application that is optimal also in terms of manufacturing efficiency [26, 61].

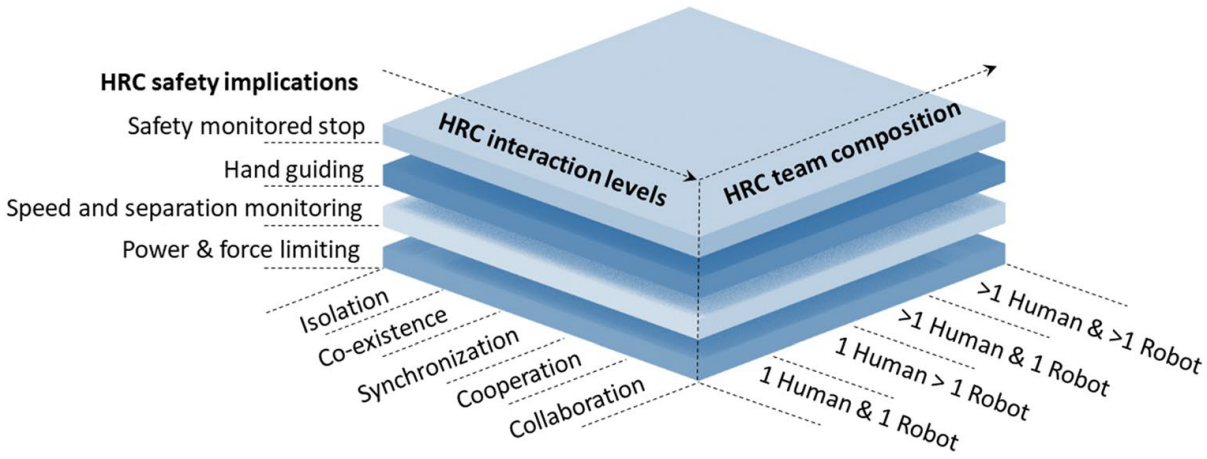

Figure 2.2: Architecture model for human-robot collaboration by Malik et al. [61].

To achieve the HRC on industrial shared environments, each of the aforementioned challenges pretends to face one of the relevant dimensions to enable industrial collaborative scenarios from the physical interaction with objects and operators, to the optimization of tasks distribution between human workers and robots to avoid as much as possible bottlenecks during production [12]. Thus, by addressing the different challenges

in a coordinate manner in current industrial manufacturing environment from the lower to the upper level a safe and efficient collaborative manufacturing could be achieved.

2.1.3 Environment recognition and segmentation

Vision algorithms to segment and recognize the elements of the robots surrounds take on an additional dimension when applied to collaborative robotics [63]. In this environments, being capable of recognizing human motions or, even, their intentions is a key factor to avoid collisions between human and robots to preserve shop floor safety [34]. With the aim to preserve the operator safety, this vision based algorithms are usually focused on identifying just dynamic obstacles, considering the static elements of the pictures as part of the robot scene [64].

In order to do this differentiation between dynamic and static elements of the scene different type of techniques are employed by several authors in the literature. Depending on the aim of the segmentation, these techniques can be divided into three groups as shown in Figure 2.3: the ones based just in extracting the environment (static components) [65–67], the techniques to detect movement (dynamic elements) [68–70], and the techniques to recognize both static and dynamic elements of the scene [71, 72].

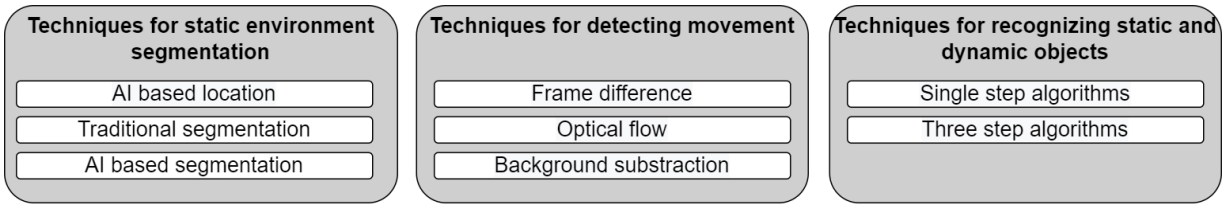

Figure 2.3: Classification of the environment segmentation techniques.

The first group, the techniques for static environment segmentation, are divided again into three subgroups: the Artificial Intelligence (AI) based location techniques, the traditional segmentation ones, and the AI based segmentation. The AI based location techniques serves to know the location of a particular object, however, the full segmentation of the scene is not executed [73]. Even though this method is faster than a full scene segmentation thanks to the usage of AI, the received information might be not enough for robotics application. To obtain more information from the vision sensors, there are also the traditional segmentation techniques where it can be found colour segmentation approaches [65], contour detection solutions [74], and the combination of colour and depth images segmentation (see Figure 2.4) [66]. Among these techniques, the most interesting is the last one since it offers depth information of a segmented environment. Lastly, there are also employed tecnhiques based exclusivelly on AI algorithms that after a proper training, can segment easily the trained object from a picture [67] or from a video [75]. The main drawbacks of these techniques is that whenever they are based on colour filter exclusively they does not bring enough information about the environment. In addition, if the depth filter is used, the resultant algorihtm is to heavy reaching up to 11s to perform a full image segmentation when its crowded of elements [66].

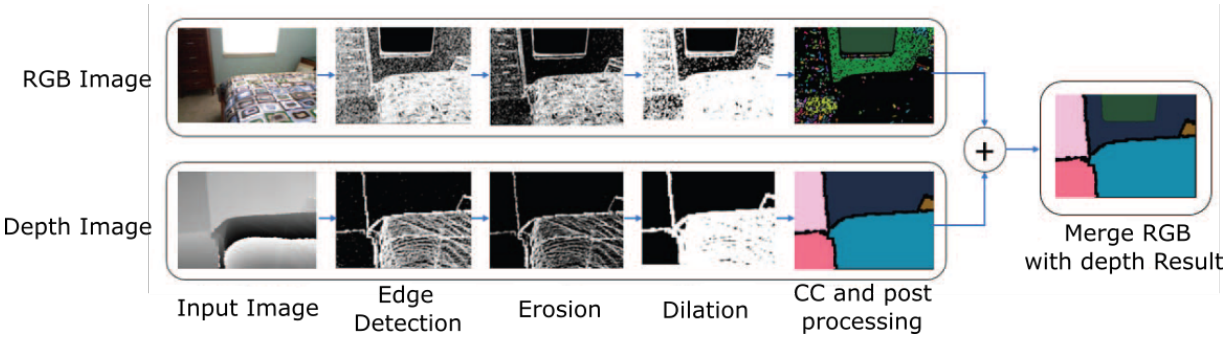

Figure 2.4: Example of a colour and depth combined segmentation [66].

Regarding the standalone techniques to detect movement, it has being identified three categories of techniques to detect movement as displayed in Figure 2.3: the frame difference method, the optical flow algorithm, and the background subtraction technique. The frame difference method relies in generating a temporal buffer of images to analyze the difference between each frame, in this way, the differences between frames represent moving obstacles [68]. Consequently, the optical flow algorithm is based on studying the movement directions of the image or optical flow [76]. Thus, whenever there is an obstacle moving in the camera, the value of optical flow from those pixels will differ from the rest of the picture [69]. The last technique, the background subtraction, is based in subtraction of a first picture with moving objects to one with a pregenerated model of the background, leaving a mask that contains only the moving obstacles of the picture [70, 77]. The main advantage of the presented algorithms is their capability to detect movement with a lightweight algorithm form a picture. However, they present the disadvantages of requiring a good model of the background or comparison image as well as not giving any additional information about the positioning of the moving object.

Due to the limitations of the previously explained vision algorithms, it is required the last group, the techniques for recognizing static and dynamic elements of the scene. From among these techniques, the single step algorithms and the three step algorithms can be distinguished. The first group of algorithms is not very common and they do the location of the objects while detecting if they are static or dynamic simultaneously in a single step [71]. On the contrary, the three step algorithms use three steps to locate the objects, then analyze the movement of the image, and lastly, combining both results to obtained the segmented static and dynamic obstacles [72, 75, 78]. These algorithms can employ a combination of colour and depth segmentation combined with a frame difference [72], or a Convolutional Neural Network (CNN) based segmentation combined with an optical flow technique [78], among other solutions [75]. Since these filters are the ones that give information about positioning of segmented objects, they seem suitable for standard robot applications.

However, just by segmenting and positioning the static and dynamic obstacles of a set of images or a video is not enough for collaborative applications [56]. To enable a real collaborative scenario, is also required to detect from the static elements of the scene which objects is the robot allow to interact or grab. Therefore, additional segmentation steps or algorithms are still required. Moreover, a precise information about the robot surrounding is required by the control algorithms based on

vision techniques or distance measurements to compute adequate control responses in time to evade risky situations. For this reason, the following section will analyze the most commonly employed algorithms that avoids collisions using information from the closest environment to the robot.

2.1.4 Advanced control algorithms to avoid collisions

As stated in the introduction (Subsection 2.1.2), the collision avoidance approach is considered one of the most suitable challenges to keep safety while assuring manufacturing efficiency due to avoid non-required safety stops. The collision avoidance approaches belong to the challenges presented in Subsection 2.1.3, more specifically to the Environment avoidance challenge [31]. It consists of a set of techniques and algorithms to avoid dynamically the collision on a changeable environment [79]. Nevertheless, managing robot movements is a complicated topic due to the possible randomness in the movements of the operators [80–82]. Additionally, the performance of the robot tasks can also be affected by the trust of the human operator in the robot, offering a worst performance the lower the confidence is [43, 83]. Because of this intricate external factors that influence directly the performance of the manufacturing process, the collision avoidance is still a field that requires some efforts on research [84, 85].

To cope with avoiding collision on shared environments where the surroundings of the robot are on a continuous evolution, current collision avoidance techniques are mainly based on two types of reactive techniques: heavy and light computationally algorithms [86]. On the one hand, the heavy computationally algorithms involve vision based-techniques to predict human behavior combined with the calculation of new collision-free trajectories [87]. On the other hand, the light computational algorithms rely on techniques to directly react to changes in the surroundings of the robot, mainly composed by different approaches of Artificial Potential Fields (APFs) [79].

Among the heavy computationally techniques it is worth mentioning Kalman Filters (KFs) [88] or probabilistic based methods such as the Hidden Markov Models (HMMs) [35] or the Gaussian Mixture Regression (GMR) [62], among other techniques to predict human intention [34, 36]. Even though these techniques can predict with sufficient consistentency the operator intentions to get ahead of the collision, one of their main issues corresponds to the necessity of several vision sensors to avoid the occlussion problem [89]. These need of several vision devices to monitor and model the surroundings of the robot implies heavier algorithms to process the industrial environment [28]. Moreover, once the intention of the operator has been predicted, then a collision-free planning technique is applied to find a new safe path that the robot should track [90]. These techniques typically involve an Online Trajectory Generator (OTG) to compute a singularity free trajectory working in the joint space through polinomial smoothed trajectories [91, 92], or in the task space through the manipulability optimization [93, 94]. In the end, these techniques present heavy computational algorithms not just because some re-planning algorithms are computationally demanding, but also for the continuous computation of new trajectories for every change detected in the scene or in the human intention by the vision systems [85].

Regarding the light computational algorithms, several approaches of APF based techniques can be found as the classic standalone obstacle avoidance [92, 95, 96], the risk fields [97, 98], the dynamic fields [99, 100], the evolutionary fields [101, 102], or simplifications through minimum distances fields [89, 103][2]. Since on an changeable environment the operator is not static, one of the most commonly employed methods are the dynamic APFs, which avoid colliding with the human worker without the need of heavy computational algorithms to predict human intention [96]. Some examples of these type of dynamic APFs algorithms are the ones that relies on Dynamic Movement Primitives (DMPs) [99], those utilizing a tangent vector to the tool velocity combined with a risk field [100], or the ones using the memory of the lastest positions of the obstacle to compute a mixed response for that set of last positions [18], among other solutions. Even though they seem like a suitable option due to their quicker response to changes in the environment, because of their direct dependency on the robot jacobian to compute the different responses and control signals, these algorithms could lead the robot to a singular region [104].

2: Even though information relative each specific application can be found in the given reference, the risk fields are the ones that employed risk distance measured following the guidance of ISO-TS 15066; the dynamic fields correspond to the approaches that also takes into account the velocity of the obstacles; and, lastly, evolutionary fields which combines APF methods with genetic algorithms to derive optimal potential functions.

The main drawback of leading a cobot to any of these configurations is that the robot will stop due to excessive velocities of torques computed as control signals [105]. When this occurs, the cobot will trigger its safety stop mechanisms to avoid injuring the operator while generating a window of non-productive time during manufacturing. Therefore, not only the safety of the operator will be compromised but also the manufacturing efficiency will be reduced [40]. To avoid such negative effects during manufacturing processes, some approaches proposed the combination of APF-based controller to avoid collision with a inverse kinematics generalized model bounded to a potential function to compensate the misbehavior of the robot when is on a singular region [106]. However, the potential function can only be computed if the singularities of the robot are known, otherwise they will require a singularity study [105].

3: Some examples of cobot with sperical wrist are the LBR iiwa [107], the CR model of FANUC [39], among other models.

This is possible only for some modern collaborative robots, more specifically to those who present an spherical wrist[3]. The forward and inverse kinematics (FK and IK) solutions for these type of industrial collaborative manipulators are far well-known and a singularity analysis for these robots can be found in [41, 108, 109]. The importance of having an spherical wrist relies in the capability of decoupling the positioning and orienting problem of the robot, what leads to a more simple velocity kinematic model with a simplified Jacobian[4]. This simpler Jacobian is the enabling key to compute the full set of singular configurations for these type of cobots, allowing the subsequent application of inverse kinematics generalized models to manage the null space configurations dynamically [105, 110]. Nevertheless, some of the latest commercial are non-spherical wrist cobots instead[5], so these advantages are not present on the FK and IK models of those cobots [40, 111]. The following subsection will go deeper in the analysis of the limitations of non-spherical wrist kinematics model for control algorithms.

4: The advantage of this simplified Jacobian relies on a $\mathbf{J}_{12} = \mathbf{0}$ matrix, easing determinant computation as block matrices.

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix}$$

5: Some collaborative manipulators models that lack of spherical wrist are the URX/URXe from Universal Robots [37], the Omron Collaborative robot [38], or the CRXi robot models from FANUC [39], among other robots

2.1.5 Non-spherical wrist robots current limitations on control algorithms

To cope with the limitations of APF control algorithms for non-spherical wrist cobots for handling the singularities while avoiding collisions, complementary control techniques have emerged [109, 112, 113]. These techniques to handle singularities and avoiding collision simultaneously can be divided into two big groups as shown in Figure 2.5: those that relies on inverse kinematics generalized models to handle the singularities [114, 115], and the ones that tries to somehow simplify the cobots kinematics to handle computationally the characterization of singularities [116].

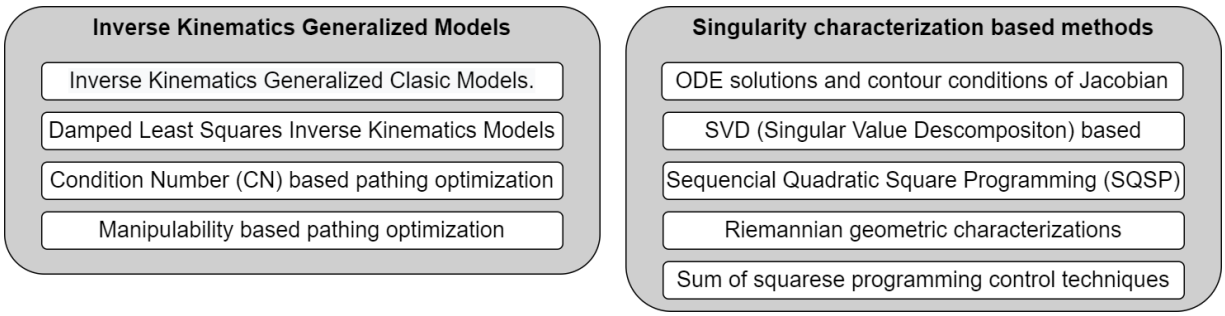

Figure 2.5: Groups and techniques to handle singularities while avoiding collisions for non-spherical wrist robots.

The approaches that relies on inverse kinematics generalized models are those which rely on the Jacobian pseudoinverse ($\mathbf{J}^{\dagger}$) to handle the null space configurations when computing the IK. The general mathematical description of this type of models is typically similar to the expression displayed in equation 2.1. In this equation, it can be appreciated how these inverse kinematic generalized models are composed by, at least, a term to compute the regular velocity from the task space ($\mathbf{J}^{\dagger}\dot{\mathbf{x}}_e$), and another one to compute the compensation due to the singular configuration or other parametrizable objective ($(\mathbf{I}_n - \mathbf{J}^{\dagger}\mathbf{J})\dot{\mathbf{q}}_0$) [41, 108, 109].

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\mathbf{x}}_e + (\mathbf{I}_n - \mathbf{J}^{\dagger}\mathbf{J})\dot{\mathbf{q}}_0 \tag{2.1}$$

Nevertheless, the general formulation displayed in the equation above can be modified for more specific approaches such as the Damped Least Square (DLS)-based controllers [114, 115], the computation of the Condition Number (CN) to calculate and maximize the distance to singularity [41, 108], and the maximization of the manipulability (μ), among other solutions that also rely on inverse kinematics generalized models.

In the case of the controllers that utilize the DLS inverse kinematic model, the general formulation from equation 2.1 is transformed into the one of equation 2.2. As displayed in the equation, the difference with the original inverse kinematic generalized model is the damping coefficient (λ) introduce to compensate the velocities whenever the manipulator is in its null space. This way, the manipulator avoids having a Jacobian which is not full rank so the velocities around the singular configurations of the manipulator are under desired thresholds [108]. However, since this algorithms are characterized by the successive product of Jacobian matrices, they increase considerably the demanded computational cost of

the controller. In addition, they neither solve solve the singularity problem since the singular configuration are shifted from a set of configurations to a newer set [115, 117]. Therefore, this approaches are not desired to be applied in collision avoidance reactive algorithms since blockages due to singularities can occur.

$$\dot{\mathbf{q}} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}_n)^{-1}\dot{\mathbf{x}}_e \tag{2.2}$$

The other two approaches, the CN and the manipulability, both are methods that requires the continuous computation of the Singular Value Decomposition (SVD) [118]. While the CN is defined as the quotient of the smallest (σ_{lowest}) and the biggest ($\sigma_{highest}$) singular values of the Jacobian determinant matrix (see equation 2.3) [41, 108], the Manipulability is the product of each singular value (σ_i) of the Jacobian matrix (see equation 2.4) [93, 109]. Both strategies computes repeatedly the singular values to compute a singularity free optimized path. These continuous calculations are even heavier bearing in mind that the computation should be executed each cycle to found a new collision free path that maximize either of both [119]. Moreover, these two techniques present an additional blind spot due to the possibility of falling into singularities whenever elongated manipulability ellipsoids has short axis [41]. Therefore, similar to the DLS based controller case, these techniques does not seems suitable for its online application to complement reactive collision avoidance control strategies.

$$CN = \frac{\sigma_{lowest}}{\sigma_{highest}} \tag{2.3}$$

$$\mu(\mathbf{q}) = \prod_{i=1}^{n} \sigma_i(\mathbf{J}) \text{ , or } \mu(\mathbf{q}) = \sqrt{\mathbf{J}\mathbf{J}^T} \text{ (if redundant)} \tag{2.4}$$

Contrary to the first group of solutions, the group of approaches that simplifies the cobots kinematics pretend to solve the computationally demanding problem of the previous algorithms during the handling of singular configurations. Even though some kinematic simplifications through the DH convention can be achieved for this kind of robots, they are not enough to fully decouple the positioning and orienting problem of the robot or, at least, compute a characterized set of singular configurations [105]. Some classic examples of approaches that serves to obtain a parametrization on the singular configurations are the SVD and the Ordinary Differential Equation (ODE) combined with boundaries conditions [109].

The SVD method relies on the singular value decomposition of the Jacobian matrix (see equation 2.5) [118]. From among the terms of the SVD equation, the term $\mathbf{\Sigma}$ corresponds to the diagonal singular values (or eigenvalues, σ_i) matrix [41]. In light of this relationship, it could be reasonable to think that it can be possible to obtain a singularities parametrization based on the analytical solution for each of the singular values. However, there is no analytical method to obtain a variable (joint in this case) dependant characterization of the singular values whenever the rank of the Jacobian is equal or greater to 6 ($rank(\mathbf{J}) \geqq 6$) [118].

$$\mathbf{J} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^* \tag{2.5}$$

Another strategy to characterize singular configurations consists of utilizing an ODE and a boundary condition to obtain a set of solutions for the contour equations. In this case, the contour equations corresponds to the established limits between singular configurations and regular positions. As the analytical Jacobian of the manipulator can be defined as shown in equation 2.6, it could be thought to study the singularity as an ODE, obtaining an implicit ODE system that for the condition of non-singularity will be transformed into an explicit ODE system [120]. With this general solution, a particular solution could be found applying a set of initial conditions that fits the definition of singular region (mathematical definition can be found in equataion 2.7). However, similar to what happens with the approach based on the SVD computation, whenever the manipulator present more than 6 DoF, the parametrization cannot be executed due to the exponential explosion of the possible solutions [120]. Therefore, this approach is neither suitable to implement for advanced robot controller to avoid dynamically the collisions.

$$\mathbf{J}_a = \frac{\partial \mathbf{k}(\mathbf{q})}{\partial \mathbf{q}} \tag{2.6}$$

$$\Psi = \{\mathbf{q} \in \mathbb{R}^n : |\mathbf{J}(\mathbf{q})| \leq s_0\} \tag{2.7}$$

To overcome the limitations of traditional approaches to characterize singular configurations, there are more modern techniques directly integrated into control algorithms. The main utilized techniques correspond to Riemannian geometric characterizations of the robot [121], the sequential quadratic square programming control barrier functions [122], and the sum-of-squares programming control techniques [123], among other solutions. These techniques add constraints to the robot kinematics in different formats. In the case of the Riemannian geometric characterization the constraint is based on a novel singularity distance index [121]. The constraints for the Sequential Quadratic Square Programming (SQSP) are described through a control barrier function in the task space [122]. Lastly, in the Sum-of-Squares Programming approach, it is proposed two polynomially described ellipsoids to create a constrained singularity-free workspace [123].

For the two first approaches, the constraints pretend to compensate the end effector velocity within singular regions [121, 122]. However, in the case of the last solution, the constraint describes an inner and outer volume of a limited and approximate singularity-free workspace to compute a smoother velocity for the end effector [123]. Since these approaches relies on kinematic additional constraints, the computational power demanded by the controllers is still elevated due to additional calculations to introduce the restrictions [116]. This leads to a local handling of the singularity each cycle instead of having a characterized set of solutions for these configurations [112]. Therefore, the velocity is handled within the singular regions, but those regions are not avoided so they are not candidates to be applied reliably into advanced reactive control algorithms for robotics.

The presented algorithms above have the main problem of being too demanding in computational power or being unable to obtains a parameterized set of singular configurations. On the one hand, the solutions of the first group (the ones that relies on inverse kinematics) rely on heavy matrix products computation or the calculation of the singular values of the Jacobian [115]. Since typical collaborative manipulators are 6 or 7 DoF [45], the analytical solutions for these approaches cannot be obtained because there is no analytical method to do this, forcing the robot task programmer to apply some numerical approximations [118]. On the other hand, the solutions that simplifies the kinematic model are not either suitable due to the high DoF of non-spherical wrist manipulators [116]. This leads, as displayed in the robotics theory literature [41, 108, 109], to the study of the analytical solutions directly from the Jacobian determinant of a 6 or 7 DoFs. This solutions can only be obtained if the Jacobian matrix has a $\mathbf{J}_{12} = \mathbf{0}$ block matrix for robots with greater than 6 DoFs. In other words, it requires the kinematic model to be decoupled to achieve that the lineal velocity of the kinematic model is independent from the last three joints of the manipulator [109].

2.1.6 Conclusion

Collaborative industrial scenarios are a complex environment where it must be taken into account several agents from the human operator, going through the vision systems and sensors, to the robot. Regarding the human operator, the main issue to guarantee is its safety. Otherwise these environments that lacks of additional protection measures cannot be achievable, impeding reaching the desired level of autonomy. However, this could only be achieved by a proper used of the vision systems and external sensors, to recognize precisely the surrounding of the robot, combined with advanced control algorithms to keep the safety while maximizing the manufacturing efficiency. This only could be achieved by addressing adequately the proposed challenges from Subsection 2.1.2.

Moreover, some of the most advanced current vision algorithms are able to segment the environment of a picture while giving information about the position of each element. Nevertheless, they lack on distinguishing between static elements that are relevant to the robot application from the components of the picture that are just a mere part of the scene (the robot should not interact directly with them). The static elements of the scene should be avoided during the offline phases of the planning stages, while the remaining static elements correspond to work pieces or objects the robot should carry from one place to another. Not giving any information about this last type of static elements of the scene, might end up in a misbehaviour or mishandling of the work pieces during the application. Therefore, there is still required a segmentation algorithms capable of distinguishing between the scene of the robot, the static elements, and the obstacles to be avoided dynamically.

Then, this information should be feed to the robot controllers to handle efficiently the collision avoidance operations in order not to waste manufacturing time with continuous non-required safety stops. However, these algorithms present the chance of fall into a singular configuration, endangering the operator or blocking the robot if the safety stop is triggered enough soon. Even though the management of singularities

and collisions simultaneously seems a task solved long ago, current solutions to handle simultaneously collisions and singularities on dynamic environments lack on a closed set of solutions for the IK and the singular configurations of non-spherical wrist cobots. This leads to computationally demanding control algorithms that cannot avoid the singular configurations, increasing the chances of a safety stop due to falling into singularities.

In this manner, to achieve a more efficient industrial collaborative scenario that cares for the operator safety without disregarding manufacturing performance, a simplified kinematic model is required for non-spherical wrist cobots. These required necessities can be summed up into a suitable kinematic model that allows the decoupling of non-spherical wrist manipulators to have a simplified but efficient kinematic model. This simplified model, if based on a decoupled kinematic model, can enable the computation of an analytical closed set of solutions for the IK, a result that is still not achieved. Moreover, this simplification of the kinematic model will come along with a simplification of the Jacobian matrix of the non-spherical wrist cobot. In this way, the singularities can be easily studied allowing the desired characterization that current approaches lacks of. Then, this results can be combined into advanced APF based controllers to avoid simultaneously collisions and singularities instead of just hanlding them. By reaching this three millestones safer and more efficient collaborative environments can be obtained, being one step closer to the desired industrial collaborative scenario.

2.2 Hypothesis

Among all the solutions displayed in the State of the Art, the APF-based controllers are considered one of the most suitable methods to achieve real time safe reactive scenarios. As this kind of controller does not require a planning phase to react to changes in the environment, they seem suitable to keep the production performance while maintaining safety of the operators in industrial collaborative scenarios.

Regarding collision avoidance, theAPF-based controllers are one of the most employed techniques to dynamically avoid collisions in environments that are in continuous evolution. This is because it relies directly on the distance from each obstacle to the distributed control points along the structure of the manipulator. The distances computed are employed to build a virtual velocity or force potential field. Subsequently, this field is employed to emulate a virtual damping system between the robot and the environment, allowing the robot to avoid collisions. However, to transform or compute the virtual repulsive velocities or forces (joint commands), it is required the inverse of the jacobian of the robot structure.

Due to these reasons, when the control command is being computed, and the robot is in a kinematic ill-conditioned state, the computed response can be excessive, increasing the application risks and triggering a non-desired safety stop [105]. Thus, these controllers are usually combined with auxiliary methods to handle singularities based on projections over the null space, such as the Damped Least Square (DLS) based controllers;

or kinematic simplifications through additional constraints, as the Riemannian geometric characterizations. Even though these approaches can be employed to handle obstacle avoidance and singularity simultaneously, they do not fully solve the singularity problem for non-spherical wrist cobots. What these controllers intend to do is to compensate the end effector velocity within singularities, or shift the singular configurations to other configurations away from the original ones. Therefore, the robot control is not free to fall into ill-conditioned configurations that triggers a safety stop, wasting time and money. To avoid falling into singular regions, a complete knowledge of the solutions to the IK that characterize the singularities is required [105]. Since this set of solutions is only available with low time-cost for spherical wrist cobots, some of the current commercial cobots models, being of 6 or 7 Degrees of Freedom (DoFs) non-spherical wrist robots, cannot benefit from the current collision avoidance controllers efficiently.

Because of the panorama exposed above and regarding the motivation of this research, the following statement is proposed as the hypothesis of this doctorate:

Hypothesis

Through the coordinated action of two different Artificial Potential Field (APF) controllers, one for avoiding collisions and the other for repelling singular configurations, and its combination into a novel dual Artificial Potential Field (d-APF) based controller for simultaneous collision and singularity avoidance, it is pretended to keep the required safety levels without disregarding manufacturing cycle times.

Thus, responding to the presented hypothesis is expected to go forward in the development of real industrial collaborative scenarios where safety is kept (through the collision avoidance) without falling into blocking configurations (which increases non-desired manufacturing bottlenecks).

2.3 Objectives

The global objective of the current doctorate study is aligned with the motivation of the research, as well as it tries to get an answer to the hypothesis with a suitable advanced control strategy. Therefore, the main objective of this work can be summed up in the statement presented below:

Global objective

This Ph.D. aims to achieve an industrial collaborative environment where collisions are avoided while dodging other risky situations to preserve the safety of the operators and reduce the production bottlenecks due to safety stops of industrial collaborative robots (6 or 7 DoFs) through APF-based controllers.

Achieving such a goal requires the attainment of several partial objectives. Each of the partial objectives presented below pretends to face at least one of the aspects detected that make the implementation of fluid manufacturing processes on collaborative tasks difficult[6].

- **Objective I - Safe collision avoidance controller**: The first partial objective of the Ph.D. consists of a suitable real time implementation of an APF-based controller to avoid the collisions between robots and operators. Thus, this first development is related to handling the safety issues for shared environments and avoiding risky situations between operators and robots by modifying the proposed reference controller that can be found in [18, 63]. The challenge is to adapt its behavior to classic singularity handling algorithms such as the DLS-APF based controllers.
- **Objective II - Singularity characterization**: as stated before, implementing an APF-based controller to avoid singularities requires knowing them in advance. In this manner, they can be treated as if they were obstacles from the surroundings of the robot, allowing dynamic avoidance while evading obstacles. Therefore, one of the major partial objectives of this Ph.D. consists of a kinematic singularity approach that allows the parametrization of the singular configurations depending on the joints of the manipulator.
- **Objective III - Safe singularity avoidance controller**: Once the singular configuration has been parameterized, the APF-based controller for collision avoidance should be modified to integrate the solutions of the singularity characterization, resulting in the proposed dual Artificial Potential Field (d-APF). In this manner, the chosen strategy to carry out the singularity avoidance integration should repel the singular configurations as if they were obstacles.
- **Objective IV - Performance test on collaborative application**: This last partial objective pretends to test the performance of the proposed controller based on APF. Therefore, some collaborative assembly or operation should be executed to check the behavior of the controller in real or simulated manufacturing scenarios. This test will also help evaluate the chances of implementing collaborative scenarios in current industrial manufacturing environments.

By managing the partial objectives presented above, it is pretended to provide a solution to the problem stated in the hypothesis.

6: Complementary to the presented partial objectives a proper environment segmentation to distinguish between human workers or obstacles, workpieces to operate with, and static components of the scene is also needed. Even though a good segmentation of the environment is required for a successful implementation of this type of advanced robot controllers, the vision algorithms are based on the work of previous authors. Since they cannot be considered as a novelty of this work, they are not presented as a partial objective to achieve. However, it is worth mentioning the importance of the vision system as a key component for the proper execution of these kind of applications.

Research Challenges, Methods and Equipments

3

In this chapter a quick overview of the challenges and why the simultaneous avoidance of collisions and singularities is considered a novelty under the method proposed in the hypothesis. Consequently, the employed research method to carry out the investigation on the field of collaborative robotics for advanced industrial shared scenarios is showed. Lastly, this chapter also shows the available material and equipment to accomplish the research objectives.

3.1 Challenges of the Research

This thesis pretends to go one step forward in the development of industrial collaborative scenario through the implementation of a dual Artificial Potential Field (d-APF) advanced controller to guarantee the safe operation in the surroundings of the robot without disregarding manufacturing efficiency. Therefore, the main challenge to address by completing the objectives of this work correspond to dealing with the singularity on non-spherical wrist cobots while avoiding efficiently and dynamically the collision with operators or obstacles.

Since the part of a safe collision avoidance is already achieved for both spherical wrist and non-concatenated cobots, the real deal of this research is to avoid singularities when the non-spherical wrist cobots is avoiding an obstacle. During the planning phases of a robot task, the controller plans the trajectory handling singularities. However, when applying reactive collision avoidance control algorithms, the most efficient methods avoid continuous re-planning of the trajectory. In this manner, the robot can respond quicker to changes in the environment. These avoidance strategies commonly relies in the Jacobian to compute new reference control signals.

Due to these reasons, while avoiding the obstacle the robot does not track the original trajectory which was prepared to handle singularities, computing a new set of velocity reference signals dynamically. It is precisely this dynamic calculation of the velocites through the Jacobian what can lead the robot to a singular configuration. As shown in Section 1.2, if the robot does not has an non-spherical wrist, the available singularity handling algorithms does not solve efficiently the problem. Therefore, one of the challenge of this research relies in the necessity of modifying the current solving of the kinematics of non-spherical wrist cobots to obtain a more simple FK and IK model.

This model should allow the kinematic decoupling of the positioning and orienting problem to enable the computation of simpler forms of the Jacobian that guarantees the parametrization of the singular configuration for the non-spherical wrist cobots. Indeed, this characterization of singularities can be considered as another contribution to address by this Ph.D. thesis.

3.2 Research Method Proposal

The followed research method is based on a quantitative set of procedures to evaluate the advantages of the proposed d-APF controller over traditional methods to avoid collision while handling singularity on an hybrid environment. This means that the chosen methods to measure the performance and suitability of this proposal have been tested first on a simulated environment and then, on a real laboratory scenario.

This hybrid approach to test the effectiveness of the proposed controller relies on the virtualized (simulated) environments to test the performance of the developments made. In this manner, a safe and controlled first trials can be executed without risking the integrity of any machine or worker. Once the approach has been validated on this simulated environment, it is tested on a real scenario without risk any for the real laboratory equipment or any laboratory assistant helping in the experimentation phases.

In order to assure the success of the proposed hybrid method the migration of the implemented simulated application to the real system should be transparent. This means that the implementation of the controller in simulation should be as closed as possible to a *plug and play* integration for the real robot controller. To achieve this type of implementation, guarantee the code re-usability between robot applications, and develop hardware agnostic advanced robotic controllers, from among other benefits, the proposed research method relies on Robot Operating System (ROS) [124], more specifically the Melodic Morenia distribution. This particular distribution has been chosen because it was the latest maintained ROS distribution by the time the Ph.D. started. Due to how ROS internally works, by choosing this hardware agnostic framework it is ensured for both, the simulated and the real scenario, to share the developed controllers without further modifications than a typical controller parameter tuning. Even thought ROS simplifies the task of sharing simulated and real robot controllers, this seamless switch can only be achieved if the simulation environment is reliable.

In order to achieved the aforementioned interchangeability between simulated and real application, the behavior and placement of each simulated component should emulate its real world mirror component as similar as possible. Thus, in addition to just copy the distribution of the different components configuration and distribution, each element should also be modeled to simulate their physical dynamic behavior. Indeed, the idea is to create a reliable virtual model of the robot and its environment. In order to model virtually the robot scene, ROS is supported by two complementary softwares already integrated with it: MoveIt [125] and Gazebo [126]. In this manner, MoveIt will handle the kinematics and trajectory execution of the robot, while Gazebo will be in charge of simulating the physics of the robot model and its joints. Therefore, while MoveIt will be active in both environments, Gazebo will only be active while simulating the robot, being the natural replacement of the robot controller in simulated application.

This research method has also been selected because it can help reducing commisioning times on real collaborative robotics applications. On the one hand, the fact that the controller can be tested first on a simulated

environment makes possible to develop advanced control application even when a real robot is not available at the moment. Moreover, if the Gazebo simulation is accurate and emulates a similar behavior to the real robot, the controller parameter tuning process is a straightforward task to do without further complications when switching to the real robot. Since the ROS-ready robot repositories generally come prepared with tuned drivers to make them work, this last task of parameter tuning can be more easily executed when using ROS-ready commercial cobots[1]. Therefore, this research method seem suitable not only for research but also for a future application and development on real industrial scenarios.

1: E.g. the Universal Robots ROS driver [127], the KUKA IIWA ROS drivers [128], among other available robot repositories.

3.3 Data Collection and Analysis

For the data collection, ROS based nodes and functionalities have been employed. One of the versatilities ROS is is that each data generated in every node can be broadcasted to be listened by other node of the "ROS communication graph". These broadcasted data sets are stored, and once all the data have been collected, the Matlab® R2018b version has been utilized to process and analyze the captured data. This software version has also been used for computing all the theoretical model and simulation tests required during the Ph.D. thanks to the Symbolic Toolbox. All the results and graphs obtained by this data analysis will be presented in Chapter 8 followed by the corresponding discussion, also in this chapter.

3.4 Materials and Equipment

The equipment and material employed to reach the aim can be divided into two different groups: the validation environment and the simulation environment groups. This classification is aligned with the research research method explained at Section 3.2. In this manner, the simulation environments collects the different devices required for the simulated tests in the two different scenarios. The first test scenario corresponds to the collaborative work cell of the research laboratories of Ikerlan S. Coop.[2]. Additionally, the second validation environments is related to the collaborative cell from SUPSI (Scuola Universitaria Professionale della Svizzera Italiana). In this way, the UR10e from the collaborative work cell at Ikerlan serves as the testing subject of all the proposed d-APF controller in simulation as well as in the real scenario, while the SUPSI robot from this laboratory has been utilized as a support to verify and test that the designed control algorithms are suitable also for other deliberative robots.

2: An actual member of the BRTA (Basque Research and Technology Alliance) for the development of advanced technical solutions in the Basque country.

In order to carry out the simulations for both scenarios, the tests have been executed in the same computer as no particular hardware setup is required. In addition to these results, the laboratory tests executed are collected in the validation environments category. Unlike in simulation environments, their specific equipment and material for the real tests is presented for each of the scenarios. For this reason, for both environments the collaborative work cell from Ikerlan and the SUPSI will be respectively addressed in the subsections below.

3.4.1 Validation Environments

The first of the facilities presented corresponds to the Digilab laboratory at Ikerlan, the available resources correspond with the following listed elements:

- 1× UR10e robot from Universal Robot.
- 1× Robotiq 2F-140 gripper.
- 1× Intel® Realsense D435 depth camera sensor.
- 1× Harting Ha-VIS eCon 2000 industrial switch.
- 1× Omron S9VK-G12024 DC power supply.
- 1× External control PC with an Ubuntu 18.04 OS patched with the PREEMPT_RT kernel [129] and ROS Melodic distribution[3].
- 1× Application PC with an Ubuntu 18.04 OS with regular kernel and ROS Melodic distribution[4].

All the elements listed above are displayed and highlighted in Figure 3.1. This figure represents the position of every component in the collaborative working assembly cell available at the laboratory of Ikerlan from different perspectives. Thus, the available space to develop collaborative applications can be better understood.

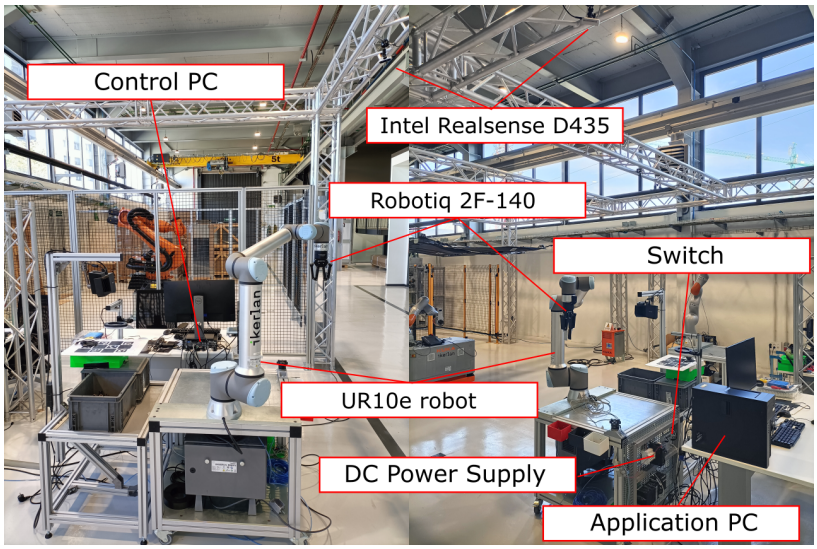

Figure 3.1: Ikerlan laboratory (Digiliab): collaborative assembly cell. The image displays the same robot from two different perspectives.

In the case of the collaborative scenario of the SUPSI University, the available equipment and materials are listed below:

- 1× SUPSI 6 DoFs self-developed cobot.
- 1× Robotiq Hand-E gripper.
- 2× Intel® Realsense D455 depth camera sensor.
- 1× Control and application PC with an Ubuntu 18.04 OS patched with the PREEMPT_RT kernel and ROS Melodic dstribution[5].
- 1× Vision processing PC with an Ubuntu 20.04 OS with regular kernel and ROS 2 Humble distribution[6].

In Figure 3.2 a clear view of the collaborative assembly cell of the SUPSI laboratory can be appreciated. In this figure, it has also been highlighted the different devices listed above to ease the understanding of the collaborative work well distribution.

3: The control PC is a DELL OptiPlex 3040 D11S equiped with an Intel® Core i7-7700 CPU (@ 3.60$GHz \times 8$ cores), integrated intel graphics card (Intel® HD Graphics 630 - KBL GT2), 8GB of memory, and internal storage of 256 HDD.

4: The application PC is a DELL OptiPlex 9020 D13M workstation and has the following specifications: Intel® Xeon(R) CPU E3-1270 (@3.60$GHz \times 8$ cores), dedicated graphics card (NVIDIA Quadro K2200/PCIe/SSE2), 16GB of memory, and internal storage of 256 HDD.

5: The control and application PC is Lenovo ThinkCentre M720q 10T700AAMZ with an Intel® Core i9-9900T (@ 4.4$GHzx8$ cores), integrated intel graphics card (Intel® UHD Graphics 630), 32GB of memory, and 512GB of SDD internal storage

6: The vision processing PC specifications correspond to a Intel® Core i9-9900K (@ 4.8$GHzx8$ cores), dedicated graphics card (NVIDIA RTX 3070), 64GB of memory, and 1TB of SDD internal storage

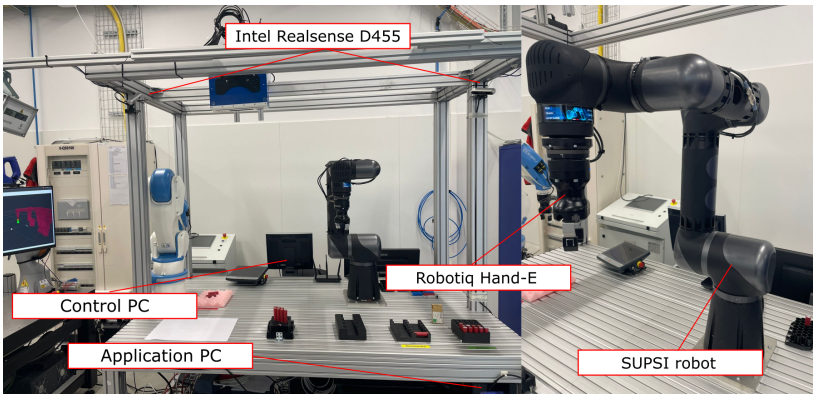

Figure 3.2: SUPSI laboratory: collaborative assembly cell. The image displays the same robot from two different perspectives.

3.4.2 Simulation Environments

The only device employed for the simulation of both scenarios is a laptop with an Ubuntu 18.04 OS patched with the PREEMPT_RT kernel and ROS Melodic distribution[7].

The two laboratories have been virtualized independently to fit with each scenario. Since components like the switch or the power supplies are only required in the physical environment, those devices and connectors are not mirrored. Therefore, the only components represented in the simulation will be the robot, the stand where they are fixed, and other additional sensors required, as displayed in Figure 3.3 and Figure 3.4[8].

7: The simulation PC is a DELL latitude E5570 with the following specifications: Intel® Core i7-6600U CPU (@ $2.60GHz \times$ 4 cores), integrated intel graphics card (Intel® HD Graphics 520 - SKL GT2), $8GB$ of memory, and internal storage of 512 HDD.

8: Note that the blue walls of Figure 3.4 that appear exclusively on the RViz side are non-real world elements that have been included to delimit the movements when computing trajectories with RViz to guarantee operators and other equipment integrity during offline planning phases.

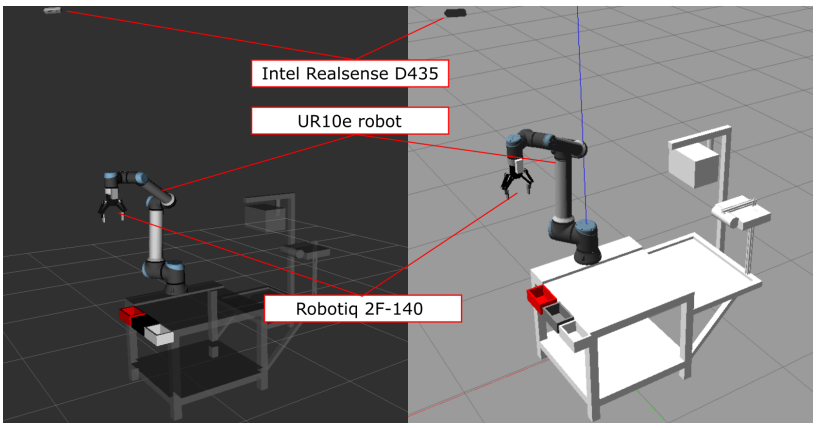

Figure 3.3: Ikerlan laboratory (Digilab) virtualization/Digital Twin: MoveIt RViz visualization (left-hand side) and Gazebo simulator visualization (right-hand side).

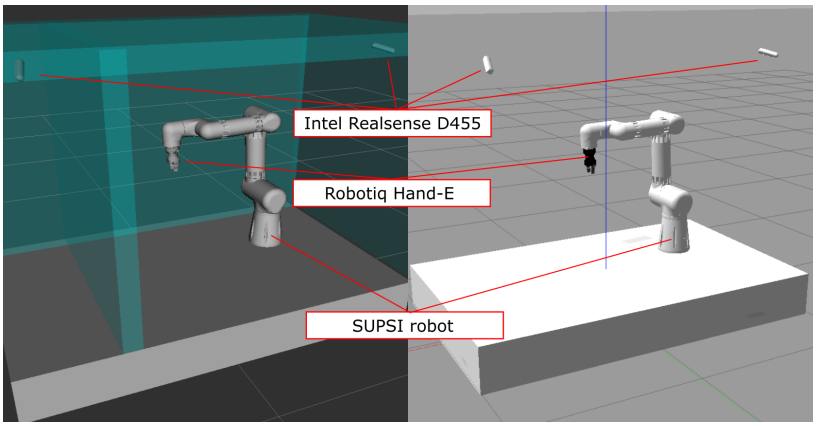

Figure 3.4: SUPSI laboratory virtualization/Digital Twin: MoveIt RViz visualization (left-hand side) and Gazebo simulator visualization (right-hand side).

Kinematic Behavior of Non-Spherical Wrist Robots

Non-Spherical Wrist Robots Decoupled Modeling

4

In classical robotics theory, modifying the arrangement of the reference systems to find a robotic system whose kinematics is more advantageous is a common practice. An example of these strategies to obtain a simpler kinematic model is the kinematic decoupling (with or without DH convention applied) [41, 108, 109], among other available simplifications seen in the State of the Art (Section 2.1). Nevertheless, since the last three joints shafts of non-spherical wrist cobots does not meet at the same spot of the robot structure, their kinematic model cannot be naturally decoupled, so the orientation and positioning problem cannot be handled independently. This inability to naturally decouple the kinematic model of non-spherical wrist cobots impedes the computation of both, a closed set of solutions for the IK, and a full joint dependant characterization of singular configurations.

Moreover, even though some kinematic simplifications through the DH convention can be achieved for this kind of robots, they are not enough to fully decouple the positioning and orienting problem as displayed in Chapter 2. On top of that, due to the existing limitation of analytic methods to solve the determinant of matrices with rank greater than 5, the parametrization of the singular configurations for a 6-th or greater DoF robot is not doable [REF]. Therefore, approaches such as the numerical zeroing of the manipulability, a parametrization depending on the SVD, or an Ordinary Differential Equation (ODE) and a boundary condition strategy, or even a straightforward study of the solutions of the Jacobian matrix determinant cannot be applied to obtain a parametrized set of singular configurations. In other words, this restriction in the mathematical calculation of the singularities causes the necessity of a decouple kinematic model to obtain a kinematic model where the linear velocity of the manipulator is independent from its last three joints.

As can be seen from the State of the Art and the paragraphs above, in order to achieve a closed set of solutions for the IK and the singular configurations, the 6 and 7 DoF non-spherical wrist cobots require a novel kinematic model that could decouple the position and orientation of these robot structures. Therefore, the first contribution this chapter addresses is the design of a suitable decoupled kinematic model for 6 and 7 DoFs robot. For this aim, the fundamental features and requisites to apply the kinematic decoupling for non-spherical wrist cobots are presented. It is also displayed the FK and IK of the proposed kinematic model for both, the positioning and orienting, and for the velocities of the manipulator. Lastly, another contribution is discussed related to the kinematic behavioral implications and singularities simplification when employing the proposed model instead, leading to a brief conclusion of the specific topic addressed in Chapter 4.

4.1 Model Fundamentals

The aim of the proposed decoupled kinematic model is to emulate the kinematic behavior of the decoupled model displayed in the literature [109] applicable only to spherical wrist robots but for non-spherical wrist cobots instead. In this manner, the positioning and orientation of the end effector becomes independent, so they can be treated separately while doing the IK computations on planning stages of the robotic task. Moreover, the kinematic decoupling also alleviates the computational workload for computing dynamic responses or control signals through a simplified Jacobian matrix for the spherical wrist robots, enabling the computation of a closed set of solutions for the IK and the singularities. Both analytical solutions can be latter applied to achieved more efficient advanced control algorithms in industrial collaborative scenarios.

The kinematic decoupling works because the last three joint shafts meet at a point of the robot structure, allowing the computation of the kinematic model from this spot instead. This phenomena can be appreciated at the left-hand side of Figure 4.1 on a traditional industrial spherical wrist robot. Both constant distances are represented by r_1 and r_2 corresponding to the $\overline{FA}$ and $\overline{AE}$ vectors, respectively. This point of the robot where the three joint shafts meet is also known as the decoupling point. How to select a suitable point along the robot structure to be the decoupling point is a critical aspect of decoupling the robot kinematics because it allows to relate the wrist positioning with the position of the Tool Center Point (TCP) automatically, and vice versa[1].

1: This relationship can be better appreciated in the representations above the left-hand side manipulator with the kinematic schematics of two spherical wrist examples.

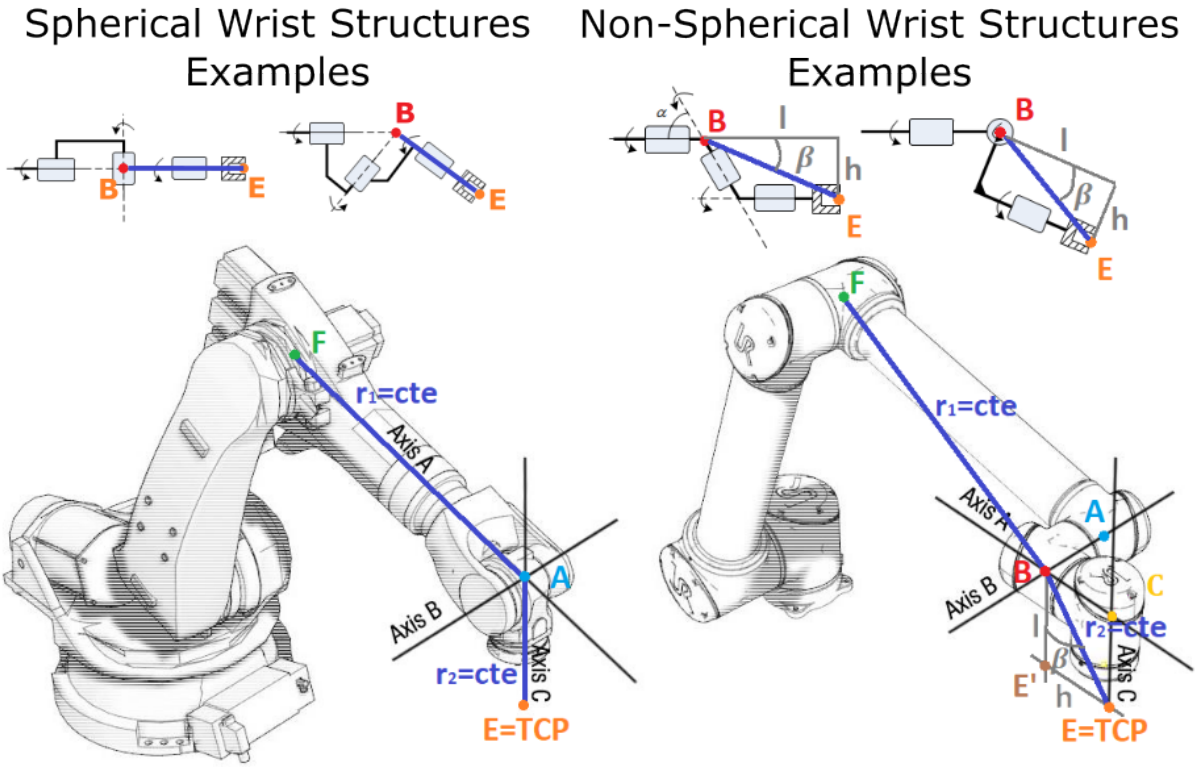

Figure 4.1: Spherical (left-hand side) and non-spherical (right-hand side) wrist industrial manipulators examples (combined and modified from sources [119, 130])

With the intention of replicate this behavior, two new r_1 and r_2 constant distances should be selected for the non-spherical wrist structure. These two distances, also displayed at the right-hand side of Figure 4.1, are the responsible of creating the required constant relationships to build the proposed kinematic decoupling for non-spherical wrist cobots: the distance between the last joint prior to the wrist and the decoupling point (distance $\overline{FB}$), and the one between the decoupling point and the TCP (distance $\overline{BE}$). Any point in the robot structure that gather the criteria of keeping these two distances constant can be chosen as the decoupling point [2]. This process of selecting the two constant distances to establish a univocal relationship between the arm position, the decoupling point, and the TCP correspond to one of the main contributions proposed by this Ph.D. and is denoted as wrist spherification. The aim of this technique is to obtain a quasi-spherical wrist, or in other words, a non-spherical wrist that kinematically behaves as an spherical one.

2: For instance, for the Figure 4.1, the selected point along the robot structure on the right-hand side is the **B** point.

In order to apply the proposed wrist spherification the cooplanarity between **F** with **B**, and **B** with **E**, respectively, must be kept. This condition guarantees that the selected $\overline{FB}$ and $\overline{BE}$ vectors remains constant independently of the robot configuration. It means that both parameters are linked to the robot design structure, and can be considered structural parameters. In this manner, the $\overline{BE}$ vector is always pointing to the TCP, positioning the end effector in the contour of a constant radius sphere centered around the decoupling point **B**. Moreover, the fact that $\overline{BE}$ is a structural parameter makes it possible to relate the decoupling point (**B**) and the TCP (**E**) through the β angle also represented in Figure 4.1. The β angle can be defined as the one formed between the parallel axis to $\overline{CE}$ that goes through the **B** point and the $\overline{BE}$ vector. Thus, the β angle and its modulus ($\overline{BE}$ vector) can be computed as shown in equation 4.1.

$$\beta = \tan\left(\frac{h}{l}\right); |\overline{BE}| = r_2 = \sqrt{h^2 + l^2} \tag{4.1}$$

The aim of selecting a decoupling point **B** is enabling the reconfiguration of the positioning of the reference frames of the wrist of the cobot. This way, the 4-th, 5-th and the 6-th reference frames ($\mathbf{O}_4$, $\mathbf{O}_5$ and $\mathbf{O}_6$, respectively) can be moved into the decoupling point. In general, the decoupling point **B** coincides with $\mathbf{O}_5$, if so, only $\mathbf{O}_4$ and $\mathbf{O}_6$ should be displaced. Bearing in mind these considerations, the proposed kinematic model for non-spherical wrist cobots will behave as the decoupled one for spherical wrist robots. In Figure 4.2 a simplified closed kinematic chain is shown where this relationships can be appreciated. In this simplification it can be seen how the TCP shares the orientation with the decoupling point, and both are related with the constant know distance of $\overline{BE}$, where $\overline{BE} = \mathbf{p}^0_{BE} = \mathbf{p}^0_{EB}$. These orientation and positioning relations are described mathematically by the equations 4.2a and 4.2b[3].

$$\mathbf{p}^0_E = \mathbf{p}^0_B + \mathbf{p}^0_{BE} = \mathbf{p}^0_B + \mathbf{R}^0_\beta \mathbf{p}^\beta_{BE} \tag{4.2a}$$

$$\mathbf{R}^0_E(\mathbf{q}) = \mathbf{R}^0_B(\mathbf{q}) \tag{4.2b}$$

3: In these equations, $\mathbf{R}^0_\beta \in \mathbb{R}^{3x3}$ is the rotation matrix of the β auxiliary reference system according to the world, $\mathbf{p}^\beta_{BE} \in \mathbb{R}^3$ corresponds with the $\overline{BE}$ vector expressed in β reference frame, and $\mathbf{R}^0_B(\mathbf{q}) \in \mathbb{R}^{3x3}$ and $\mathbf{R}^0_E(\mathbf{q}) \in \mathbb{R}^{3x3}$ represent the orientation as rotations matrix of the decoupling point and the TCP, respectively.

The model and relationships displayed in equation 4.2 correspond to the relations required for the FK. In other words, they express how to relate

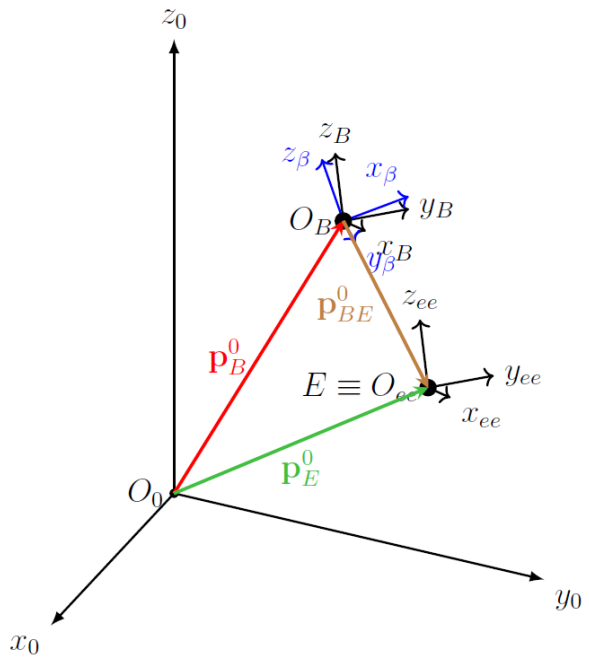

Figure 4.2: Simplified position and orientation transformations between the decoupling point and the end effector for non-spherical wrist decoupled kinematic model. In this representation, the 0-th reference system is the robot base reference system, the B reference system is referred to the last reference sysstem after applying all the joints to the model, the ee reference system corresponds to the one of the end effector, and β reference system is related to the pointing reference auxiliary system that bounds the decoupling point with the end effector.

the position of the TCP from the decoupling point where the kinematic model is solved. This implies that to compute the IK model, the inverse relation should be computed. This inverse relation is detailed below in equation 4.3

$$\mathbf{p}^0_B = \mathbf{p}^0_E + \mathbf{R}^0_\beta(-\mathbf{p}^\beta_{BE}) \tag{4.3a}$$

$$\mathbf{R}^0_B(\mathbf{q}) = \mathbf{R}^0_E(\mathbf{q}) \tag{4.3b}$$

By using the expressions from equations 4.3a and 4.3b, the position and orientation can be computed to fit the position and orientation of the real world workpiece. This is relevant, because naturally, human workers and engineers are more used to work in the task space instead of the joint space. Therefore, the ultimate goal of the mathematical formulation fundamentals consists on obtaining a compact and suitable expression to express the desired position and orientation of the decoupling point. Thus, to describe the decoupling point pose it is employed the equation 4.4, where $\mathbf{p}^0_B$ represent the Cartesian position of the decoupling point, and $\mathbf{o}^0_B$ correspond to the minimal expression of the orientation expressed as the roll, pitch and yaw angles (corresponding to the latter expression $\mathbf{f}(\mathbf{R}^0_B(\mathbf{q}))$).

$$\mathbf{x}^0_B = \begin{bmatrix} \mathbf{p}^0_B(q_1, q_2, q_3) \\ \mathbf{o}^0_B(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \mathbf{p}^0_E + \mathbf{R}^0_\beta(-\mathbf{p}^\beta_{BE}) \\ \mathbf{f}(\mathbf{R}^0_B(\mathbf{q})) \end{bmatrix} \tag{4.4}$$

With the combinations of equation 4.4 in with equations 4.2 or 4.3, whether it is desired to address the FK or the IK, respectively, the kinematic model is ready to address generically the formulation of both, FK or IK.

4.2 Forward Kinematics

The Forward Kinematic (FK) of a non-spherical wrist cobot pretend to fill the desired position and orientation given by the joint configuration of the robot (for a 6 and 7 DoF robot). On a decoupled kinematic model these position and orientation is computed according to the decoupling point instead. Therefore, in order to compute the decoupling point pose ($\mathbf{x}_B^0$), it is first required the calculation of the homogeneous transformation matrix ($\mathbf{T}_6^0 \in \mathbb{R}^{4\times4}$) as shown in equation 4.5

$$\mathbf{T}_n^0 = \mathbf{T}_1^0 \cdot \mathbf{T}_2^1 \cdots \mathbf{T}_{n-1}^{n-2} \cdot \mathbf{T}_n^{n-1} = \begin{bmatrix} (\mathbf{R}_n^0)_{3x3} & (\mathbf{p}_n^0)_{3x1} \\ (\mathbf{0})_{1x3} & 1 \end{bmatrix} \tag{4.5}$$

Due to the kinematic fundamentals in the previous subsection for carrying out the kinematic decoupling of non-spherical wrist cobot, the last three origins of the reference systems coincide and positioned in the selected decoupling point[4]. Consequently, the position of the decoupled point will also be coincident with the previous reference frames origins (see equation 4.6).

4: It means that $\mathbf{O}_{n-2} = \mathbf{O}_{n-1} = \mathbf{O}_n$, so it is also held that $\mathbf{p}_n^0 = \mathbf{p}_{n-1}^0 = \mathbf{p}_{n-2}^0$.

$$\mathbf{p}_B^0(q_1, \cdots, q_{n-3}) = \mathbf{p}_{n-2}^0(q_1, \cdots, q_{n-3}) \tag{4.6}$$

From the equation above (equation 4.6), it is remarkable that the position of the decoupling point is only dependent of the first joints of the manipulator[5].

5: In the case of a 6 DoF robot $n-3=3$, while in the case of a 7 DoF robot $n-3=4$.

On the other hand, computing the minimal parameter expression for the orientation of the robot ($\mathbf{o}_B^0(\mathbf{q})$) requires solving the ZYX Euler rotation angles[6]. Hence, these angles can be computed as displayed in equation 4.7, where n corresponds to the DoFs of the cobot (in this case, the model is presented generically for 6 or 7 DoFs), and the $(R_n^0)_{(i,j)} \in \mathbb{R}^{3\times3}$ terms represent the i-th row and j-th column of the corresponding rotation matrix.

6: The angles of roll (ψ_B^0), pitch (θ_B^0), and yaw (ϕ_B^0), which corresponds to rotations about the X, Y, and Z axes, respectively.

$$\psi_B^0 = \arctan\left(\frac{(R_n^0)_{(3,2)}}{(R_n^0)_{(3,3)}}\right) \tag{4.7a}$$

$$\theta_B^0 = \arctan\left(\frac{-(R_n^0)_{(3,1)}}{\sqrt{(R_n^0)_{(3,2)}^2 + (R_n^0)_{(3,3)}^2}}\right) \tag{4.7b}$$

$$\phi_B^0 = \arctan\left(\frac{(R_n^0)_{(2,1)}}{(R_n^0)_{(1,1)}}\right) \tag{4.7c}$$

Thanks to equations 4.6 and 4.7 the forward kinematic model can be considered fully addressed. Nevertheless note that, once the position and orientation is computed, the relationships from equation 4.2 should be applied to compute the position and orientation of the end effector.

Once the positioning and orienting problems are addressed through the FK, the velocities aspects of the FK can also be handled. The general formulation for FK can be computed as displayed in equation 4.8[7]. This

7: In this equation $\dot{\mathbf{x}}_B^0 \in \mathbb{R}^n$, $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{6xn}$, and $\dot{\mathbf{q}} \in \mathbb{R}^n$.

expression is obtained after deriving the decoupling point pose ($\mathbf{x}_B^0$). For that reason, as well as for position and orientation, the FK velocity model is referred to the decoupling point instead of the end effector.

$$\dot{\mathbf{x}}_B^0 = \mathbf{J}(\mathbf{q}) \cdot \dot{\mathbf{q}} \tag{4.8}$$

Unlike traditional coupled kinematic models, to take into account the end effector influence when computing the velocities for the decoupled point, the computation of the Jacobian matrix should be modified. To do so, the angular velocity of the n-th joint should be decomposed into their components over the auxiliary reference system β[8] (how to compute it is shown below at equation 4.9).

8: Bear in min that $\boldsymbol{\omega}_n^\beta \in \mathbb{R}^{n-3}$, $\mathbf{R}_n^\beta = (\mathbf{R}_{n-1}^n \mathbf{R}_\beta^{n-1})^{-1} \in \mathbb{R}^{3x3}$, and $\boldsymbol{\omega}_n^n \in \mathbb{R}^{n-3}$.

$$\boldsymbol{\omega}_n^\beta = \mathbf{R}_n^\beta \boldsymbol{\omega}_n^n \tag{4.9}$$

Once the components of the angular velocity of the n-th reference system are computed, the unit vector of the joint axis for each joint are required ($\mathbf{z}_{i-1} \in \mathbb{R}^3$). Each joint axis unitary vector represent the contribution into the speed of the manipulator according to the decoupling point for each axis expressed as an unitary vector. Since to maintain and take into account the end effector in the decoupling, the angular velocity of the n-th joint has been decomposed into the auxiliary β reference system. This consideration generates a variation on the Jacobian computation as displayed in equation 4.10 that makes it dependant on the unit vector of the Joint n in the β reference system ($\mathbf{z}_\beta \in \mathbb{R}^3$, computed as displayed in equation 4.11).

$$\mathbf{J} = \begin{bmatrix} \mathbf{z}_0 \times (\mathbf{p}_e - \mathbf{p}_0) & \cdots & \mathbf{z}_\beta \times (\mathbf{p}_e - \mathbf{p}_B) \\ \mathbf{z}_0 & \cdots & \mathbf{z}_\beta \end{bmatrix} \tag{4.10}$$

$$\mathbf{z}_\beta = \mathbf{R}_1^0(q_1) \cdots \mathbf{R}_{n-1}^{n-2}(q_5) \cdot \mathbf{R}_\beta^{n-1}(\beta) \cdot \boldsymbol{\omega}_n^\beta \tag{4.11}$$

By applying the previous equations, the computation of the Jacobian matrix should result into a similar expression of the one displayed in equation 4.12, where a can be $a = 3$ in case of a 6 DoF robot, or $a = 4$ in case of a 7 DoF one. Thus, the linear velocity of the end effector is independent of the last three joints, achieving the desired $\mathbf{0}_{3\times3}$ matrix that simplifies the kinematic behavior of the non-spherical wrist cobot. In this manner, for any given joint configuration a corresponding linear and angular velocity can be computed for the decoupling point.

$$\mathbf{J} = \begin{bmatrix} (\mathbf{J}_{11})_{3\times a} & (\mathbf{0})_{3\times3} \\ (\mathbf{J}_{21})_{3\times a} & (\mathbf{J}_{22})_{3\times3} \end{bmatrix} \tag{4.12}$$

4.3 Inverse Kinematics

The real advantage of a decoupled kinematic model lies in the computations of the Inverse Kinematic (IK) solution instead of the FK formulation. These kind of kinematic models allows the robot control engineers to extract a closed set of analytical solutions for the inverse kinematics.

Allowing the usage of more optimal or specific control solutions for a particular type of manipulator.

In order to compute the aforementioned set of closed solutions for the IK of a generic 6 and 7 DoF non-spherical wrist cobot, the kinematic decoupled model has to be solved to match with the expression displayed in equation 4.3. For such aim, in Figure 4.3 is represented a generic structure for this type of robots from which the kinematic modelling should start.

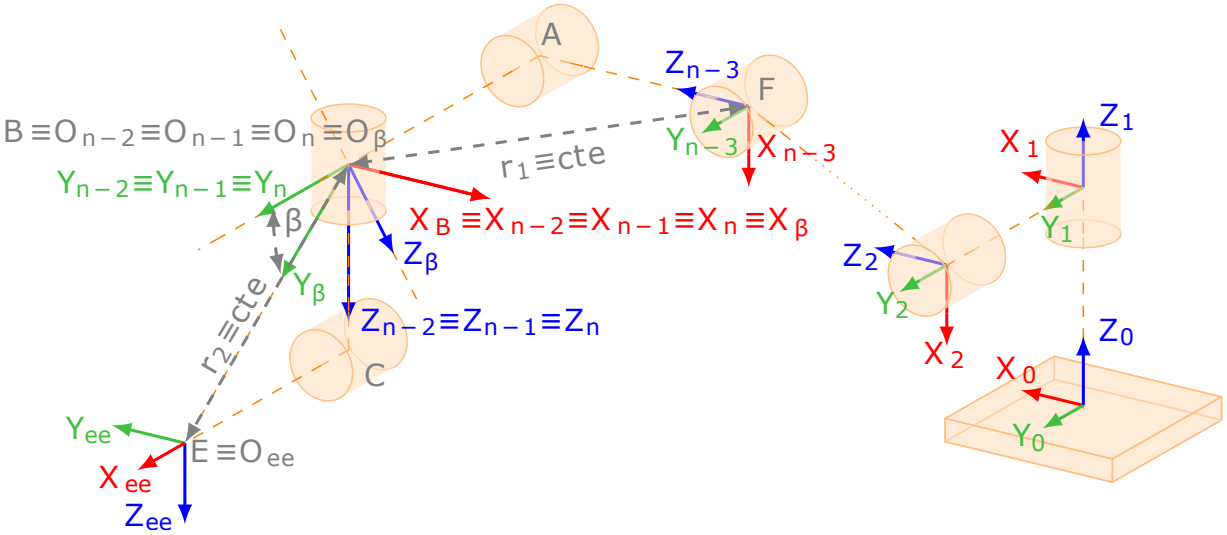

Figure 4.3: Transformed model for kinematic decoupling for a generic 6 or 7 DoF cobots.

In Figure 4.3, it can be appreciated that the criteria of constant radius and a pointing auxiliary reference system to the end effector exposed in Section 4.1 are gathered for the decoupling point **B**. Once the reference systems are placed along the robot structure, the proposed contribution to compute the inverse kinematic model is a method that relies in positioning first the decoupling point, to solve latter the orientation of the decoupling point that should be shared by the end effector too (as generally do in the IK algorithms). To obtain a general formulation for a a 6 ($n-3=3$) or 7 ($n-3=4$) DoF robot, the following steps should be followed:

1. Compute the decoupling point position ($\mathbf{p}_B^0(q_1, \cdots, q_{n-3})$) as in the FK (see equation 4.6).
2. Solve analytically the IK for $q_1, \cdots, q_{n-3}$.
3. Compute the corresponding $\mathbf{R}_{n-3}^0(q_1, \cdots, q_{n-3})$ for the previous step solutions.
4. Compute numerically the $\mathbf{R}_n^{n-3}(q_{n-2}, q_{n-1}, q_n) = (\mathbf{R}_{n-3}^0)^T \cdot \mathbf{R}_n^0$.
5. Solve analytically the IK for (q_{n-2}, q_{n-1}, q_n) as a ZYX Euler rotation.

In order to address the steps from 1 to 3, the ones related to the IK of the decoupling point positioning problem, several examples of how to solve the IK for different manipulator structures can be found in traditional robotics theory references [41, 108, 109]. Consequently, the computation of steps 4 and 5, the ones related to solve the IK of the orientation of the decoupling point, a similar strategy to the one seen at equation 4.7 can be employed. In this way, the Euler rotation angles should be computed referred to $\mathbf{R}_n^{n-3}(q_{n-2}, q_{n-1}, q_n)$[9] where the roll ($\psi_n^{n-3}$), pitch ($\theta_n^{n-3}$), and

9: Note that $\mathbf{R}_n^{n-3}(q_{n-2}, q_{n-1}, q_n)$ should be computed in the 4-th step. In that step, it is fulfilled:

$$\mathbf{R}_n^{n-3}(q_{n-2}, q_{n-1}, q_n) = (\mathbf{R}_{n-3}^0)^T \cdot \mathbf{R}_n^0 = \\ = \mathbf{R}_n^{n-3}(\psi_n^{n-3}, \theta_n^{n-3}, \phi_n^{n-3}).$$

yaw (ϕ_n^{n-3}) angles correspond to the last three joints (q_{n-2}, q_{n-1}, q_n) as shown in equation 4.13.

$$[q_{n-2}, q_{n-1}, q_n]^T = [\psi_n^{n-3}, \theta_n^{n-3}, \phi_n^{n-3}]^T \tag{4.13}$$

Addressing the step 5 and the equation 4.13 the IK for the position and orientation problem can be considered fully solved.

Similar to the strategy followed in the previous section, Section 4.2 Forward Kinematics, the velocity aspects of the proposed decoupled kinematic model for non-spherical wrist cobot should also be addressed. In this case, the velocity IK expression can be computed through the equation 4.14.

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1} \cdot \dot{\mathbf{x}}_B^0 \tag{4.14}$$

Therefore, to compute a response based on the inverse kinematics, the Jacobian matrix computed with equation 4.10 must exist. Otherwise, the computation of the joint velocities could not be possible. In order to assure the existence of the jacobian matrix, the determinant of the Jacobian should differ from zero ($\exists \mathbf{J}^{-1} \iff |\mathbf{J}| \neq 0$). This fact leads to the problem and discussion of whether the Jacobian exists or not. Those configurations where the inverse of the Jacobian matrix cannot be computed because the determinant is zero are known as singular configuration. A first study of this configurations, will be theoretically discussed in more depth in the following section.

4.4 Differential velocity limitations

As hinted at the end of the previous section, the IK can only provide a solution whenever the solution exist. In other words, for the IK closed set of solutions computed, there are more than a single configuration that satisfies the Cartesian position. If this happens, the robot misbehaves, leading to excessive velocities or even, stopping the robot movements to make the safety prevail. These configurations where the IK has no solution are the singular configurations.

The knowledge of this configurations beforehand is relevant to avoid excessive velocities or forces that could risk both the operators safety and the task execution performance. Therefore, this section presents the analysis of the behavioral implications of selecting the decoupling point as the kinematic reference instead of the end effector, as traditionally done for non-spherical wrist cobot. Moreover, it also faces the problem of knowing beforehand the singular configurations of this type of robots in a generic manner that allows its latter particularization for two different models. This two contributions helps in understanding how the robot should move and which configurations might block the robot due to trespassing the safety thresholds of the application compared to traditional non-spherical wrist robot kinematic models.

4.4.1 Velocity behavior and implications

There are several aspects to be discuss about the theoretical model and its usage implications of the proposed decoupled kinematic model. This section will start with the most trivial concerns about the positioning and orienting problems from the FK, moving on to the discussion of the FK model of velocity, and ending with the implications for solving the inverse kinematics of position and velocity. In this manner, this subsection pretends to highlight the benefits and drawbacks of employing the decoupling proposed model instead of a traditional one for non-spherical wrist cobots.

Firstly, the proposed approach is suitable to apply to real industrial manipulator offering a model that is expected to compute efficiently the position and orientation of the collaborative manipulator for a given set of joint positions. In this manner planning techniques can be efficiently applied to the decoupled kinematic model to compute the trajectories to track. It is also remarkable that contrary to what happens with traditional approaches as the DH convention based ones, the position and the orientation are computed according to the decoupling point instead. It is worth to highlight that in general the decoupled FK models are more efficient in computation than its analogue coupled models, since the desired posed and orientation of the TCP can be calculated independently. Therefore, this behavior is also expected to be appreciated in Chapter 8.

In addition, it has been obtained a simplified FK model formulation for the velocity of the manipulator according to the decoupling point[10]. Relative to this simplified velocity kinematic model, the first remarkable fact is the independence from q_4 when computing velocities due to the $\mathbf{0}_{3\times3}$ block matrix from the Jacobian. It means that the first wrist joint would not contribute to the lineal velocity of the decoupling point, unlike other traditional model does. This is not necessarily a misbehavior of the kinematic model, but it is relevant to bear in mind while designing robot tasks. Consequently, the position and orientation decoupling is also a characteristic of computing the velocities for the decoupled kinematic models. In other words, the lineal velocity of the manipulator depends only on the arm positioning related joints, leaving the angular velocity to be handled additionally by the last three ones to move all the robot structure as a whole[11].

This chapter also presents a general inverse kineamtic modelling for non-spherical wrist cobots. In Chapter 7 an example that particularized the proposed kinematic decoupled model for the UR10e and the SUPSI robot is presented. However, due to the diversity of different robot structures, the solutions presented in this work might slightly vary from the ones for other robot structures. Moreover, by following the steps presented in Section 4.3 it can be decoupled the position and the orientation for the IK too. In this manner, the problem of dimensionality when the robot present a high order of DoF is reduced, enabling the computation of a closed set of analytical IK solutions. Fact that seems very convenient to compute more efficiently the desired joint configurations while applying task space planning techniques.

Lastly, the velocity IK model[12] main problem relies in the study of the

10: Remembar that the velocity model relies on the jacobian computation, which can be expressed generically as:

$$\mathbf{J} = \begin{bmatrix} (\mathbf{J}_{11})_{3\times a} & (\mathbf{0})_{3\times3} \\ (\mathbf{J}_{21})_{3\times a} & (\mathbf{J}_{22})_{3\times3} \end{bmatrix},$$

where a can be $a = 3$ in case of a 6 DoF robot, or $a = 4$ in case of a 7 DoF one.

11: Attention: It does not mean that the three first joint does not contribute to the angular velocity of the manipulator. But the contribution to the angular velocity is a mixed of all the joints of the robot. So the movement of all robot structure as a whole is still possible and coordinated.

12: Recalling from equation 4.14:

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1} \cdot \dot{\mathbf{x}}_B^0$$

.

singular configuration that will be addressed in the following subsection. However, the fact that the Jacobian matrix has a $\mathbf{0}_{3\times3}$ block matrix, simplifies the computation required to calculate a velocity-based response for the controllers[13]. This last issue also reduces the demanded computational power when computing the singular configurations of the manipulator, and enables the computation of a closed set of singular configurations to be applied online more efficiently.

13: In order to check the existence of the Jacobian matrix, the required computation is simplified because the determinant of the Jacobian can be reduced as shown below:

$$|\mathbf{J}| = |\mathbf{J}_{11}| \cdot |\mathbf{J}_{22}|$$

The equation above, will be better detailed in the subsection 4.4.2.

4.4.2 Singularity general study

As stated in the previous subsection, Velocity behavior and implications, the main problem of the velocity IK is the singularity handling. One advantage the decoupled kinematic model present over other approaches for non-spherical wrist cobot is that the singular configurations can be computed beforehanded[14] to be more efficiently applied online[15]. In the following paragraphs, the demonstration to prove that all singularities are taken into account by using the proposed kinematic model is presented for both 6 and 7 DoF cobot.

14: Durign the modeling design phases of a robotic application.

15: As the singular configurations are known beforehanded, more optimal handling strategies can be applied to avoid those configurations.

In both situation, the theoretical study of singularities starts with equation 4.15:

$$|\mathbf{J}| = \left| \begin{bmatrix} (\mathbf{J}_{11})_{3\times a} & (\mathbf{J}_{12})_{3\times3} = (\mathbf{0})_{3\times3} \\ (\mathbf{J}_{21})_{3\times a} & (\mathbf{J}_{22})_{3\times3} \end{bmatrix} \right| \tag{4.15}$$

For a 6 DoF robot ($a = 3$), since the $(\mathbf{J}_{12})_{3\times3} = (\mathbf{0})_{3\times3}$, the singular configurations dependent of the Jacobian determinant can be computed as follows (see equation 4.16):

$$|\mathbf{J}| = |\mathbf{J}_{11} \cdot \mathbf{J}_{22} - \mathbf{0} \cdot \mathbf{J}_{21}| = 0 \Rightarrow |\mathbf{J}| = |\mathbf{J}_{11}| \cdot |\mathbf{J}_{22}| = 0 \tag{4.16}$$

In equation 4.16 can be observed that the Jacobian of the decoupled model will only be singular whenever the $|\mathbf{J}_{11}| = 0$ or the $|\mathbf{J}_{22}| = 0$.

Moreover, for the case of 7 DoF ($a = 4$), a similar strategy can be followed as shown in equation 4.17:

$$|\mathbf{J}| = |(\mathbf{J}_{11})_{3\times4}| \cdot |(\mathbf{J}_{22})_{3\times3} - (\mathbf{J}_{21})_{3\times4} \cdot (\mathbf{J}_{11}^{-1})_{4\times3} \cdot \mathbf{0}| = 0 \Rightarrow \tag{4.17a}$$

$$\Rightarrow |\mathbf{J}| = |(\mathbf{J}_{11})_{3\times4}| \cdot |(\mathbf{J}_{22})_{3\times3}| = 0 \tag{4.17b}$$

As in the previous case, the Jacobian model will only be singular whenever the $|(\mathbf{J}_{11})_{3\times4}| = 0$ or the $|(\mathbf{J}_{22})_{3\times3}| = 0$. Since the determinant of a non-square matrix does not exist, someone might think that there is no solution for the case of $\mathbf{J}_{11}$. However, as it is only desired to know when the model does not have an inverse,the pseudoinverse Jacobian is used as in for the manipulability. Therefore, for the case $|(\mathbf{J}_{11})_{3\times4}| = 0$, it will be equivalent to check whether $\sqrt{|\mathbf{J}_{11}\mathbf{J}_{11}^T|} = 0$. So the singular configurations can also be computed and parameterized for this use case.

It is remarkable that even though it could happen that $\mathbf{J}_{11}$ and $\mathbf{J}_{22}$ can be singular at the same time, they do not represent coupled singularities.

In other words, by chance, it can happen a dual singular configuration, but each one is independent and not bounded to the other. Moreover, due to the simplifications into block matrices and the fact that the decoupled kinematic model seeks for nullifying one of those block matrices ($(\mathbf{J}_{12})_{3\times3} = (\mathbf{0})_{3\times3}$), it can be assured that the velocity behavior of the proposed model is different from the traditional of the literature. With this mathematical characterization of the kinematic model a more simplified robot velocity and position kinematics is obtained allowing the extraction of a closed set of singular configurations as displayed in Chapter 7 for two different robots.

4.5 Conclusion to Chapter 4

In this chapter a suitable expression for the position and velocity FK and IK model has been explained in the form of mathematical fundamentals and general formulation to be applied to any 6 or 7 DoF non-spherical wrist cobots. The proposed model seems to be accurate enough to reduce the computational performance of current collision avoidance techniques under the right algorithm. As this section focuses on theoretical aspects of the kinematic model proposed, all the statements given by the discussion of Section 4.4 will be mathematically addressed in the particularization of Chapter 8. Lastly, the singular configuration not only can be parameterized due to simpler expression of the Jacobian matrix, but it is also guaranteed that the singular configuration of the arm are independent of the wrist ones and limited in both cases.

Controller Design and Implementation

Obstacle and Singularity Handling Reference Controller

5

As discussed in Chapter 2, State of the Art and Objectives, current singularity handling methods combined with collision avoidance controllers cannot guarantee avoiding singular configurations that activates the emergency safety stops of cobots. Due to that reason, the prior chapter (Chapter 4) develops a novel kinematic decoupling technique, the wrist spherification, to achieve a decoupled kinematic model for non-spherical wrist cobots.

In order to test the advantages of utilizing the decoupled kinematic model proposed in this work, this chapter exposes the design principles of the reference controller. This reference controller is based on currently used DLS kinematic based controller for collision avoidance. It will also be the reference controller to which the performance of the d-APF controller explained in Chapter 6 will be tested against. Another particularity of the reference controller proposed in this chapter is that it is based on the developments of Flacco and De Luca from [18]. In this way, the reference controller corresponds to the starting point from which the proposed d-APF is developed (detailed in Chapter 6). The common aspects of both controllers leads to a more accurate comparison of the controllers behavior (addressed in Chapter 8).

Therefore, this chapter explains the requirements and the control architecture employed to design the reference controller. These design principles explains the adaptation of the Flacco and De Luca approach from [18] to fit a DLS kinematic model while avoiding obstacles. Since the employed hardware is not the same as they utilized, some modifications have been also made to the algorithms to compute the closest distance to the obstacle, and how the robot dynamics are handled. This modified version of the controller that is utilized as the reference controller is called DLS-APF controller and each of the required variations are covered from the theoretical design point of view in the following sections.

5.1 Design Basics

In this section, the design requirements are presented as well as an overview of the proposed control architecture that allows the latter implementation of the controller in Chapter 7. In this manner, the design requirement presents the criteria to bear in mind to achieve not only a well performance of the controller but also a safe and trustworthy human-robot shared environment. Additionally, the control architecture section details the adjustments made on the control loop and the functionalities of each of these modified control blocks, leaving the software architecture aside for Chapter 7.

5.1.1 Requirements

The requirements presented in this section are exclusively related to the developed controller requirement and the collaborative application. Since auxiliary components as the vision systems are used as an additional tool to test the controllers but they are not a goal of the Ph.D., their requirements are not exposed[1]. Therefore, the requirements that the controller have to meet are the following:

- **Guarantee of safety at any time**: the implemented controller cannot violate the instrincis safety of the commercial cobot.
- **Real time control cycle**: the controller execution frequency must be adequate to respond to robot application risks by communicating with the robot controller without dropping any message (at least of 500 Hz[2]).
- **Avoiding any type of collision or non-critical safety stops**: additional risk, collisions or singularity blocking conditions due to the closest obstacle avoidance are not allowed.
- **Adequate to perception and processing update rates**: the perception data processing time[3] must be lower than the environment perception time[4], and the controller execution frequency must be lower than both[5] to provide enough information of the environment each control cycle.
- **The controller should allow the simulation of the robot behavior**: the controller should also be a valid one for its Gazebo and MoveIt[6], allowing the simulation testing without risking components.
- **Easy configurable collision avoidance parameters**: the specific parameters as the maximum repulsive velocity must be easily tuneable to adapt the controller to different collaborative applications.
- **Tracking of the precomputed offline trajectory**: while the robot is not repelling a collision, the trajectory tracking during robot task should minimize position and velocity errors.

Complementary to the controller requirements, the application requirements should fulfill the following criteria:

- **Safe for the operator**: under any circumstances the operator might result endangered or harmed by any robot trajectory tracking or collision avoidance movement.
- **Movement without endangering other components**: it means that the application must allow robot movements without disregarding the integrity of other device or sensor in the scene.
- **Increase the efficiency of manual tasks**: the process must be faster than the manually done industrial task, though it is slower than a fully automated one.
- **Avoidance of non-critical safety stops**: when the integrity of an operator is not at risk, the controller safety breaks should not be triggered.
- **Modular and integrated application**: the solution requires to be versatile and flexible, allowing the quick integration of new components[7] and enabling the synchronization with other machines.

By addressing both types of requirements the application and controller requisites can be handled, enabling an efficient and safer interaction between human and robots. In the section below, the proposed control

1: More or less, it is expected to update the scene status between 5 and 10 times each second, at least.

2: This requirement is obtained from [37] datasheet. Another way to state this requirement is the following:

$$f_{ctrl} < 1KHz.$$

3: This timing parameter is referred to the required amount of time to process the raw data given by the different sensors of the scene.

4: This timing parameter is referred to the time required by the external sensors to obtain a full cycle of information of the robot environment.

5: In a more compact statement, the requirement can be summarized as follows:

$$f_{perc} > f_{proc} > f_{ctrl}.$$

6: More specifically that a MoveIt Digital Twin, it is referred to the pseudo-Digital Twin generated in the ROS Parameter Server. However, as it is MoveIt the one in charge of the robot kinematics, it has been employed MoveIt in the main text instead.

7: A component can either be a software functionality or new sensor information of the collaborative task.

architecture proposed to fulfill all the presented requirements will be detailed.

5.1.2 Control Architecture

The main modification made to the controller proposed at [18] is related to the usage of a DLS kinematic model instead for the APF controller proposed by those authors. This controller that combines a DLS kinematic model whithin a APF controller for collision avoidance is denoted as DLS-APF controller. This proposed control architecture can be appreciated in Figure 5.1. As shown in this figure, the controller is divided into five big blocks of functionalities, each one corresponding to one specific color.

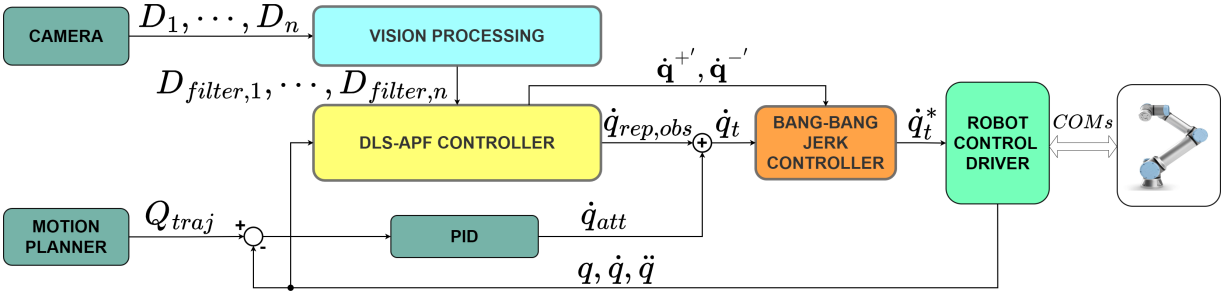

Figure 5.1: Overview of the control diagram for the reference DLS-APF controller for collision avoidance and singularity handling.

From the five colour code, the control boxes coloured in dark green include auxiliary capabilities for the DLS-APF controller such as the camera driver manager, the trajectory planner, and the high level PID controller to guarantee following a trajectory. The aim of this control boxes is to process the environment of the robot and generate offline computed collision-free trajectories to plan the reference trajectories for the robot operation. These control boxes are the one in charge of feeding the controller with the reference trajectory in the joint space($\mathbf{Q}_{traj} = \{\mathbf{q}_{traj}, \dot{\mathbf{q}}_{traj}, \ddot{\mathbf{q}}_{traj}\}$), the different raw depth maps and color images taken by the camera ($\mathbf{D}_1 \cdot \mathbf{D}_n$), and the attractive component of the APF ($\dot{\mathbf{q}}_{att}$) obtained from a negative feedback control loop of the reference trajectory and the current state of the robot ($\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ for the angular position, velocity and acceleration, respectively).

The light green coloured box is similar to the previous group because it corresponds to the capabilities to command a specific robot. In other words, it corresponds to the specific driver of the robot that communicates a PC-based control architecture with the robot controller. Therefore, it is in charge of creating a specific communication with the robot that fits the application and its requirements. As latter exposed in Chapter 7, two different drivers are going to be used for each of the uses cases (the UR10e robot and the SUPSI robot).

In the case of the light blue box, the aim of this controller components is to execute all the tasks relative to the environment segmentation. The aim of this control box is to first filter the robot from the images and then, analyze the remaining environment to segment the rest of the elements of the picture depending on the desired interaction. In this way, the static elements of the scene, the work piece and the obstacle can be discerned to obtain different depth maps ($\mathbf{D}_{filter,1} \cdots \mathbf{D}_{filter,n}$) to allow

the interaction in the first two cases and avoid the collision in the last case. A more in-depth view is given at the Subsection 5.2.1.

The yellow box corresponds to the DLS-APF Controller. This element of the control loop computes the distance to the obstacle, the collision risk index, the repulsive vector and the compensation due the DLS kinematic model (addressed at Subsections 5.2.2, 5.2.3, 5.2.4, and 5.2.5, respectively) in order to generate a suitable and reachable response for the robot for avoiding obstacles. From the signals mentioned earlier, this component feeds the control loop with the different repulsive joint velocities due to the obstacle ($\dot{\mathbf{q}}_{rep,obs}$), and top and bottom thresholds for limiting the joint velocities to assure the safe operation ($\dot{\mathbf{q}}^{-'}$, $\dot{\mathbf{q}}^{+'}$).

Lastly, the orange coloured box is related to the dynamics handling control components. The aim of this control block is to introduce the dynamics of the robot by limiting the torques, acceleration and velocities of the robot according to its specifications. Therefore, the reactive kinematic response is accommodated to robot reachable velocities for each control cycle through a Bang-Bang Jerk Controller. Further details of how this component works are given at Subsection 5.2.5 Damped Least-Square (DLS) Kinematic Model.

5.2 APF Based Obstacle Avoidance Controller

In this section the modifications with respect to the Flacco and De Luca controllers are explained. The contributions can be summed up into a new scene segmentation algorithm to take advantage of the distance measurement method, the usage of a DLS kinematic model instead, the computation of the repulsive vector and its limiting thresholds, and the new dynamics handler employed. Thus, this section covers the general formulation and theoretical commonalities of the improvements proposed for the reference controller. All the specific implementation details of each component are addressed in Chapter 7.

5.2.1 Advanced Scene Segmentation

An adequate scene segmentation is a key component for the success of the APF controller, though it is not an research objective of the Ph.D. Without it, these type of controllers can misbehave, resulting in risk increase for the operator. Since the particular implementation and available cameras determine the selected vision algorithm, the specific vision algorithm employed will be detailed in each implementation of Chapter 7. Therefore, this section exposes the general considerations of the vision based filter and segmentation that the selected algorithm must gather.

The Advanced Scene Segmentation algorithm aims to both: obtaining a new depth map from which the robot is already filtered ($\mathbf{D}_1^*, \cdots, \mathbf{D}_n^*$) from the original pictures ($\mathbf{D}_1, \cdots, \mathbf{D}_n$), and utilizing the robot filtered depth map to extract a set of new depth maps with relevant features for the application ($\mathbf{D}_{filter,1}, \cdots, \mathbf{D}_{filter,n}$). Therefore, the control loop from Figure 5.1 can be more detailed expanding the functionalities as displayed in Figure 5.2.

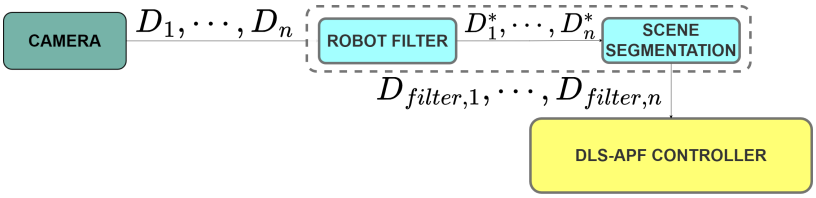

Figure 5.2: Detail on the Vision Processing control component for the DLS-APF controller.

In this manner, the first challenge that the Advanced Scene Segmentation algorithm faces is to filter the robot from the depth scene. As seen in the Chapter 2 State of the Art and Objectives, the main problem filtering the robot lies in the fact that it corresponds to an element of the scene that can be dynamic or static according to the part of the manufacturing task that is executing. This duality bounded to the randomness of a reactive collision avoidance environment forces the usage of Digital Twin based filters or Artificial Intelligence (AI) based filter to delete the robot from the colour or depth image [64].

Subsequently, the depth map should be processed to extract from the remaining elements of the scene the features as how many and where are the workpieces or static elements, or where and how are the obstacles moving[8]. This type of image processing filter usually combines colour images with depth images to generate a segmentation similar to the one appreciated in Figure 5.3. In this Ph.D., at least the information relative to the dynamic obstacles have to be extracted from the depth map to feed the controller with it in order to avoid them as risky objects that compromises the safety of the application. However, it is also convenient to segment the workpieces and the static elements of the scene to allow the manipulation and the approach of the robot without being repelled[9]. More detail about how the advanced scene segmentation filter can be found in Chapter 7 while describing the implemented vision algorihtms.

8: This last group, is the one where the operators of a manufacturing process are included

9: In case this last fact is not possible, they should be included in the obstacle related depth map to avoid crashes that can break the robot, the gripper or any other element or sensor.

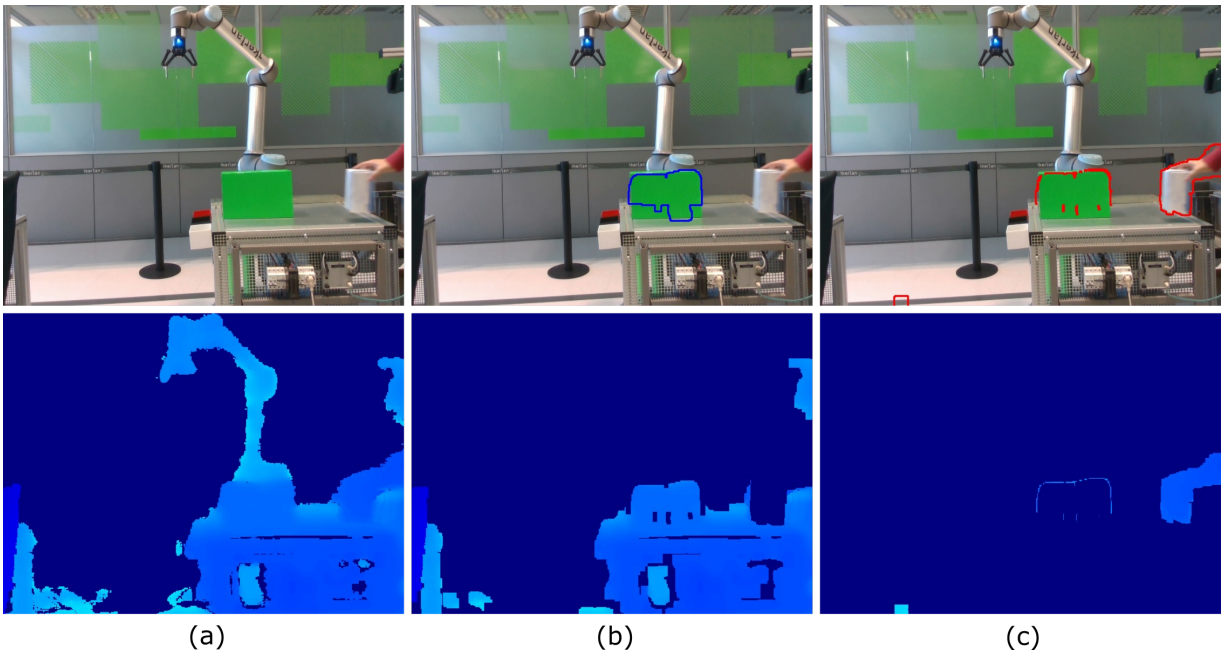

Figure 5.3: Example of an advanced scene segmentation of the robot environment: (a) Raw data obtained from the camera sensor. (b) Static elements of the scene and workpieces segmentation. This segmentation helps in selecting which elements of the scene are static to interact with them for planning and grasping tasks (c) Dynamic obstacle (operator) segmentation. In this control algorithms, they are considered as the obstacles to be avoided.

Once this image processing has been done, the obtained depth map are fed to the DLS-APF Controller component to interact as desired with the environment.

5.2.2 Distance to the Obstacle Computation

The first of the components in which the DLS-APF Controller component can be decomposed is in charge of computing how far the obstacle is to the robot through the information given by the aforementioned segmented depth maps. A depth map image is a projective space in $\mathbb{R}^3$ that represents the projection of a Cartesian point onto a plane and the distance between the point and the plane (see Figure 5.4). The conversion is facilitated by the intrinsic an extrinsic parameters camera parameters. On the one hand, the intrinsic parameters corresponds to the focal lengths for both axes (f_{s_x}, f_{s_y}) and the optical center pixel coordinates (c_x, c_y). On the other hand, the extrinsic parameters correspond to the rotation matrix between the camera reference and the global reference frame ($\mathbf{R}_r^c \in \mathbb{R}^{3\times3}$), and the translation between both reference frames ($\mathbf{t}_r^c \in \mathbb{R}^3$). By utilizing the intrinsic parameters (the projection matrix specifically, $\mathbf{P} \in \mathbb{R}^{3\times3}$) the transform between a point in the optical frame and the camera frame can be computed as follows:

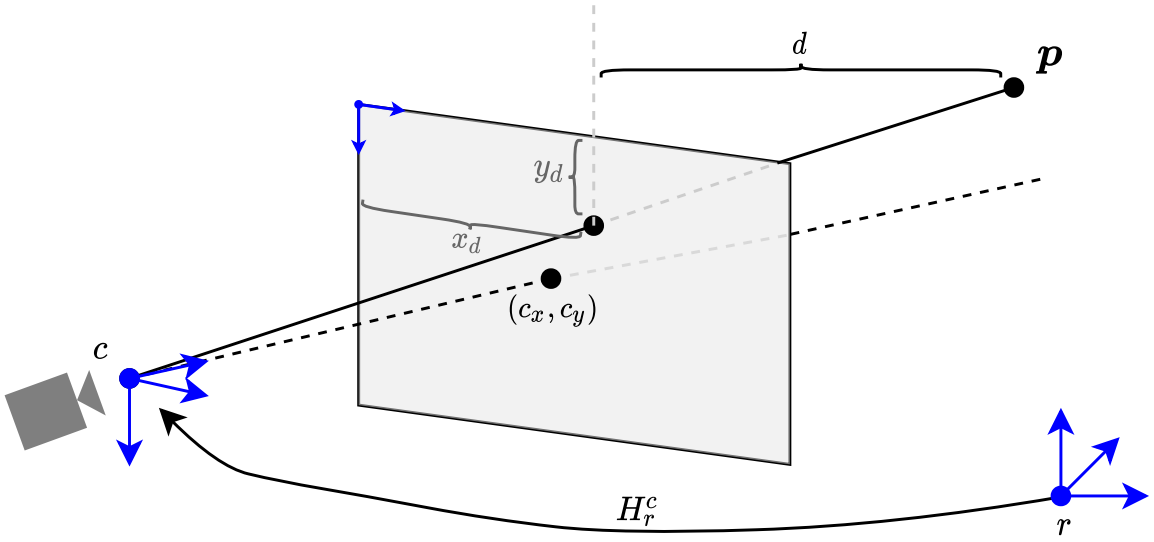

Figure 5.4: Representation of the depth space within the 3D Cartesian space, illustrating the projection process and frames involved.

$$\mathbf{P} = \begin{bmatrix} f_{s_x} & 0 & c_x \\ 0 & f_{s_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1a}$$

$$x_d = \frac{x_c f_{s_x}}{z_c} + c_x,\ y_d = \frac{y_c f_{s_y}}{z_c} + c_y,\ d = z_x \tag{5.1b}$$

Furthermore, with the extrinsic parameters the position of a point according to the optical frame can be calculated thanks to the transformation between the optical frame and the camera ($\mathbf{H}_r^c \in \mathbb{R}^{4\times4}$):

$$\mathbf{H}_r^c = \begin{bmatrix} \mathbf{R}_r^c & \mathbf{t}_r^c \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{p}_c = \mathbf{R}_r^c \mathbf{p}_r + \mathbf{t}_r^c \tag{5.2}$$

In addition to the segmented environment depth images, the controller lies on the Control Points (cp_i) to compute the distances to the obstacle. The control points are spots virtually positioned equidistantly and distributed along the robot structure that will help in measuring the distances between the robot and the different obstacles that surrounds it. Figure 5.5 includes an example of distances and Control Points distribution to help understanding this concept. Therefore, the computed distances to feed the DLS-APF Controller are computed according to the distance from the obstacle to i-th control point in the depth map.

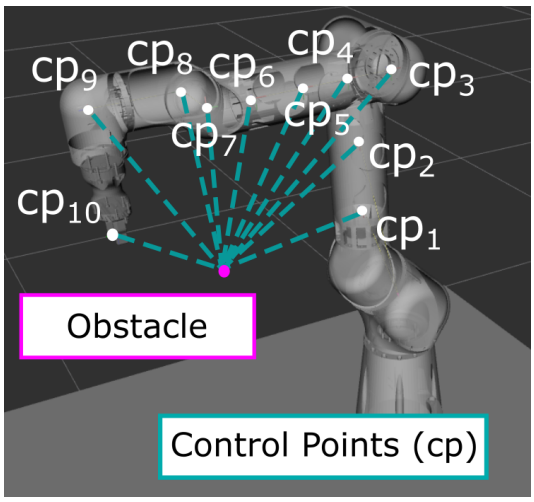

Figure 5.5: Example of Control Points distribution along the SUPSI robot.

Since the extrinsic and intrinsic parameters of the camera are known, the transform of the Control Points reference frames position (from the ROS Parameter Server virtualization) to the depth space is straightforward to obtain. In this manner, the pixel location of the Control Points is known in the depth space. The main consideration to compute these distances ($\mathbf{d} = (d_x, d_y, d_z)$) from a depth image are the occlusions as only the first pixel is read in the depth image. To come up with a solution, Flacco et al. solve this issue by computing the distance to the pixel projection line instead [63]. Therefore, the collision will be likewise avoided because if the obstacle is occluded, the minimum distance cannot be lesser than the distance with the closest projection line that masks the obstacle behind the pixel. The calculation of the distance between the obstacle ($\mathbf{o} = (o_x, o_y, o_z)$) and the control point ($\mathbf{p} = (p_x, p_y, p_z)$) is computed as indicated in equation 5.3 (visually represented in Figure 5.6).

$$d_x = \frac{(o_x - c_x)o_d - (p_x - c_x)p_d}{f_{s_x}} \tag{5.3a}$$

$$d_y = \frac{(o_y - c_y)o_d - (p_y - c_y)p_d}{f_{s_y}} \tag{5.3b}$$

$$d_z = o_d - p_d \tag{5.3c}$$

A problem with this approach is that modelling an obstacle as a single point leads to inaccurate distance measurement as identified in their

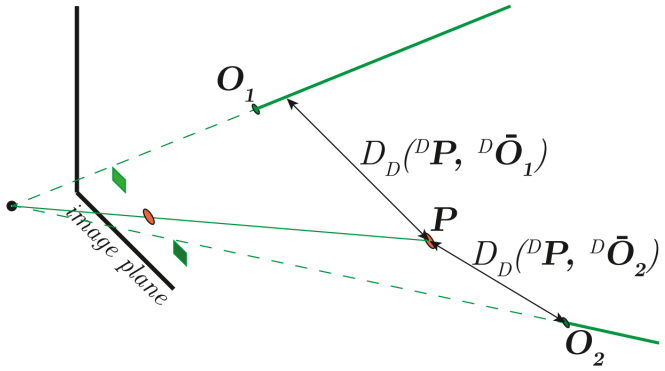

Figure 5.6: Depth space distance evaluation by pixel center projection line. Showing examples of the two possible cases $o_d < p_d$ ($\mathbf{O}_1$) and $o_d \geq p_d$ ($\mathbf{O}_2$). Figure from [63].

second work [18]. To solve this issue, in their later work they propose to compute the minimum distance to the occluded volume by the pixel (shaped as a rectangular frustum). Thus, the closest distance to the obstacle is computed on a two steps process which computes first the closest subpixel along the edge of the 2D pixel grid ($\hat{\mathbf{o}} = (\hat{o}_x, \hat{o}_y, \hat{o}_z)$) and, subsequently, substitutes the original $\mathbf{p} = (p_x, p_y, p_z)$ from equation 5.3 by $\hat{\mathbf{o}} = (\hat{o}_x, \hat{o}_y, \hat{o}_z)$. This two-step process is visualized in Figure 5.7 while the computation of $\hat{\mathbf{o}}$ is observed in equation 5.4.

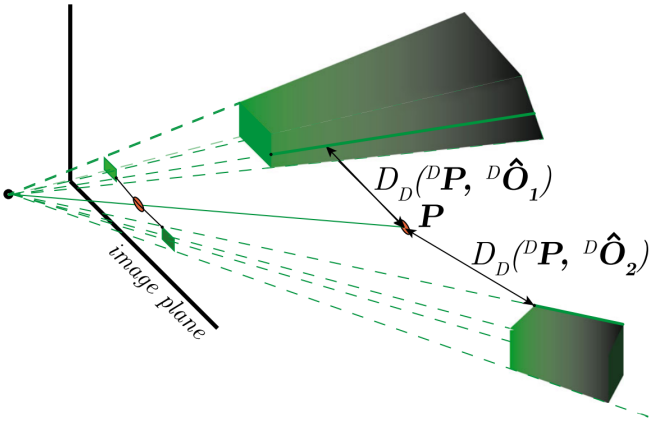

Figure 5.7: Depth space distance evaluation by pixel frustum projection. Showing examples of the two possible cases $\bar{o}_d < p_d$ ($\hat{\mathbf{O}}_1$) and $\bar{o}_d \geq p_d$ ($\hat{\mathbf{O}}_2$). Figure from [18].

$$\hat{o}_x = \begin{cases} \hat{o}_x & \text{if } p_x < \bar{o}_x \\ \hat{o}_x + 1 & \text{if } p_x > \bar{o}_x + 1 \\ p_x & \text{otherwise} \end{cases} \tag{5.4a}$$

$$\hat{o}_y = \begin{cases} \bar{o}_y & \text{if } p_y < \bar{o}_y \\ \bar{o}_y + 1 & \text{if } p_y > \bar{o}_y + 1 \\ p_y & \text{otherwise} \end{cases} \tag{5.4b}$$

$$\hat{o}_d = \bar{o}_d \tag{5.4c}$$

Once these distances are computed, to perform a more efficient depth space search of the closest obstacle, it has been defined a region of interest (**ROI**) around each Control Point. In this manner, instead of searching in the whole depth map for the obstacle, it only will be searched in the vicinity of the Control Point according to a configurable maximum

distance parameter (ρ). This region of interest is defined as a frustum in the depth space whose corners are given by equation 5.5.

$$\mathbf{ROI} = [p_x - \frac{x_s}{2}, p_x + \frac{x_s}{2}] \times [p_y - \frac{y_s}{2}, p_y + \frac{y_s}{2}] \times [p_d - \rho, p_d + \rho] \tag{5.5a}$$

$$\text{where:} \quad x_s = \rho \frac{f_{s_x}}{p_d - \rho}, \quad y_s = \rho \frac{f_{s_y}}{p_d - \rho} \tag{5.5b}$$

Lastly, it is convenient to note that the distances still need to be checked to be smaller than ρ, otherwise the region of interest in the depth map can include pixels further from the limit depth value. Once these distances are computed, they are shared with the Collision Risk Indexes and the Repulsive Component control block as shown in the following subsections.

5.2.3 Collision Risk Indexes Computation

The collision risk indexes (f_{cp_i}) are one of the main parts that compose the DLS-APF controller proposed. The collision risk indexes are the responsible to limit the joint velocities[10], preventing the robot body from colliding accidentally with another obstacle of the scene while avoiding the collision with the closest obstacle to the TCP. These indexes are nourished by the information given from the Distances Computation component, more specifically they utilize the minimum distance to the closest obstacle from the i-th Control Point (see equation 5.6).

10: The upper and lower limit joint velocities are denoted as $\dot{\mathbf{q}}^{+'}$, and $\dot{\mathbf{q}}^{-'}$, respectively

$$f_{cp_i} = \frac{1}{1 + e^{(d_{min}^{cp_i} 2/\rho - 1)\alpha}} \tag{5.6}$$

After computing the collision risk indexes, their influence have to be applied into each joint they are related to. To do so, the collision risk indexes have to be scaled by the minimum distance unitary vector to each control point ($\mathbf{d}_{min}^{*,cp}$). Subsequently, they are also projected into the joint space through the Jacobian of each of the control point ($\mathbf{J}_{cp_i}$)[11]. Both operation have been joined below into a single step in equation 5.7 where the influence of each control point into the joint limits ($\mathbf{s}_{obs}$) is computed.

11: In other words, each risk index computed can only limit the previous joints from where it is place in the robot structure.

$$\mathbf{s}_{obs} = \mathbf{J}_{cp_i} \mathbf{d}_{min}^{*,cp_i} f_{cp_i} \tag{5.7}$$

Finally, the joint velocity limits are then obtained for the i-th joint thanks to combination the specific collision risk index, the joint risk influence, and the design joint velocity limit to guarantee operator safety[12] (see equation 5.6. The computed joint velocity limit will downscale the computed response given by the repulsive vector.

12: Corresponding to q_i^+ or $\dot{q}_i^-$ whether it is referred to the upper or the lower limit.

$$\begin{aligned} \text{if} \quad s_i \geq 0, \quad & \dot{q}_i^{+'} = \dot{q}_i^{+}(1 - f_{cp_i}), \\ \text{else}, \quad & \dot{q}_i^{-'} = -\dot{q}_i^{+}(1 - f_{cp_i}) \end{aligned} \tag{5.8}$$

The computation of the joint velocity limits (equations from 5.6 to 5.8) can be express as a control block diagrams as shown in Figure 5.8.

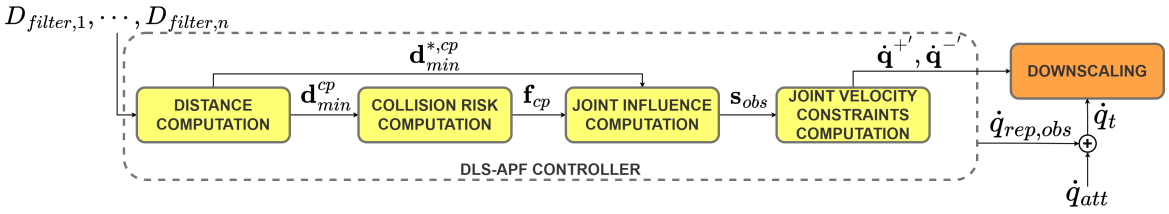

Figure 5.8: DLS-APF Controller: collision risk index computation blocks.

5.2.4 Repulsive Component Computation

The repulsive component or the repulsive joint velocity due to the obstacle avoidance ($\mathbf{v}_{rep,obs} \in \mathbb{R}^3$) aims to generate a virtual repulsive velocity field which pushes the robot away from the path of the obstacle to avoid. Since the repulsive vector ($\mathbf{t}$) is computed in the task space[13], the obtention of the repulsive joint velocity requires the Inverse Kinematic (IK) of the manipulator. In this way, the repulsive joint velocity vector also relies in the pseudo-inverse Jacobian ($\mathbf{J}^{\dagger}$) from the DLS kinematic model (further addressed in Subsection 5.2.5) as displayed in equation 5.9.

13: As the repulsive vector comes from the depth map computed distances and the easiest way to interpret it is a twist according to the closest distance to the TCP.

$$\dot{\mathbf{q}}_{rep,obs} = \mathbf{J}(\mathbf{q})^{\dagger}\mathbf{t} \quad \text{where:} \quad \mathbf{t} = \begin{bmatrix} \mathbf{v}_{rep,obs} \\ 0 \end{bmatrix} \tag{5.9}$$

The general approach to compute the repulsive vector, is by using the direction of the average location of the surrounding obstacles and the magnitude based on the minimum distance among all obstacles. The average distance ($\mathbf{d}_{mean}$) is computed as the mean distance taking into account all the distances between the outside pixels of the obstacle in the depth map and the TCP ($\mathbf{D}_{ee}$) as calculated in equation 5.10. The average distance vector is also employed to compute the unitary vector of the repulsive velocity, since they coincide (i.e. $\mathbf{v}_{rep}* = \mathbf{d}^{*}_{mean}$). This averaging strategy has been selected to make the system more resistant to noise when the end effector is partially surrounded.

$$\mathbf{d}_{mean} = \frac{\sum_{\mathbf{d}\in\mathbf{D}_{ee}} \mathbf{d}}{n(\mathbf{D}_{ee})} \tag{5.10}$$

14: $\mathbf{d}_{min} = \min_{\mathbf{d}\in\mathbf{D}_{ee}}|\mathbf{d}|$.

The module of the repulsive velocity vector is afterwards computed for the smallest distance to the end effector[14]. It creates a velocity mapping ruled by a sigmoid function, similar to the one from Figure 5.9 (it can vary depending on the selected parameters), presented in equation 5.11 where the saturation limit correspond to a user set maximum velocity parameter ($|\mathbf{v}_{max,obs}|$). This sigmoid function also leans in the threshold distance set to compute the **ROI** from the previous subsection and a shaping factor (α[15]), both parameterized by the user according to the application.

15: The higher value α takes, the more abrupt slope will present the sigmoid function

$$|\mathbf{v}_{rep,obs}| = \frac{v_{max,obs}}{1 + e^{(|\mathbf{d}_{min}|\frac{2}{\rho}-1)\alpha}} \tag{5.11}$$

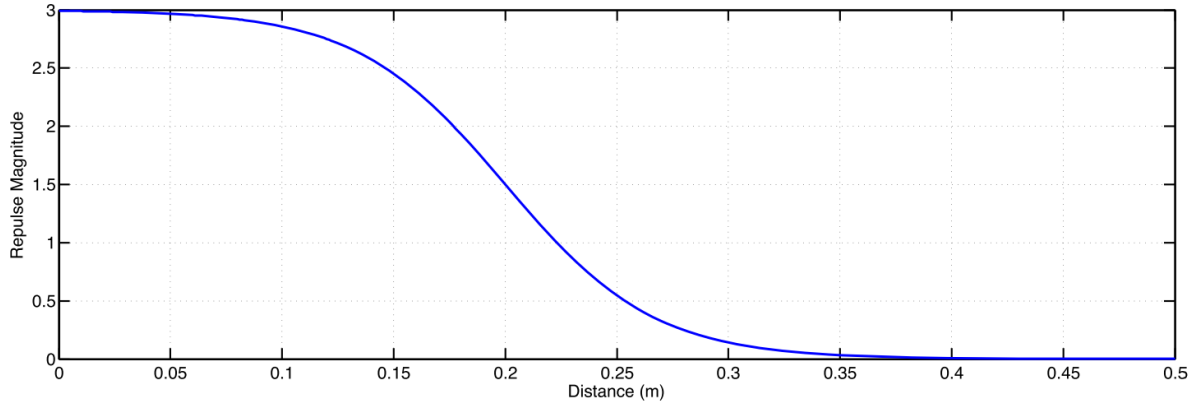

Figure 5.9: Sample profile of the repulsive vector magnitude function. Parameters used: $|\mathbf{v}_{max}| = 3\mathbf{m/s}$, $\rho = 0.4$m and $\alpha = 6$. Figure from [18].

The computed velocity repulsive vector is computed in the end effector frame, however it is desired to know the influence from the world reference frame, the repulsive vector should be rotated employing the corresponding rotation matrix as shown in equation 5.12.

$$\mathbf{v}_{rep,obs} = \mathbf{R}^{r}_{ee}\mathbf{v}_{rep_{ee}} \tag{5.12}$$

Where $\mathbf{v}_{rep,obs}$ correspond to the repulsive velocity due to the obstacle in the task space computed in the robot base reference frame, $\mathbf{R}^{r}_{ee}$ is the rotation matrix from the end efector to the robot base frame, and $\mathbf{v}_{rep_{ee}}$ corresponds to the repulsive velocity of the obstacle in the task space computed in the end effector reference frame.

By implementing the equations above in the control algorithm, the robot will move in a straight line away from the obstacle whenever it is close enough to activate the repulsion policy. In [63], the authors also provide a pivoting repulsive vector to take into account the variations of the direction in the velocity of the obstacle[16]. With all this equations, the control block diagram of DLS-APF Controller can include the following blocks displayed in Figure 5.10.

16: Note that it can be computed as follows:

$$\dot{\mathbf{v}}_{rep,obs} = \frac{\Delta u}{\Delta t}\mathbf{v}_{rep,obs},$$

where $\dot{\mathbf{v}}_{rep,obs}$ is the pivoting vector velocity, $\frac{\Delta u}{\Delta t}$ represents the variation of the direction of the velocity with respect to the time, and $\mathbf{v}_{rep,obs}$ is the computed repulsive velocity due to the obstacle.

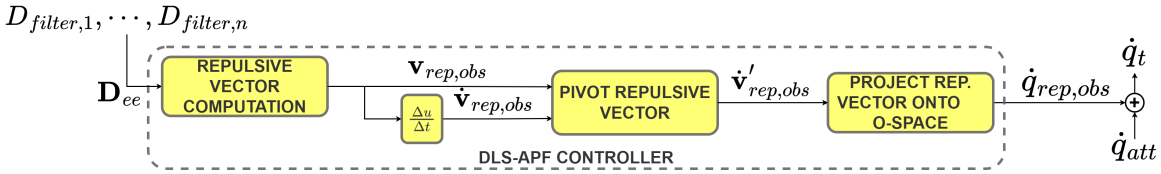

Figure 5.10: DLS-APF Controller: repulsive vector computation blocks.

5.2.5 Damped Least-Square (DLS) Kinematic Model

The Jacobian matrices utilized in Subsections 5.2.3 and 5.2.4 project the computed Cartesian components from the task space to the joint space. To make this conversion possible, the APF based controller uses the IK model. Contrary to the Jacobian right-pseudo inverse as proposed in the reference work [18], this work promotes the usage of the DLS kinematic model

17: Since the State of the Art has shown that it is the most widely used option.

instead[17]. This contribution to the reference controller has been made to simultaneously handle the singular configurations while avoiding obstacles through a APF based controller. More specifically, this work proposes the usage of a weighted Damped Least Square (wDLS) algorithm from [131] instead of a basic DLS one.

18: Mathematical formulation of DLS kinematic models:

$$\dot{\mathbf{q}} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}_n)^{-1}\dot{\mathbf{x}}_e.$$

The selected wDLS kinematic algorithm differs from the traditional DLS[18] in that it encapsulates the resultant matrix that multiplies the Cartesian velocity in a single weighted pseudo inverse Jacobian ($\mathbf{J}^\dagger$). Therefore, the inverse kinematics algorithms is based on a weighted pseudo inverse with damped least-square to calculate the transformation from the task space to the joint space. The computation of the weighted pseudo inverse Jacobian is presented in equation 5.13.

$$\mathbf{J}^\dagger = (\mathbf{M}_q)^{-1}\mathbf{B}(\mathbf{M}_x)^{-1} \tag{5.13}$$

Bearing in mind that $\mathbf{B}$ corresponds to the SVD decomposition perse ($\mathbf{B} = \mathbf{U}_B\mathbf{\Sigma}_B\mathbf{V}'_B$), and that both joint space weighting symmetric matrix ($\mathbf{M}_q$) and task space weighting symmetric matrix ($\mathbf{M}_x$) are symmetric matrices[19], the equation 5.13 can be transformed into the expression displayed in equation 5.14 [131].

19: It means their inverse will coincide with their regular matrix. For further knowledge about the wDLS and other generalized inverse kinematic models check out [132].

$$\mathbf{J}^\dagger = \mathbf{M}_q\mathbf{B}_B\mathbf{\Sigma}_B\mathbf{U}'_B\mathbf{M}_x \tag{5.14}$$

Consequently, the IK model output for the joint velocities can be computed thanks to the Jacobian obtained from the wDLS kinematic model and the current task space velocities (linear and angular) of the end effector as in equation 5.15.

$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \cdot \dot{\mathbf{x}} \tag{5.15}$$

5.2.6 Dynamics Handling: Ruckig

The final layer prior to the ROS-based robot driver is computing a suitable control signal to be followed by the robot. This implies computing a control signal that can be tracked by the low-level controllers of the robot control box. Therefore, the dynamics of the robot must be taken into account. Instead of building the whole dynamic model of the robot, due to be more efficient, the Ruckig Library has been employed instead of the original Reflexxes Type IV Motion Library as proposed in [18].

The joint velocities signals ($\dot{\mathbf{q}}_t$) that arrives to the Bang-Bang Jerk Controller component (the one that integrates the usage of Ruckig library) are the addition of the attractive component of the velocity ($\dot{\mathbf{q}}_{att}$) and the repulsive velocity ($\dot{\mathbf{q}}_{rep,obs}$). On the one hand, the attractive joint velocity is computed by MoveIt and goes though a Proportional-Integral-Derivative (PID) controller to track the precomputed trajectory. On the other hand, as seen in Subsection 5.2.4, the repulsive component is computed utilizing from equation 5.9 to 5.12. These relation between both velocities is expressed in equation 5.16.

$$\dot{\mathbf{q}}_t = \dot{\mathbf{q}}_{att} + \dot{\mathbf{q}}_{rep} \tag{5.16}$$

On top of this, as explained in Subsection 5.2.3, the joint velocity limits downgrade the velocity obtained from equation 5.16. Consequently, the velocity that arrives to the Bang-Bang Jerk Controller will be the one filtered according to the APF to avoid the collision. At this point, the Ruckig library will take the physical joint limits of each robot actuator (the maximum jerk, acceleration, velocity and position) and compute a suitable state for each controller step. This is possible since Ruckig is an Online Trajectory Generator (OTG) library which interpolates with a time optimal joint space trajectories from a given joint state to the desired joint state [133]

5.3 Conclusions to Chapter 5

In Chapter 5 the reference controller design considerations are exposed. The original control loop exposed in Figure 5.1 has been detailed along all the chapter. On the one hand, the advanced scene segmentation filter has been decoupled into the robot filter and the scene features segmentation components. In addition to that, the mathematical formulation of the APF based controller has been exposed, addressing topics as the computation of the distance to the obstacle, the collision risk indexes and repulsive component calculation, and also the wDLS kinematics and the dynamics handled by Ruckig, leading to the DLS-APF controller. With all these considerations and formulations, the final control architecture results as shown in Figure 5.11. However, the presented controller still have the drawback not avoiding singular configurations [117, 131], a solution is proposed in Chapter 6.

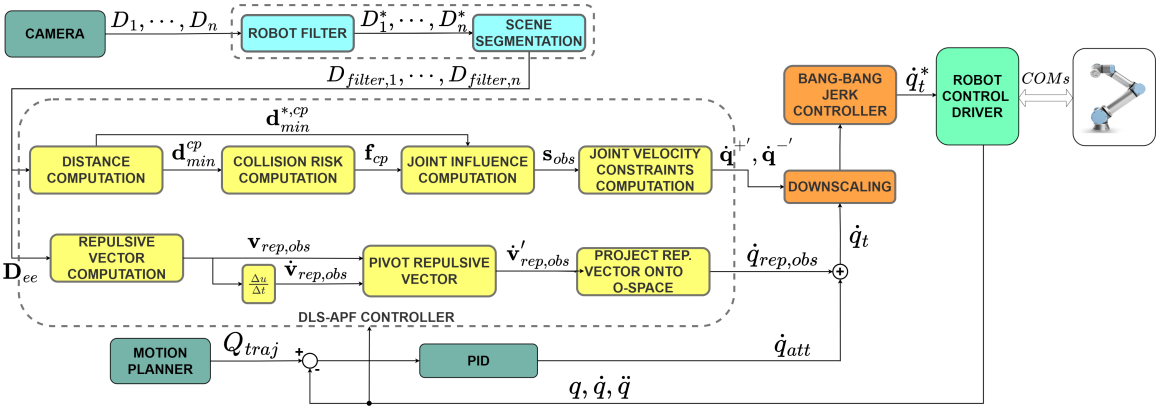

Figure 5.11: Final control loop of the reference controller based on a wDLS control architecture.

Dual Artificial Potential Field (d-APF) Based Controller

6

As exposed in Chapter 2, State of the Art and Objectives, current singularity handling techniques combined with collision avoidance controllers do not guarantee avoiding singular configurations that activates the emergency safety stops of cobots. The reference controller explained in Chapter 5 is no exception of this [117, 131]. In order to face these limitations, a novel kinematic model is proposed in Chapter 4 to simplify the cobots kinematics through the decoupling of the position and orientation. However, the model by itself does not avoid the singular configurations, it does require an specific controller that take advantage of the closed set of solutions for the IK and the singular configurations to avoid simultaneously the collision and the singularities.

For these reasons, this chapter presents the theoretical bases to implement a novel controller that addresses both simultaneously. This novel controller can be considered other main contribution of the current work. The followed strategy consists of combining two APF: the first one for avoiding the obstacle while the remaining APF is for evading the singularities. Thanks to the proposed kinematic model from Chapter 4, the obtainable joint dependent singularity characterization is utilized to compute the distances to the singularity and the virtual potential functions to avoid them. Then, the repulsive component computed with this characterization is added up to the repulsive vector due to the obstacle, allowing the simultaneous avoidance of singularities and collisions. Due to this dual behavior, the proposed controller has been named as the dual Artificial Potential Field (d-APF) controller.

Consequently, Chapter 6 addresses how to obtain the aforementioned joint dependent characterization of the distance to the singularity and how to compute the repulsive component due to the singularity (based on the prior joint dependent characterization). On top of this, Chapter 6 also explains the main contribution to the robotics control algorithms of this work which based on the combination of two different APF, one for avoiding the collisions with obstacles and one for repelling the manipulator from the singular (or blocking) configurations. This novel control architecture, denoted as dual Artificial Potential Field (d-APF) controller, is obtained through the required modifications to the reference DLS-APF controller from Chapter 5 by substituting the wDLS kinematic model for the proposed in Chapter 4. As in the previous chapter, the implementation aspects on specific hardware have been reserved for Chapter 7.

6.1 Design Basics

The Design Basics section presents an overview on the controller requirements and the control architecture. Since the proposed novel approach lies in the reference controller, the aspects explained below are restricted to the contribution of the d-APF controller. In other words, there will be

explained the contribution of adding up two different potential fields to avoid singularities and obstacles simultaneously to keep shop floor safety without disregarding the manufacturing efficiency.

6.1.1 Requirements

Regarding the requirements for the d-APF controller, they are, indeed, the same as the ones presented in Chapter 5 to allow the proper comparison between the reference controller (DLS-APF controller) and the proposed one (d-APF controller). As a reminder, the controller requirements can be found at Subsection 5.1.1.

Achieving these requirements will aid enabling an efficient and safer interaction between human and robots on industrial collaborative scenarios. The following section proposed the modified control architecture to fulfill the aim of this work: achieving safe and efficient robotic operation for industrial collaborative scenarios. In other words, to keep a robot producing alongside human operators without harming them nor reducing the manufacturing performance on protection-free spaces.

6.1.2 Control Architecture

The proposed control architecture for the d-APF controller in big block of functionalities can be appreciated in Figure 6.1. It might seem that the purple control block is the only difference according to the reference controller; however, the utilization of a singularity avoidance APF implies slight variations on some of the remaining control blocks. Therefore, this section highlights briefly the required modification with respect to the DLS-APF reference controllers in order to help a better understanding of the main contribution of this work.

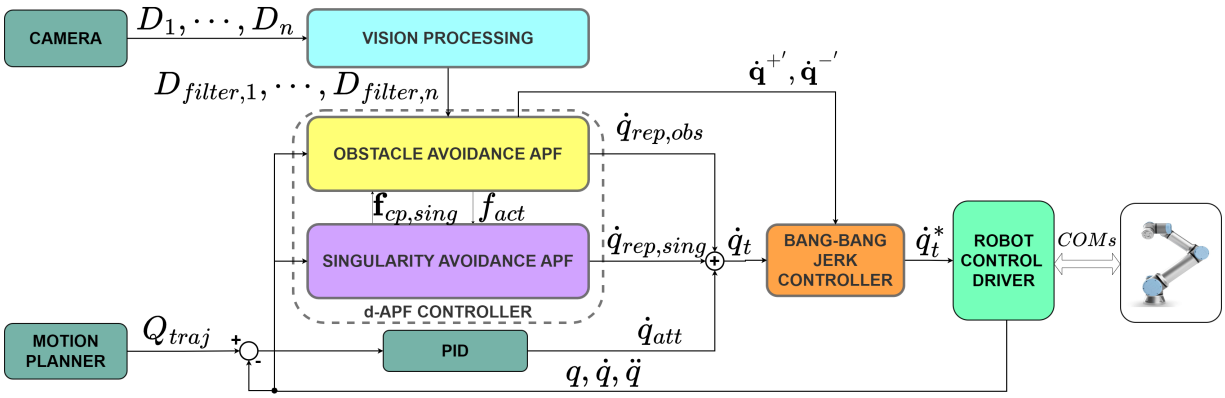

Figure 6.1: Overview of the control diagram for the proposed d-APF controller for collision avoidance and singularity handling.

From among the coloured control blocks of Figure 6.1, the ones that received some modification correspond to the yellow and purple control boxes (relative to the obstacle and singularity avoidance, respectively). On top of that, the orange control block (the one referred to the Bang-Bang Jerk Controller to handle inherently the dynamics of the robot, does not received a modification as if, but the signals that arrives to that control block are modified. Instead of receiving just a single repulsive component due to the obstacles ($\dot{\mathbf{q}}_{rep,obs}$) and the limiting joint velocities vectors

($\dot{\mathbf{q}}^{+'}, \dot{\mathbf{q}}^{-'}$), it also receives the repulsive component to avoid the singular configurations ($\dot{\mathbf{q}}_{rep,sing}$) to be added to the attractive component of the APF ($\dot{\mathbf{q}}_{att}$). The differences with the previous Bang-Bang Jerk controller are addressed in Subsection 6.2.2.

Consequently the purple control box has been proposed as a contribution to the original control loop to integrate the simultaneous collision an singularity avoidance. This component, so called Singularity Avoidance APF, aims to generate a suitable singularity avoidance control response that can be added to the response due to the obstacle avoidance block. To synchronize the obstacle avoidance component with the singularity avoidance one, two signals are employed: the activation factor ($\mathbf{f}_{act}$) that will activate the singularity repulsion only whenever the robot is avoiding and obstacle, and the singularity proximity risk index ($\mathbf{f}_{cp,sing}$) which measures how close is each control point or joint to the singular configuration. The behavior of the purple control box and the improvements required in the obstacle avoidance APF are addressed along Section 6.3.

The integration of the Singularity Avoidance APF control block to the control loop brings some modifications to the regular control behavior exposed in the previous chapter. First of all, it is remarkable that the kinematic model to compute the responses is no longer the DLS kinematic model, instead, the proposed decoupled kinematic model for non-spherical wrist cobots from Chapter 4 is employed. This change simplifies the complexity of the employed kinematic model reducing the required computational cost and allowing the computation of a closed set of singular configurations. These changes in the behavior of the controller due to utilizing another kinematic model are explained in Subsection 6.2.1.

Through all the contributions proposed to the reference controller, the new control loop helps in the computation of the controller velocity limits to downscale the control velocity setpoing while avoiding singularities during the collision avoidance in a more efficient and safer way.

6.2 APF Based Obstacle Avoidance Component

From the control architecture presented in Chapter 5, the scene segmentation algorithm and the measure of the distances to the obstacle, as well as the computation of the collision risk indexes and the repulsive vector have been kept. On the contrary, the employed kinematic model for the collision risk indexes and the repulsive vector, and the preparation of the data before entering the dynamics manager (Rucking Bang-Bang Jerk Controller control block) have been slightly adapted to adapt its behavior to a simultaneous singularity and collision avoidance without saturating the joint velocity control set point. Therefore, the implications in the behavior of the control loop due to such changes are explained in the following subsections.

6.2.1 Decoupled Kinematic Model

The main modification referred to the kinematic model employed is that it employs the decoupled kinematic model for non-spherical wrist robots intead of the DLS one. Since the decoupled kinematic model for non-spherical wrist cobot is already in-depth explained in Chapter 4, no additional notes will be given relative to the kinematic behavior of the model[1]. However, this change in the kinematic model affects the computation of both the collision risk indexes and the repulsive component of the obstacle avoidance related control blocks, their mathematical formulation remains unchangeable though[2]. Thus, the main contribution of using the proposed decoupled kinematic model for cobots relies on the simplification of the Jacobian matrix which enables the computation of a closed set of solutions for both IK and singular configurations. These solutions provide, on the one hand, more efficient computations during trajectory planning operations, while they also avoid being trapped into singular configurations during the collision avoidance phases. Thus, the efficiency of the manufacturing processes is cared while the safety for the operator is maintained.

Beginning with the risk collision index, the equation that is affected by the change of the kinematic model is equation 5.7[3]. This mathematical expression relies in the Jacobian matrix for each control point ($\mathbf{J}_{cp_i}$) to calculate the influence of each joint collision risk indexes. Therefore, the new Jacobian matrices for each control point will be calculated with respect to the non-spherical wrist decoupled kinematic model, instead. Subsequently, this influence is utilized to compute the velocity limits required to avoid crashing with the obstacle (see equation 5.8[4]).

As the velocity limits computation is exclusively referred to the collision avoidance, their calculation will remain the same. However, the risk index employed to compute the velocity limits will be modified to conform to the singularity avoidance additional requirements. In this manner, the global risk index will be a combination of each collision risk index: the one due to the collision (check equation 5.6) and the collision risk index due to singularity (explained section Subsection 6.3.2). The mathematical relationship to combine both risk indexes can be appreciated in equation 6.1.

$$f_{cp_i} = f_{cp_i,obs} + f_{cp_i,sing}, \text{ where if } f_{cp_i} > 1 \Rightarrow f_{cp_i} = 1 \tag{6.1}$$

On top of this, given that equation 5.9[5] projects the repulsive vector due to the obstacle avoidance into the joint space, the computation of the obstacle avoidance repulsive vector is also affected by the change of kinematic model. In this particular case, the equation is transform as shown in equation 6.2.

$$\dot{\mathbf{q}}_{rep,obs} = \mathbf{J}^{-1}(\mathbf{q})\mathbf{t} \quad \text{where:} \quad \mathbf{t} = \begin{bmatrix} \mathbf{v}_{rep,obs} \\ 0 \end{bmatrix} \tag{6.2}$$

From the equation below, and comparing its kinematic behavior with the expression from equation 5.9 (the equivalent one from the reference controller), the reader should note that for the new obstacle avoidance repulsive vector, there is no treatment of the singularity yet. Up to this

1: Note that if the robot already has an spherical wrist such as, for example, in the case of the LBR IIWA cobot, the required modification is even more straightforward because it can be applied the classic decoupled kinematic model for spherical wrist robots instead.

2: Note that the proposed changes are not a modification *per se*. The equations both parameters employs are not modified as if, the only required change is based on the usage of another Jacobian model. Therefore, both parameters are affected by the change of the kinematic model, but their mathematical formulation remains constant.

3: Equation 5.7 reminder:

$$\mathbf{s}_{o,s} = \mathbf{J}_{cp_i}\mathbf{d}^{*,cp_i}_{min} f_{cp_i}.$$

4: Equation 5.8 reminder:

$$\text{if} \quad s_{i_{o,s}} \geq 0, \quad \dot{q}_i^{+'} = \dot{q}_i^{+}(1 - f_{cp_i}),$$
$$\text{else}, \quad \dot{q}_i^{-'} = -\dot{q}_i^{+}(1 - f_{cp_i}).$$

5: Equation 5.9 reminder:

$$\dot{\mathbf{q}}_{rep,obs} = \mathbf{J}(\mathbf{q})^{\dagger}\mathbf{t}, \text{ where: } \mathbf{t} = \begin{bmatrix} \mathbf{v}_{rep,obs} \\ 0 \end{bmatrix}.$$

point, it is relevant to highlight that the advantage of the proposed controller is to evade the singularity, not to handle it to change the robot behavior in the nearby configurations. Therefore, to avoid falling into a singular region where the robot misbehave kinematically, the d-APF controller proposes a repulsive component that pushes the robot away from those configurations. Therefore, no singularity handling technique is required in the Jacobian matrix as the robot does not goes around the vicinities of a singular configuration.

The calculation of both, the singularity risk index and the singularity repulsive force, are further addressed in the following section (Section 6.3) where all the components to repell the robot from the singularities are explained in detail.

6.2.2 Dynamics Handling: Ruckig

With respect to the dynamics handled by the Ruckig the Online Trajectory Generator (OTG) minimal changes have been made. These modifications can be summed up into two main amendments. The aim of these changes is to prepare the robot dynamic handler to limit properly the robot joints velocities due to the combined action of the repulsive responses.

The first variation is related to the downscale control block utilized to limit the velocities to avoid the collision in the reference controller. Since in the d-APF controller the risk index is a combination of the singularity proximity risk index and the collision risk index (see equation 6.1), the downscaling obtained prior to the bang-bang jerk controller control block is referred to both. It means that the downscaling can limit the movement of any joint depending on how close is to a collateral collision and to a singular configuration simultaneously. This behavior is expected to increase the safety and efficiency of the application as no controller box safety stop is required to handle it dynamically.

The other modification is referred to the control signal that arrives to the downscaling block[6]. In the d-APF controller, two repulsive forces are computed that must be added up to the attrative potential reference. Therefore, the expression to compute the resultant component to be handled by the downscale control block before downscaling the velocities is shown below in equation 6.3.

$$\dot{\mathbf{q}}_t = \dot{\mathbf{q}}_{att} + \dot{\mathbf{q}}_{rep} = \dot{\mathbf{q}}_{att} + \mathbf{J}^{-1} \cdot \left(\begin{bmatrix} \mathbf{v}_{rep,obs} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{v}_{rep,sing} \\ \mathbf{0} \end{bmatrix} \right) \tag{6.3}$$

How to implement and configure the Ruckig library is further addressed in the following chapter (Chapter 7) where all the implementation details are displayed.

6: Quick reminder, in the previous chapter it arrives the expression from equation 5.16:

$$\dot{\mathbf{q}}_{sum} = \dot{\mathbf{q}}_{att} + \dot{\mathbf{q}}_{rep}.$$

6.3 APF Based Singularity Avoidance Component

In the previous section, it has been hinted that the potential of the proposed approach (the d-APF controller) relies in the capabilities of pushing

the robot away from the singularity instead of applying any singularity handling technique. This behavior brings two main advantages: on the one hand, no additional singularity handling algorithms is required for the kinematics of the robot and, on the other hand, since the decoupled kinematic model presents a simpler version of the Jacobian matrix, the computations of the responses will be more efficient. These two advantages have been tested in Chapter 8. Nevertheless, in the current section the mathematical formulation to implement the singularity avoidance component is detailed.

The proposed controller can be split into two big functionality blocks: the singularity proximity risk indexes and the singularity repulsive component. This division is similar to the one found for the obstacle avoidance component[7]. Therefore, the singularity proximity risk will be in charge of limiting the joint velocities to avoid entering in a singular configuration while avoiding the obstacle. Moreover, the singularity repulsive component handles the evasion of the singular configuration pushing the robot away from those positions.

7: Where there is a collision risk proximity index to limit the joint velocities, evading non-desired collisions with the body of the manipulator, and a repulsive force due to the closest obstacle to the end effector.

In order to put these two functionalities to work, the distance to the singularity has to be measured. This can be a controversial point because there is no standardized way to measure how close the robot is to the singular region[8]. This work establishes a method to measure the distance to the singularity based on the joint dependent singular configurations characterization obtained from the robot kinematic decoupling (for further knowledge check Subsection 6.3.1).

8: Note that there is not a specific distance to define a singular region even. In this thesis the criterion exposed in [105] has been chosen. However, the authors generally refers as a singular region to those configurations where the manipulability ellipsoid collapses and the robot misbehaved, but they do not give any particular specification.

Once the distance to the singularity has been evaluated, the singularity proximity risk index computation and the singularity repulsive vector control blocks are fed with this data. In the following subsections, each of the aforementioned aspects will be specifically addressed from the singularity distance computation, to the singularity repulsive vector calculation.

6.3.1 Singularity Distance Computation

As stated, the measurement of how close or far the robot is to a singular configurations is not univocally defined. This issue can be solved by different measures such as the manipulability computation[9]. However, this method brings the drawback of continuously computing the determinant of the Jacobian, which in a case of a 6 or 7 DoFs robot introduces a high computational load to the control algorithm, as shown in the literature (Section 2.1 State of the Art). Since with the proposed kinematic model, from Chapter 4, a close set of joint dependent solutions for the inverse kinematics can be obtained, the d-APF controller proposes a more efficient measure of the distance to the singularity based on this characterization.

9: As seen in the State of the Art, it is considered one of the most utilized.

Ideally, from the joint dependent characterization of the singular configurations a set of n[10] equations should describe the conditions to check whether the robot is in a singularity or not. In addition, the joint dependant singularity expressions correspond to a combination of trigonometric functions described by the coupling of different joints. Bearing this two facts in mind, this Ph.D. proposes the usage of a distance to the singularity vector ($\mathbf{d}_{sing} \in \mathbb{R}^n$) defined by the gradient or the

10: In this case, n is the DoFs of the robot, so it can be 6 or 7.

slope of the normalized singularity characterized functions (see equation 6.4).

$$\mathbf{d}_{sing} = \begin{bmatrix} f(|\mathbf{J}_{11}|)_{q_1} & \cdots & f(|\mathbf{J}_{22}|)_{q_{n-2}} & f(|\mathbf{J}_{22}|)_{q_{n-1}} & f(|\mathbf{J}_{22}|)_{q_n} \end{bmatrix}^T \quad (6.4)$$

The key for the success of the proposed singularity distance measurement lies in a suitable definition of the normalized singularity characterized functions ($f(|\mathbf{J}_{ii}|)_{q_j}$, Where ii is referred to the respective Jacobian block matrix, and j is related to the joint number from which is computed the singularity function; in other words, the joint that is characterized for the resultant expression to nullify the $|\mathbf{J}|$). By normalized is meant that each function has its image limited in the $[0, 1]$ interval. To do so, each of the singularity characterized functions will be divided by its maximum possible value[11]. Moreover, the singularity characterized functions correspond to the parametric expression where each solution of equations 4.16 (for 6 DoFs) and 4.17 (for 7 DoFs)[12] can be found. In this manner, each of Jacobian block matrix can be studied independently, obtaining a function for each one. Since the proposed model allows the kinematic decoupling, the characterization for the first $n-3$ joints corresponds to the $\mathbf{J}_{11}$ Jacobian block matrix (see equation 6.5a), and the remaining three to the $\mathbf{J}_{22}$ Jacobian block matrix (check equation 6.5b).

11: In order to compute this value, a typical advanced calculus study involving the second order derivative is required. This study can be executed offline to lighten the computational load of the singularity distance computation block.

12: In general, both equations can be summarized as follows:

$$|\mathbf{J}| = |\mathbf{J}_{11}| \cdot |\mathbf{J}_{22}| = 0.$$

$$|\mathbf{J}_{11}| = f(q_1, \cdots, q_{n-3}) = 0 \quad (6.5a)$$

$$|\mathbf{J}_{22}| = f(q_{n-2}, q_{n-1}, q_n) = 0 \quad (6.5b)$$

Subsequently, the joint dependant characterization is computed extracting the common terms for each joint of the expressions as displayed below in equation 6.6.

$$|\mathbf{J}_{11}| = f(q_1, \cdots, q_{n-3}) = f_1(q_1) \cdot f_2(q_2) \cdots f_{n-3}(q_{n-3}) \quad (6.6a)$$

$$|\mathbf{J}_{22}| = f(q_{n-2}, q_{n-1}, q_n) = f_{n-2}(q_{n-2}) \cdot f_{n-1}(q_{n-1}) \cdot f_n(q_n) \quad (6.6b)$$

By analyzing the gradient of each characterize function ($\nabla f_i(q_i)$) the variation rate for each expression according to the desired joint can be studied[13]. Furthermore, with the second order derivative with respect to the desired joint of $\nabla f_i(q_i)$, the position of the maximum value can be obtained. This last step enables the computation of equation 6.7 where the general formulation of the normalized joint dependant characterization is presented.

13: This variation rate is relevant because sin and cos trigonometric functions present a slow variation when they are close to 1. Indeed, the closer the $\nabla f_i(q_i)$ function is closer to 1, the grater slope is in the non-derived function ($f_i(q_i)$). In other words, whenever the derivative (sin or cos) is equal to 1, the original function has maximum gradient or slove, and will be a candidate to be singular.

$$f(|\mathbf{J}_{ii}|)_{q_i} = \frac{\nabla f_i(q_i)}{\max(\nabla f_i(q_i))} \quad (6.7)$$

As presented in equation 6.7, the general formulation for each normalized singularity distance function strongly depends on the particular joint dependent singular characterization for each robot structure. Therefore, further details on the implementation can be found at Chapter 7 where two examples of application are presented for each laboratory robot.

6.3.2 Singularity Proximity Risk Index Computation

The goal of the singularity proximity risk is to aid the collision risk indexes to limit and downscale the combined response of the attractive and the repulsive potentials. For such aim, the singularity avoidance APF is connected to control blocks from the obstacle avoidance APF. More specifically, as displayed in Figure 6.2, the signals that connects both APFs correspond to the activation factor for each joint ($\mathbf{f}_{act}$) and the singularity proximity risk indexes ($\mathbf{f}_{cp,sing}$). This influence between the singularity and obstacle avoidance APFs modifies the global risk index ($\mathbf{f}_{cp}$ from equation 6.1) that feeds the previously explained (in Chapter 5) Joint Influence Computation and Joint Velocity Constraints Computation control boxes to compute the upper and lower joint velocities limits[14].

14: The mathematical expressions that defines the behavior of both control

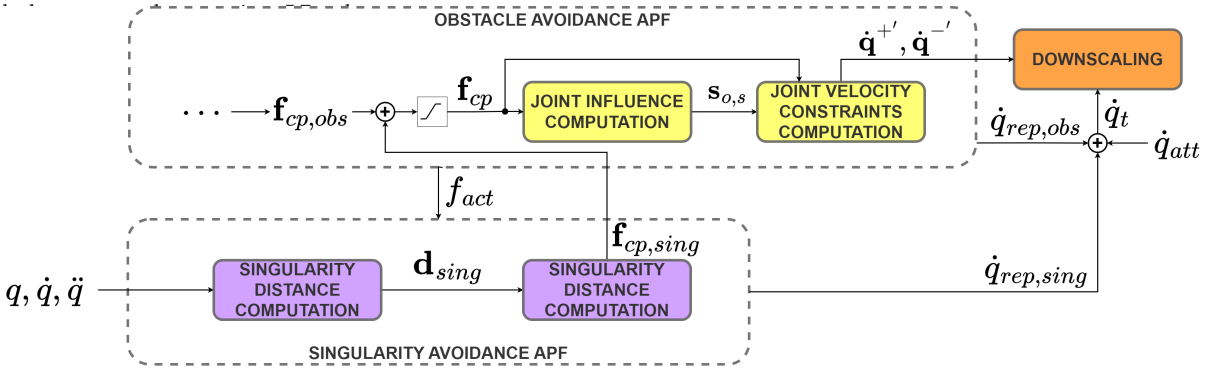

Figure 6.2: Singularity proximity risk indexes control loop of the d-APF controller.

To calculate the singularity proximity risk indexes, the information referred to the distance to the singularity ($\mathbf{d}_{sing}$) has to be received first. Then, a sigmoid function according to the distance to the singularity for each control point ($d_{sing,i}$) is employed to compute each singularity proximity risk index ($f_{cp_i,sing}$). Similar to the case of the collision risk index, this sigmoid function is also bounded between 0 and 1 as described in equation 6.8. Since the form factor of equation 6.8 is negative, the slope of the sigmoid function will be negative. It will present a similar form to one from the example in Figure 6.3. In equation 6.8, there are also two adaptable parameters that should fit the interest of the application and the safety, being configurable by the user. They correspond to the lower threshold (l_t) and upper threshold (u_t). These two limits are the ones in charge of defining the distance to start applying the limitation due to the singularity and the value where the actuator should be saturated (its movement is completely blocked).

$$f_{cp_i,sing} = \begin{cases} 0 & \text{, if } d_{sing,i} < l_t \\ \frac{1}{1+e^{-\alpha_s(d_{sing,i}-\rho_{0,s})}} & \text{, if } l_t < d_{sing,i} < u_t \\ 1 & \text{, if } d_{sing,i} > u_t \end{cases} \tag{6.8}$$

Once the singularity risk index is computed for each control point, it is sent to the joint influence computation control box (see Figure 6.3) where it is added up to the collision risk index as displayed in equation 6.1[15]. This change in the global risk index that arrives to the joint influence and the joint velocity constraints control blocks, modifies the computation of equations 5.7 and 5.8 however their mathematical formulation remains

15: Remember that equation 6.1 is defined as:

$$\mathbf{f}_{cp} = \mathbf{f}_{cp,obs} + \mathbf{f}_{cp,sing} \in [0,1]^n.$$

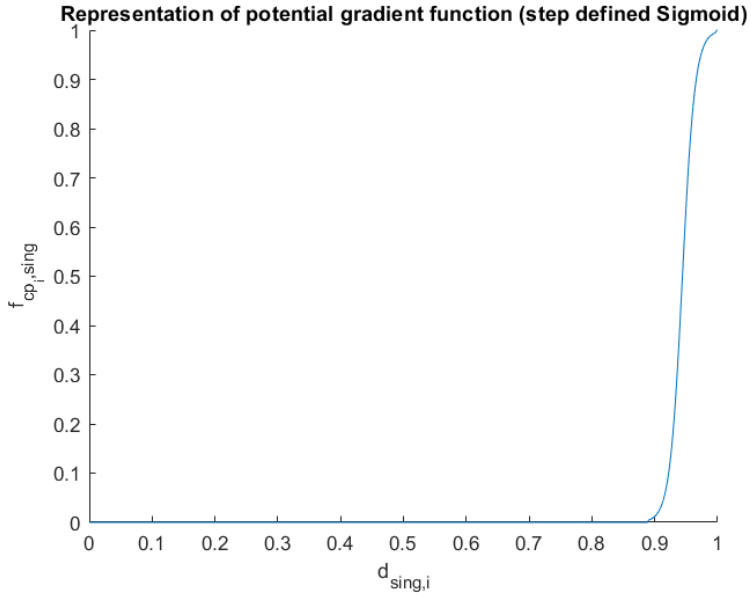

Figure 6.3: Sample profile of the singularity risk index function. Parameters used: $\rho_{0,s} = 0.9440$, $\alpha = -100$, $l_t = 0.890$, and $u_t = 0.998$

constant. Substituting the equation 6.1 into these equations, the expanded general formulation for computing the joint influence and constraints is presented in equation 6.9 for each control point.

$$\mathbf{s}_{o,s} = \mathbf{J}_{cp_i}\mathbf{d}_{min}^{*,cp_i} f_{cp_i} = \mathbf{J}_{cp_i}\mathbf{d}_{min}^{*,cp_i}(f_{cp_i,obs} + f_{cp_i,sing}) \tag{6.9a}$$

$$\begin{aligned} \text{if} \quad s_{i_{o,s}} \geq 0, \quad & \dot{q}_i^{+'} = \dot{q}_i^{+}(1 - f_{cp_i}) = \dot{q}_i^{+}(1 - (f_{cp_i,obs} + f_{cp_i,sing})), \\ \text{else}, \quad & \dot{q}_i^{-'} = -\dot{q}_i^{+}(1 - f_{cp_i}) = -\dot{q}_i^{+}(1 - (f_{cp_i,obs} + f_{cp_i,sing})) \end{aligned} \tag{6.9b}$$

where $\mathbf{s}_{o,s}$ corresponds to the computed joint influence due to the obstacle and singularity, $\mathbf{J}_{cp_i}$ is the geometric Jacobian relative to the i-th control point, $\mathbf{d}_{min}^{*,cp}$ is the minimum distance unitary vector to each control point, $(f_{cp_i}, f_{cp_i,obs}, f_{cp_i,sing})$ correspond to the global, obstacle and singularity proximity risk index for the i-th control point, respectively, and $(\dot{q}_i^{-'}, \dot{q}_i^{+'})$ are the computed upper and lower limit for the i-th joint joint velocity.

6.3.3 Singularity Repulsive Component Computation

The singularity repulsive component is in charge of computing a dynamic response to push the robot away from the singular configurations while avoiding the obstacle. It is remarkable that it has to be active only while avoiding the obstacles because during the offline trajectory planning stages of the task, the singularity avoidance is already applied. To achieve this goal, the previous components from the obstacle avoidance APF are combined with the new control blocks from the singularity avoidance APF as shown in Figure 6.4.

Beginning with the calculation of the singularity avoidance APF control blocks components, the first one of the displayed loop in Figure 6.4 corresponds to the singularity distance computation control block (check

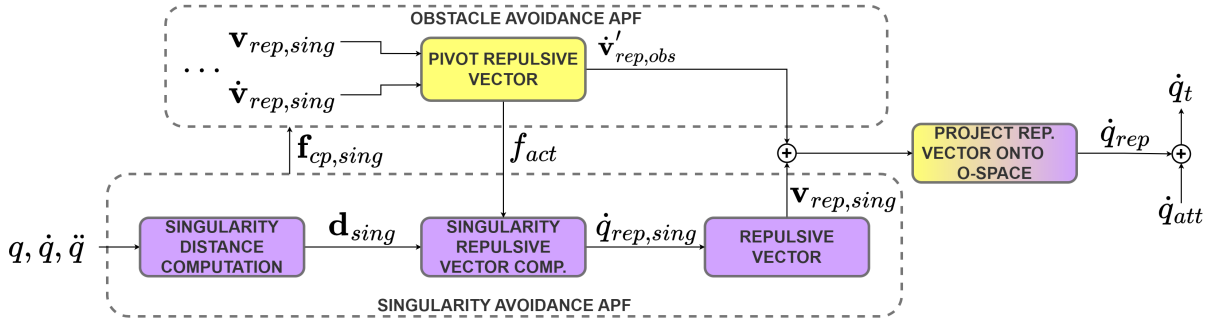

Figure 6.4: Singularity repulsive component of the control loop of the d-APF controller.

16: The computation of the distance to the singularity is addressed in equation 6.4:

$$\mathbf{d}_{sing} = \begin{bmatrix} f(|\mathbf{J}_{11}|)_{q_1} & \cdots & f(|\mathbf{J}_{22}|)_{q_n} \end{bmatrix}^T.$$

17: The mathematical relationship between the angular and the linear velocity is defined by:

$$\boldsymbol{\omega} \times \mathbf{r} = \mathbf{v}$$

Or just the relation between their modules if it is a planar problem:

$$\omega \cdot r = v$$

$\mathbf{d}_{sing}$ from equation 6.4 for further references[16]). With this input ($\mathbf{d}_{sing}$) and an activation factor ($\mathbf{f}_{act}$), the repulsive vector can be computed in the task space ($\mathbf{v}_{rep,sing}$). For this singularity repulsive Cartesian component, the virtual repulsive velocity is contrary to the instant joint velocity of the joint that is converging to the singular configuration and it is applied to the tip of the corresponding link as shown in Figure 6.5. Then, the virtual Cartesian velocity can be transformed into its joint influence ($\dot{\mathbf{q}}_{rep,sing}$) by applying classical mechanics relationships between lineal and angular velocity on a solid rigid[17]. This relationship is mathematically described for each joint in equation 6.10 where $-gn(q_i)$ is the sign function of joint velocity, f_{act} corresponds to the activation factor later explained in this section, $v_{rep,sing,max}$ is the module of the maximum allowed velocity due to the singularity repulsion, and l_i is the length of the link.

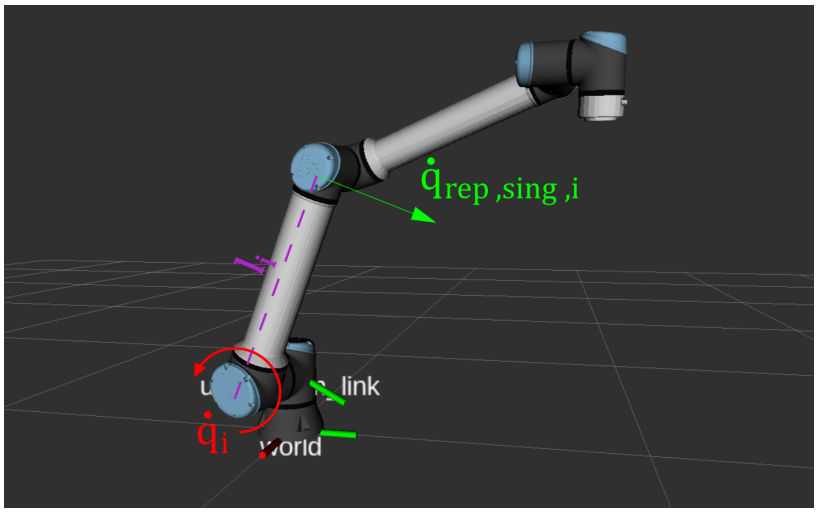

Figure 6.5: Visualization in RViz (ROS visualization) of a UR10e in a random configuration where the singularity repulsive vector is computed.

$$\dot{q}_{rep,sing,i} = -sgn(q_i) \cdot f_{act} \cdot v_{rep,sing,max} \cdot f_{cp_i,sing}(d_{sing,i}) \cdot \frac{1}{l_i} \quad (6.10)$$

Note that each joint will create only a repulsive signal that will be applied the same in all the control points that belongs to the link where the repulsive component is applied. Subsequently, the computed vector of joint repulsive forces is transformed into the task space as shown in equation 6.11. This step is required to the latter computation of the global repulsive velocity that should be projected back into the joint space ($\dot{\mathbf{q}}_{rep}$).

This process is executed in the repulsive vector control block from the singularity avoidance APF.

$$\dot{\mathbf{x}} = \mathbf{J} \cdot \dot{\mathbf{q}} = \begin{bmatrix} \mathbf{v}_{rep,sing} & \boldsymbol{\omega}_{rep,sing} \end{bmatrix}^T \tag{6.11}$$

Since it is desired to maintain the orientation of the TCP while avoiding the collision and the singularity, from the repulsive twist computed in equation 6.10 it will be only employed the first three components corresponding to the linear velocity of the manipulator ($\mathbf{v}_{rep,sing}$). This criteria guarantees the controller to avoid loosing the load due to a change in the orientation of the manipulator gripper.

In addition to the specific components for the singularity avoidance APF, two modifications have been applied into the obstacle avoidance APF control blocks from Figure 6.4. Firstly, the pivot repulsive vector component feeds the signularity repulsive vector with an activation factor (f_{act}). This parameter is employed to check whether the robot is avoiding a collision due to a dynamic obstacle or not. It can be computed as follows in equation 6.12. In this way, as shown in equation 6.10, the robot will avoid singularity only when the activation factor is different than 0.

$$f_{act} = \begin{cases} 0 & \text{, if } |\dot{\mathbf{v}}'_{rep,obs}| = 0 \\ 1 & \text{, otherwise} \end{cases} \tag{6.12}$$

Lastly, the projector of the repulsive vector onto the task space has also been updated. In this occasion, the mathematical formulation to project the Cartesian velocity of the robot into the joint space is described in equation 6.2[18]. However, as the input of the control block is the addition of the repulsive velocity due to the singularity and to the obstacle, the prior mathematical expression is substituted for the one appreciated in equation 6.13 instead.

$$\dot{\mathbf{q}}_{rep} = \mathbf{J}^{-1}(\mathbf{q})\mathbf{t} \quad \text{where:} \quad \mathbf{t} = \begin{bmatrix} \mathbf{v}_{rep,obs} + \mathbf{v}_{rep,sing} \\ 0 \end{bmatrix} \tag{6.13}$$

18: Remember that equation 6.2 formules the following relationship:

$\dot{\mathbf{q}}_{rep,obs} = \mathbf{J}^{-1}(\mathbf{q})\mathbf{t}$ where: $\mathbf{t} = \begin{bmatrix} \mathbf{v}_{rep,obs} \\ 0 \end{bmatrix}$.

6.4 Conclusions to Chapter 6

In this chapter the mathematical formulation to implement the d-APF controller are exposed. The equations present throughout the chapter can be summarized in the control loop of Figure 6.6. In addition, the modifications required to transform the reference controller into the proposed one are also presented. In this manner, some improvements are theoretically observed such as a reduction of the required computational load to compute the repulsive responses, or the fact that the singular configurations are limited so these positions can be repelled more efficiently. These benefits come from the alternative method presented due to the change in the kinematic model of the original controller. Moreover, the control algorithm presented keeps the orientation of the TCP to assure that no load is lost due to a change in the orientation while avoiding a collision. This behavior is also a more safer one because when carrying large objects, the object could hit the operator if the orientation of the

end effector varies. The advantages of the d-APF controller are tested against the DLS-APF in Chapter 8.

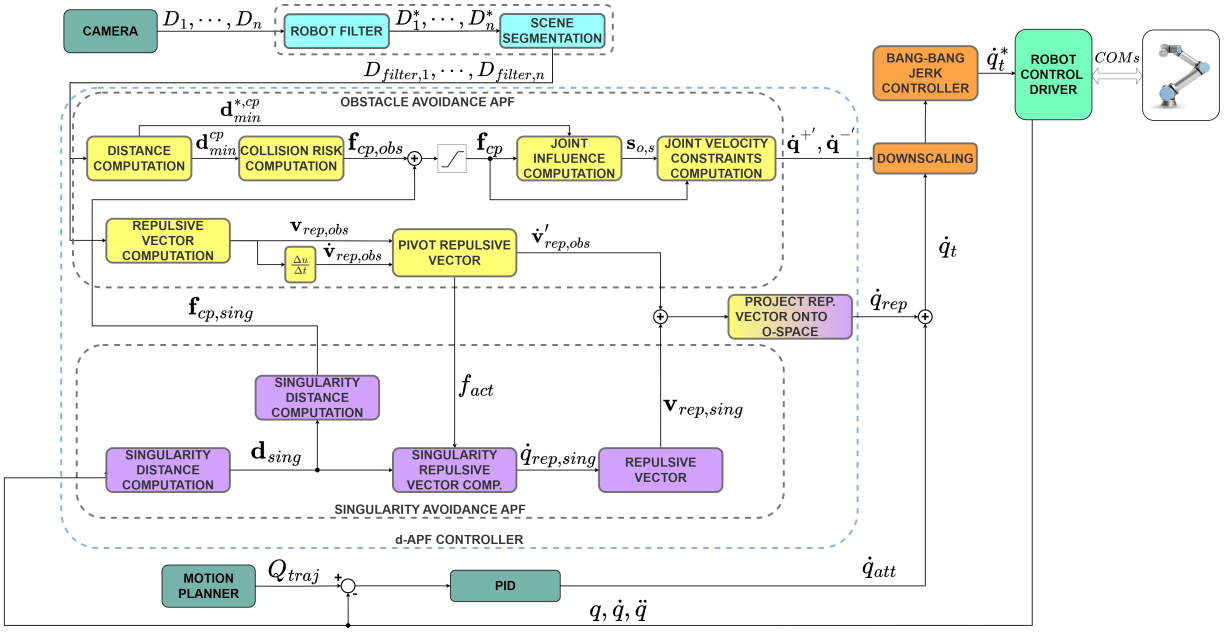

Figure 6.6: Overview of the detailed control diagram for the proposed d-APF controller for collision avoidance and singularity handling.

7 Controllers Implementation

In the previous chapters (Chapters 4, 5, and 6) the theoretical aspects about the novel decoupled kinematic model for non-spherical wrist cobots, the DLS-APF controller and the d-APF controller are covered. To be able to test the proposed controller performance, this chapter addresses how to go from the mathematical fundamentals of previous chapters to the practical implementation into real hardware. In this way, by following the considerations displayed in this chapter to implement both controllers on the UR10e and the SUPSI robot, the proposed control architecture could be utilized on industrial plants.

In order to implement a suitable system to be applied on industrial collaborative applications several aspects have to be regarded. These issues are mainly related to the real time management of the controller and the safety assurance. To address both aspects simultaneously it is proposed a combined software architecture based on ROS Control combined with Orocos (Open Robot control Software). In this manner, the ROS based controller implements the advanced control capabilities of the Hardware Interface [134], while the Orocos middleware manages transparently the real time capabilities of the OS where the application is implemented [135]. With this software architecture, the robot engineer can focus on developing the high-level skills of the controller instead of managing the RT aspects.

In the following sections the software control architecture to encapsulate ROS Control capabilities inside Orocos components is exposed. In addition, the particular Hardware Interface employed for each of the test robots is also detailed in this chapter. Since the d-APF controller requires the decoupled kinematic model from Chapter 4, the particularization of each kinematic model as well as its specific singularity study are detailed for each robot structure. This will lead to a set of closed solutions for the singular configurations of the arm and its IK. Lastly, the data structure and communications required to put up to work the controllers inside the ROS network are also described.

7.1 Controllers Software Architecture

In this section common aspects of the ROS based advanced controller are detailed. In particular, a suitable real time capability to encapsulate ROS control capabilities and the ROS network data structure are presented in detail. The first of the topics addresses the implementation of a generic software architecture to integrate different developed ROS based controller into Orocos to handle transparently the real time capabilities. The remaining subsection present the data structure used for sending information between the different ROS nodes of the network.

7.1.1 Real time Management: Hardware Interface/Orocos encapsulation

As stated along the preamble of the chapter and the section, in order to embed the ROS Control capabilities into software units that works on real time it is proposed the utilization of Orocos library. Orocos is a portable C++ library for advanced machine and robot control which encapsulates the real time capabilities of an operating systems into components. In this way, thanks to this encapsulation the user can focus in programming an advanced controller that fits the application instead of taking care of the real time management of the control cycle. For such a purpose, the software architecture presented in Figure 7.1 is employed.

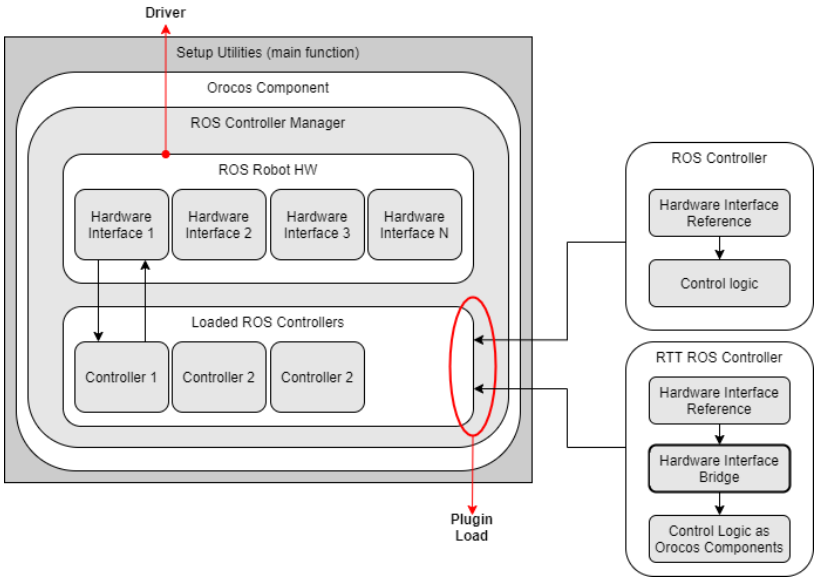

Figure 7.1: General diagram of the tools for implementing complex real time control loops in ROS Control by integrating it into Orocos. The real time Control manager is the central element where the ROS controller manager is wrapped into an Orocos component. On the right side of the diagram the plugin based controller loading process is represented together with the internal structure of regular controller as opposed to real time controllers using the RTT ROS Control Bridge.

The aim of this software architecture is to complement the real time ready capabilities of ROS Control integrating the ROS Controller Manager into a real time Orocos component. The ROS Controller Manager is a software component that manages the execution of robot controllers, providing an interface with them. Therefore, it handles two important aspects: connecting the controller to its specific Hardware Interface to send correct command to a specific robot model in real time, and managing the load, stop and start processes of the loaded ROS controllers. This last issue is relevant because whenever a controller is load into the ROS Controller Manager it does not mean it is active. By default, if there is no other active controller it will be active. If on the contrary, there are more than one controller loaded and available compatible with the Hardware Interface of the controller, only one will be active at a time. This architecture allows commanding the robot without loosing any control signal, even when the controller manager is switching from one controller to another.

This encapsulation process of the Controller Manager ends up in a real time controller manager embedded inside an Orocos component, the RTT Controller Manager[1]. This modification allows the read()-update()-write() operations from the Hardware Interface to run into real time software component in a transparent way for the user. Hence, the depicted ROS Control lifecycle depicted in Figure 7.2 allows, in a periodic and

1: RTT means Real Time Toolkit and corresponds to the part of the Orocos library in charge of handling the real time capabilities of the component in order to make them transparent and easy to use for the user [135].

deterministic manner inside an Orocos component, the cyclic read of the data from the sensors, the periodic update of the control signal, and the continuous writing of the setpoint references into the robot actuators.

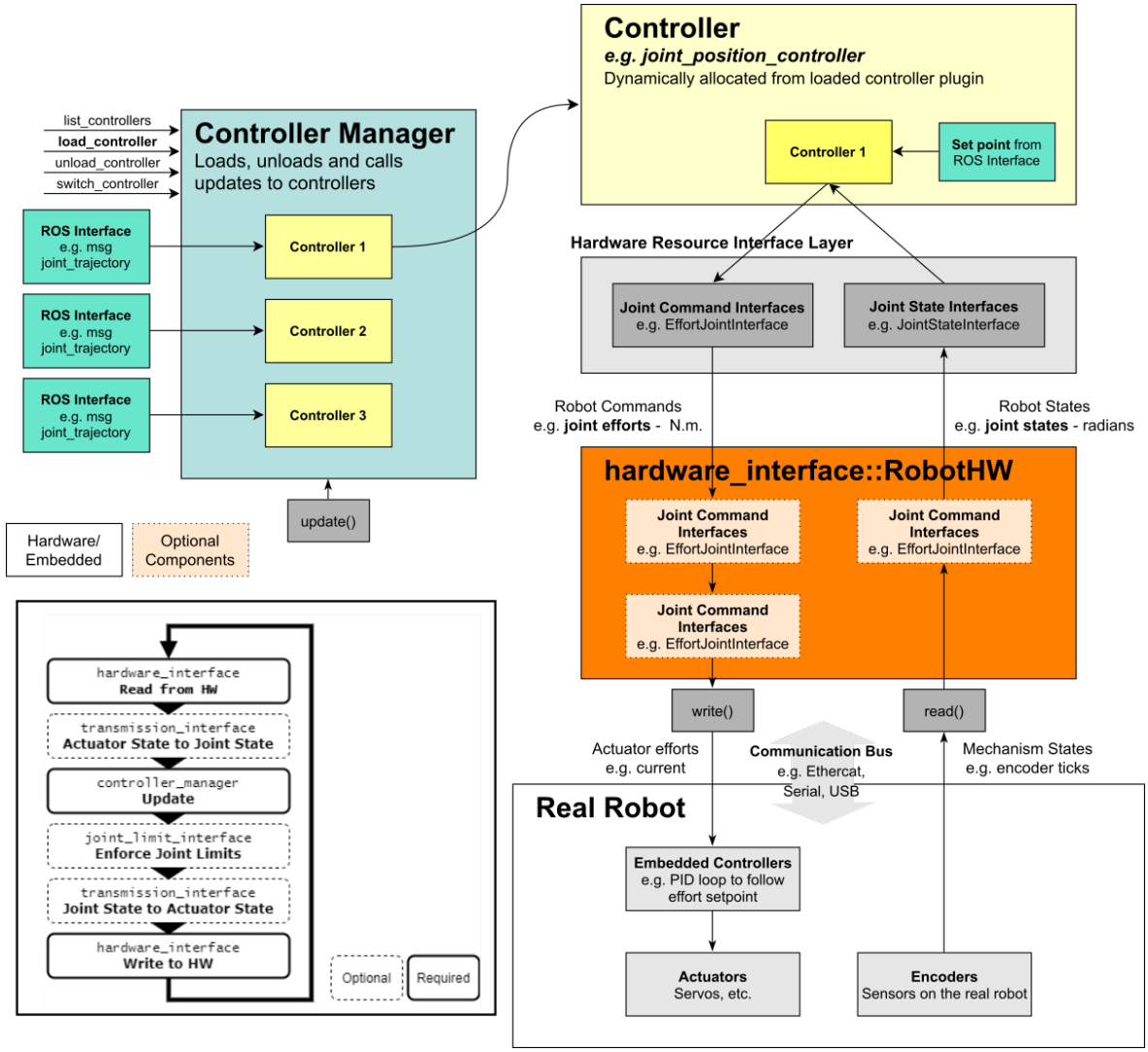

Figure 7.2: Steps of the ROS control loop [134].

Just by implementing the RTT Controller Manager is not enough to create a real time controller for the robot. This wrapper also requires a real time Hardware Interface instance called, the RobotHW, and a ROS controller to implement advanced control loop. In this manner, the RobotHW will implement the specific communications to read and write the signals from the sensors and the actuators respectively. In other words, it manages the data transmission between ROS and the robot control box. Additionally, the ROS controller implements the update function to compute the desired reference state for each joint of the manipulator. These relationship can be appreciated in Figure 7.3 where class diagram structure and inheritance are displayed. This encapsulation into classes of the ROS control capabilities is proposed specifically to fit the requirements of real time capabilities of the controller while allowing the hardware agnostic behavior of ROS framework.

Thanks to the class structure integrated as shown in Figure 7.3, the ROS

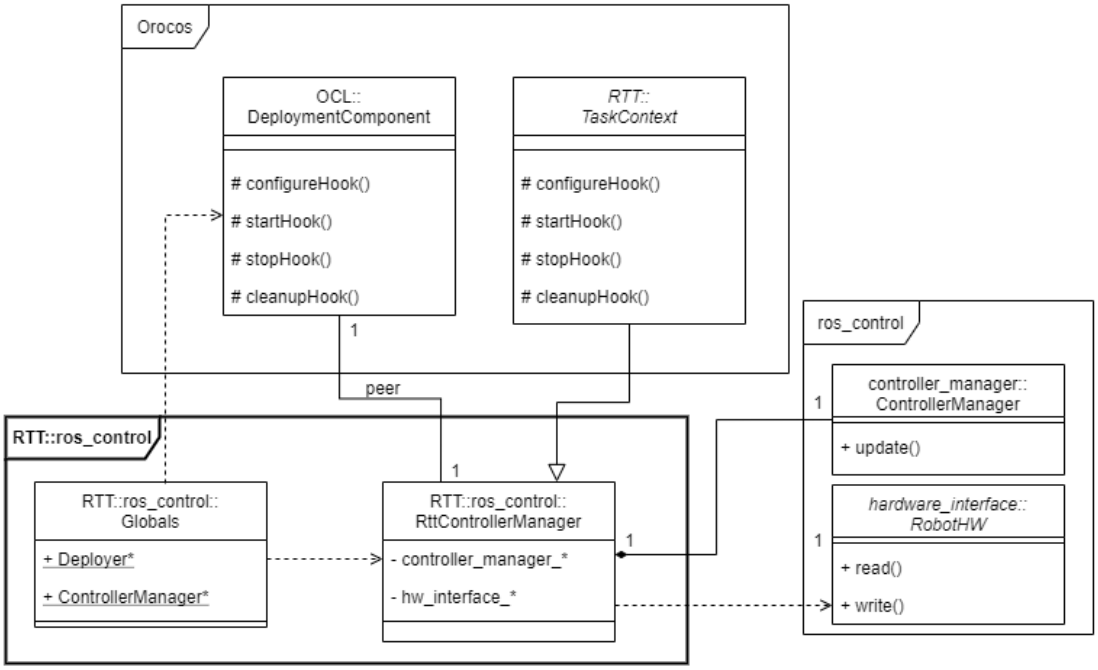

Figure 7.3: Class diagram of the real-time controller manager.

Control loop runs in the update function of a Task Context (running an Orocos component lifecycle as the one displayed in Figure 7.4) which is executed in a real time periodic activity that manages the control loop frequency as shown in Figure 7.5. In this manner, the controller manager is forced to go through the configure state transition of the Orocos component from Figure 7.4. Also, in Figure 7.5 it is shown that parallel threads can be launched to encapsulate additional ROS control based features into the Orocos components architecture. Hence, the RTT Task Context wraps the ROS control loop by having a reference to both: the RobotHW and the regular ROS instances as shown in Figure 7.3. Lastly, in order to communicate the ROS control encapsulated controller with the remaining nodes of the ROS network, the regular ROS spinner is also launched in a parallel thread, becoming the supervisor to check for the messages and events arriving at the ROS node exposed by the controller manager.

Then the wrapped component is configured to be in an Orocos ready state as seen in Figure 7.5 where each ROS control component has its own Orocos lifetime cycle. Thus, whenever the RTT activity is executing, the control loop will periodically call the read, write and update functions periodically in real time. In this part of the process, thanks to the ROS-Orocos integration package (the rtt_rosclock package), all the timing inputs for those functions are filled accessing to the ROS clock. It is worth mentioning that the real time capabilities of the control loop are bounded to the OS running in the control machine. If the OS does not present a real time kernel patch or a real time OS, the controller loop would not run in real time[2]. Therefore, this implementation has been made on an Ubuntu preempted kernel version, more specifically in the PREEEMT_RT kernel patch.

The main advantage of the proposed control architecture is that it keeps

2: In case the controller is implemented on a non-real time machine, the behavior of the controller will not trigger any error, but the determinism required for the read() and write() functions cannot be guaranteed. This behavior could lead to a fault or even stopping the robot movement due to an unstable command streaming.

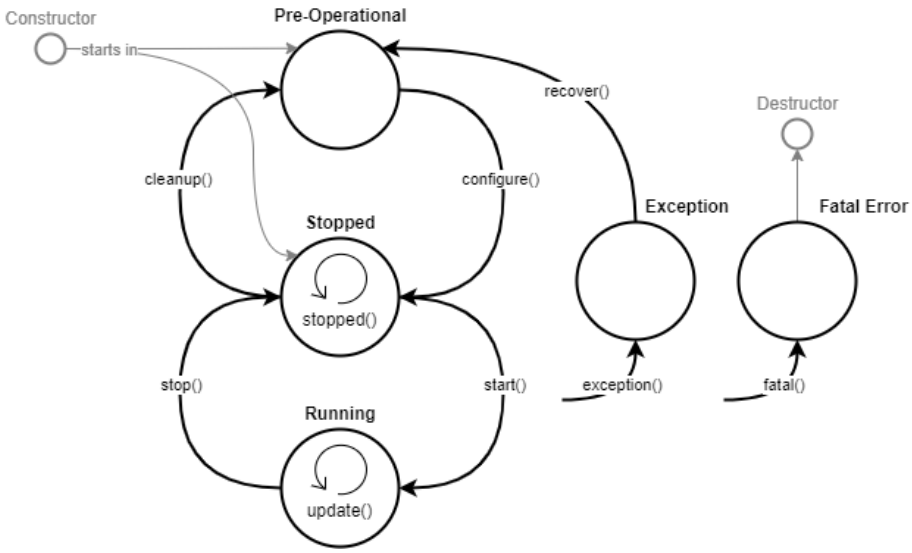

Figure 7.4: Orocos component lifecycle state machine [135].

the hardware agnostic software architecture from ROS, while combining it with transparent for the user real time capabilities given by Orocos. Thus, by utilizing the proposed control architecture, real time capabilities are provided to ROS Control controller seamless, allowing the interchangeability between controllers independently of the robot which is being used without the need for the programmer to worry about synchronizing real time software threads. In this manner, the proposed control

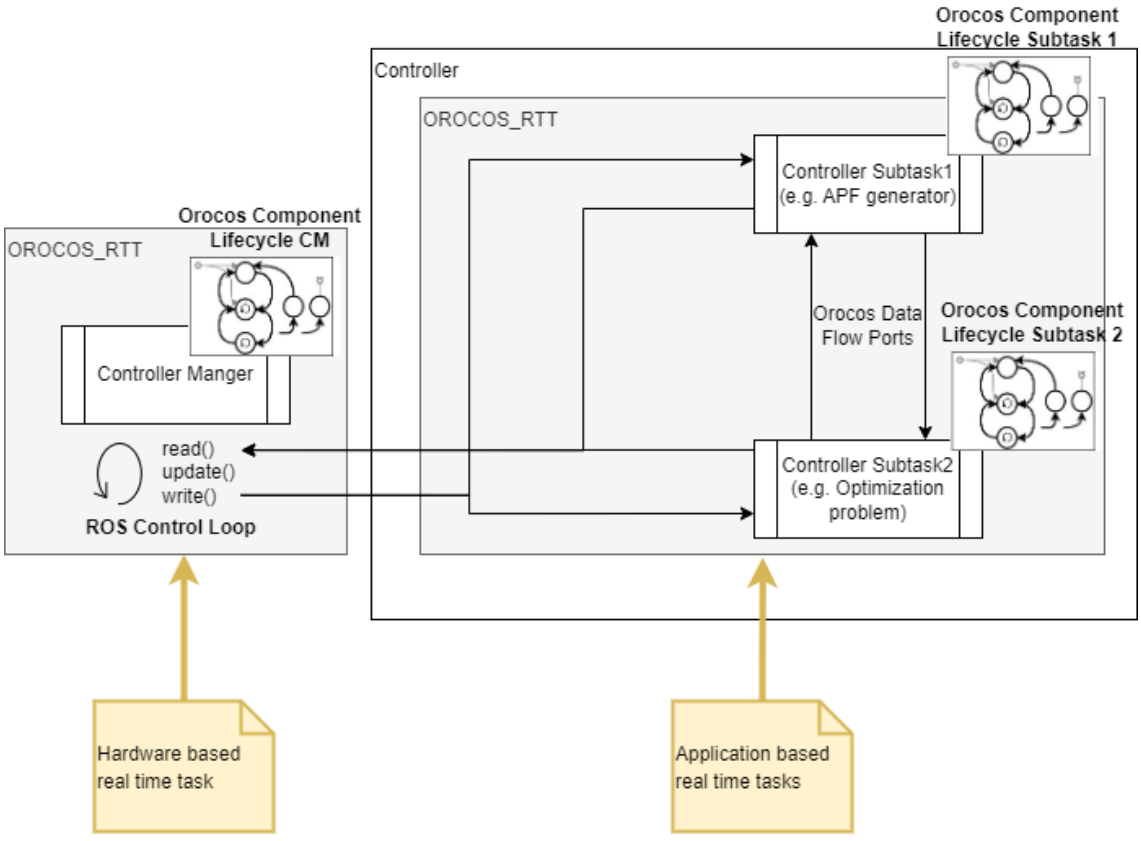

Figure 7.5: Generic ROS Controller encapsulation on Orocos components.

architecture offers a hardware agnostic software architecture to reutilize for different types and brands of commercial and non-commercial cobot. To keep on with the reusability of the code, the required knowledge about Orocos required has been reduced by the encapsulation of the instance of the RTT Controller Manager and the RobotHW in real time. This structure can be seen in the code below (Listing 1) where the function controller_manager_main() is implemented. This function requires a RobotHW instance to launch the corresponding controller manager containing it, fact that guarantees the re-usability of the RTT Controller Manager for different types of robots[3].

3: In this way, specific Hardware Interface can be implemented or reutilized if they came from a vendor ready online package to be run in this architecture.

The process of instantiation a RobotHW that integrates the RTT Controller Manager to run the ROS Control lifetime cycle in real time can be appreciated in Figure 7.6. Fisrt of all, if there is any RTT Controller already running, it will be directly instantiated the RobotHW interface managed by the RTT Controller Manager. Otherwise, the ROS network will be initialized as well as the Orocos OCL (Orocos Component Loader) Deployment Component. In this manner, the system is ready to load the RTT ROS Control capabilities and encapsulate them into the different Orocos components that would be created. At this point, the system is ready to load the RTT Controller Manager by instantiating an specific RobotHW Interface that runs the write() and read() methods of the ROS Control loop. Before running the RTT ROS Controller, the scheduler is configure to set the priority, the frequency of that Orocos component, or the capability to run in soft or hard real time, among other options. Once the Orocos component is configured, it can be finally launch the hardware agnostic RTT ROS Controller managed by the Orocos Task Browser.

Listing 1: Example usage of the setup tools for the real-time controller manager with a generic ROS-based robot Hardware Interface.

```
#include <rtt/os/main.h>
#include <rtt_controller_manager/main.h>

// Use the ORO_main macro to set up some platform
// specific settings
int ORO_main(int argc, char* argv[])
{
  // Substitute with implementation of RobotHW
  MyRobotHW robothw;

  // Start the controller manager Node
  RTT::ros_control::controller_manager_main(
    &robothw,             // RobotHW instance
    0.2,                  // Control loop Period
    "ControllerManager",  // Name of the RttControllerManager component
                          // & ROS node
    false                 // Interactive mode (use TaskBrowser
                          // component)
  );
}
```

Due to the generic Controller Manager encapsulation into Orocos implemented that enables the hardware agnostic ROS Control architecture, any of the target Hardware Interfaces for the UR10e or the SUPSI robot can be implemented. As it can be appreciated in Figure 7.2, the Hardware Interface is the one in charge of sending commands to the robots and reading

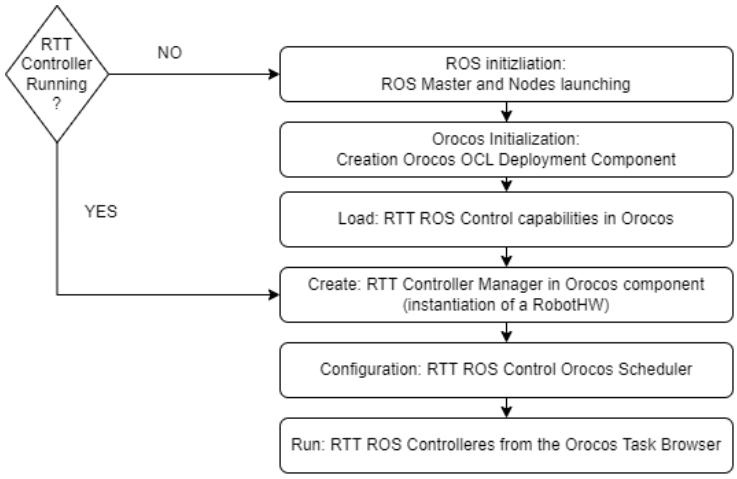

Figure 7.6: RTT Robot Controller initialization routine.

the robot current state. On top of that, a ROS Control controller should be implemented to gave capabilities such as the APF based repulsion to the robot. In this way, the part of the controller that gives the control architecture the specific hardware driver to communicate with each robot is the Hardware Interface, while the ROS Control controller brings the robot agnostic control laws. Therefore, implementing a compatible ROS Control controller is still required. To do so, the proposed software architecture relies on the usage of Orocos components to allow complex data and execution logic flows while allowing for highly modular approaches as shown in Figure 7.7. In general, these intrincated data flows require that the controller runs on a set of parallel tasks executed simultaneously. To face the required parallelism, it is convenient for the control loop to have separate threads with different configurable control frequencies. The synchronization of the different data flows and control signals to allow this modular behavior on the controller has been made through the RTT Orocos package by the implementation of a specific software component, the hardware interface bridge.

This hardware interface bridge designed also pretends to convey the information contained in the Hardware Interface from ROS Control instance to be available for the rest of the Orocos components that

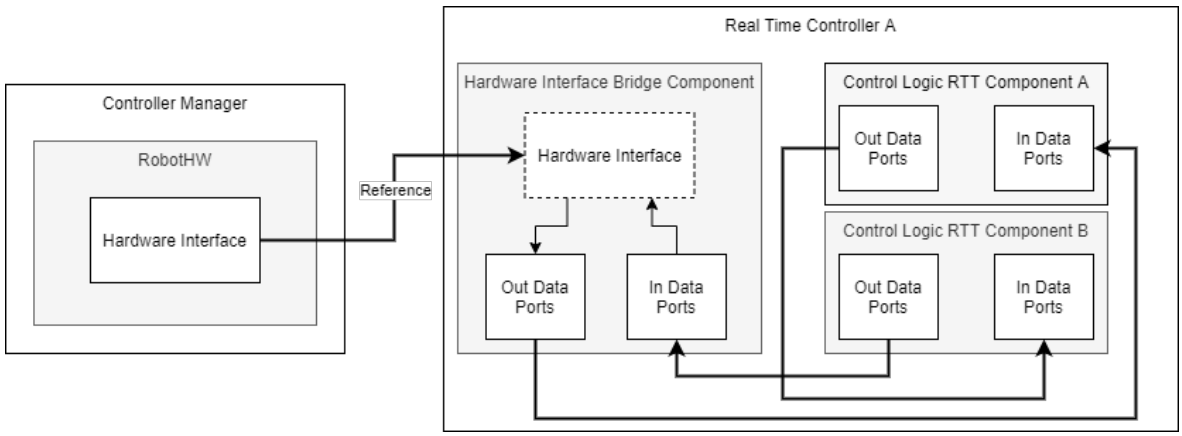

Figure 7.7: Real time encapsulation of a generic ROS control controller into Orocos components. The real time controller structure, in which the controller contains the hardware interface bridge component. The bridge component takes the hardware interface that is given to the controller when it is loaded and forwards the variables contained in it by exposing them to the rest of the Orocos components.

integrates parallel blocks of the controller. Consequently, as presented in Figure 7.7, a bridge component is required, providing RTT data ports to connect other Orocos components, and also, synchronizing all the Control Logic RTT components.

The bridge is a generic class that extends from the Orocos RTT Task Context and needs to be implemented for each specific hardware. It is a simple component that is instantiated during the loading process of a controller into the controller manager. Subsequently, the hardware interface bridge for the specific Hardware Interface compatible with the real time controller has to be implemented. Even though this implementation is specific for each Hardware Interface, every implementation shares the basic structure for instantiating the reference and loading all the configurations. Since the lifecycle of the ROS Controller coexist with the lifecycle of the Orocos component, the behavior of both have to be entwined and synchronized in a defined manner. Therefore, the conventions during the controllers lifetime are the followings:

- **Controller Stopped:** When the controller is instantiated as a plugin into the controller manager the bridge object is created within the init() function of the controller. During this stage, the Bridge component is also configured which takes it to its stopped state. The configuration done in this step includes registering the Orocos data ports and preparing the component for execution. This way of transmitting data guarantees real-time performance by avoiding dynamic memory allocation.
- **Controller Started:** When the controller is started the bridge is also started and goes into the running states. When the bridge is started checks are made on the data ports to make sure that they are connected, since, if they were not connected, the controller may incur in undefined behaviour. Connecting the data ports of the hardware interface bridge is a task that each controller implementation has to carry out by connecting the bridge component to the components that contain its control logic.

With all this considerations, a real time controller class is provided with generic references to a hardware interface bridge and pointers to the deployment controller and real time controller manager of the current process. This two last references allow the loading and registering of controller components that may not already be loaded into the system on the basic initialization. Subsequently, specific components that add functionalities to the original trajectory tracking controller can be implemented to run in parallel. In this thesis, these particular functionalities correspond to the specific APF implementation detailed in the following subsection of this chapter for each use case.

7.1.2 ROS network data structure

There are two types of data types among the data structure utilized in this work that correspond to real time data and the non-real time data. The real time data correspond to the ROS Control data and Orocos data ports to assure the deterministic response of the different components of the control architecture explained in the prior subsection. On the contrary, the non-real time data are related to ROS natural communication where

the determinism in the data exchange is not necessary. This second scenario that can be observed in Figure 7.8 under the Master-Slave ROS communication architecture is referred to ROS topics, services, actions, and the ROS Parameter Server stored configuration.

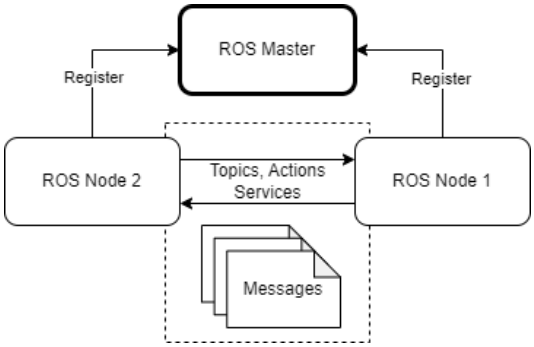

Figure 7.8: ROS Master data flow for nodes communication.

This combined data structure allows us to send control signals between the different components of the ROS controller while keeping the regular communication between nodes without violating the determinism in the control signal when required. To achieve these desired behaviors, all the different data matching the real time and non-real time distribution are shown in Figure 7.9, corresponding the blocks from the Trajectory Follower Component to the first group and the remaining components to the last one.

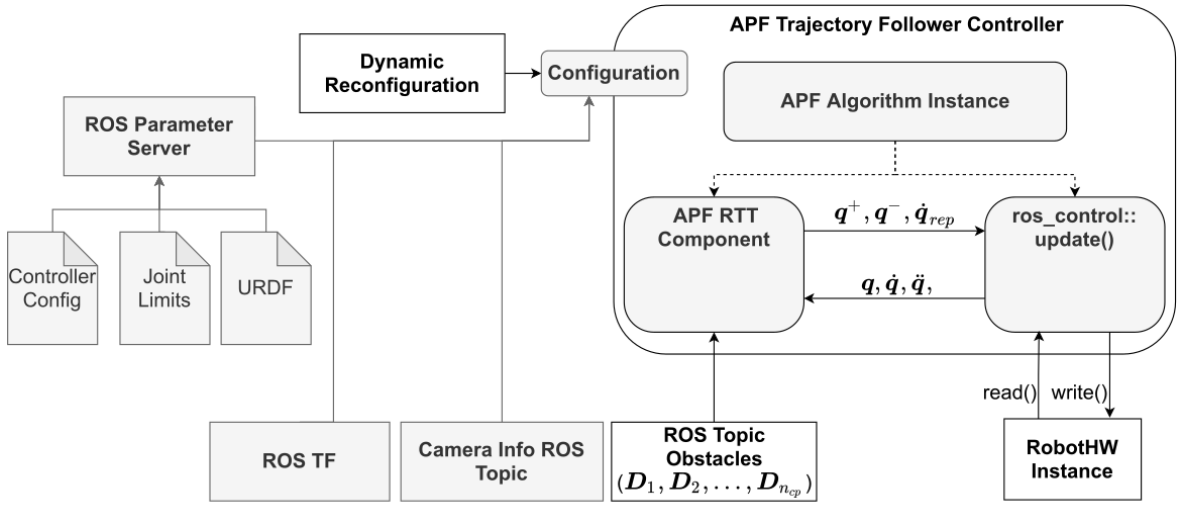

Figure 7.9: Software interfaces of the designed controller, greyed out to the left the sources for the configuration of the controller and on the right the internal structure of the controller and their data interfaces.

In the Trajectory Follower Controller block of Figure 7.9, it can be appreciated that the corresponding signals to this block are the ones related to the limiting repulsive joint velocities, the projected repulsive vector and the trajectory references. The representation from Figure 7.9 is a minimal parameter representation. However, all the signals described in Chapters 5 and 6 are grouped in this block. In addition, the data structured contained in the Trajectory Follower Controller that allows the robot to track a trajectory, also read from the ROS network the depth images processed to compute the repulsive signals, and the current configuration of the robot from the ROS Parameter Server. After computing a suitable control signal, all these data are packed into a compatible structure to feed

the robot controller via network communication, receiving the current state to be publish in the ROS Parameter Server.

The non-control related data correspond to the non-real time data division. The data structures that can be found are related to the ROS Parameter Server, the transformation between the different reference systems (ROS TF block from Figure 7.9, and the different ROS topics as the camera info ROS topics. This data types are required to load the robot configuration and create a virtual model of the robot that is a mirror of the current state of the robot and the different devices connected in the application. This information network allows the share of information and data between the different nodes, enabling the development of application that integrates the control aspects with the robot task and environment management.

7.2 UR10e particularization

The UR10e robot particularization is supported by the previous data structure, the RTT Controller Manager and the Hardware Interface bridge to implement a specific solution for this commercial robot. Since the proposed controller requires the particularization of the decoupled kinematic model proposed to allow the singularity repulsive components to be computed, this section addresses the particularization of the FK and IK, the singularity study, the specific hardware interface implemented, and, lastly, the software packages and dependencies for building the controller are also reviewed.

Regarding the FK and the IK, both are presented in the particularized kinematic behavioral study (kinematics and singularity) for each Universal Robot UR10e or similar structure robot in the first subsection. The goal of this first part is to obtained the closed set of solutions for the IK and the singular configurations during an offline study to apply this results to online simultaneous collision and singularity avoidance controllers. Subsequently, this solutions will be implemented in the specific hardware interface instance in the form of a RTT ROS Controller in the middle subsection. To aid this real time controller a set of libraries and software dependencies are required, being the ones belonging to the software packages and dependencies subsection. The following subsections addresses each of these aspects for the case of the UR10e robot.

7.2.1 Kinematic behavioral study

Since the proposed kinematic decoupling modifies the original robot kinematics for each specific cobot structure, Figure 7.10 shows the original reference system distribution from the ROS driver packages of each vendor. Figure 7.10 is also accompanied by Table 7.1 where the kinematic parameter distribution to build this model is presented. This structure will be employed for the implementation of the DLS-APF controller. However, as no kinematic modification is required in the URDF (Unified Robot Description Format) or configuration package, the DLS kinematic model is considered already explained in Subsection 5.2.5.

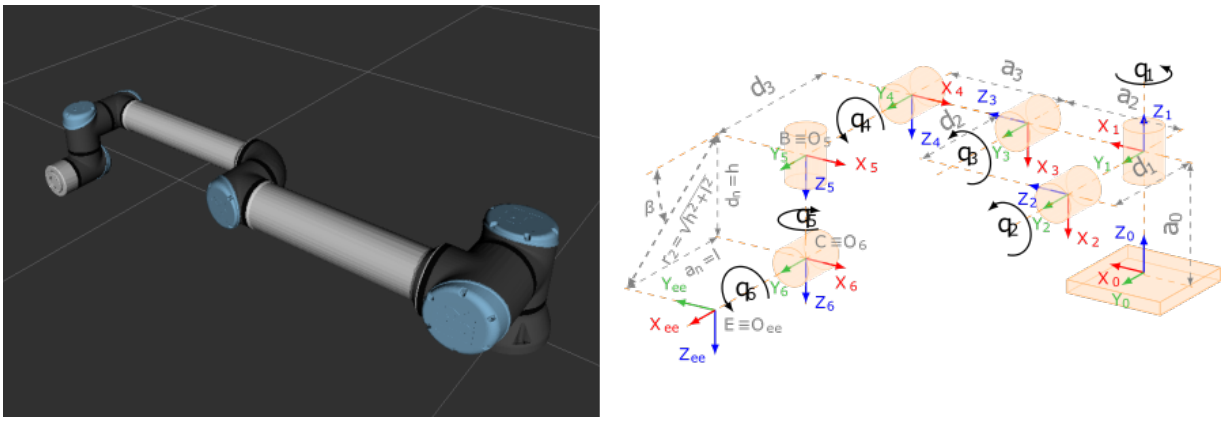

Figure 7.10: Kinematic model of the UR10e vendors package displayed in its resting position.

X [m]	Y [m]	Z [m]	θ_x [rad]	θ_y [rad]	θ_z [rad]
0	0	d_1	0	0	q_1
0	a_2	0	0	$q_2 + \pi/2$	0
0	$-a_3$	d_3	0	q_3	0
0	0	d_4	0	$q_4 + \pi/2$	0
0	a_5	0	0	0	q_5
0	a_6	d_6	q_6	0	0

[a] This table does not follow regular Denavit Hartenberg (DH) convention as normally do.
[b] The X, Y, and Z coordinates represent the Cartesian translation between reference systems.
[c] The θ_x, θ_y, and θ_z rotations reprents the rotations along the corresponding subindexed axes.

Table 7.1: Parameters for the kinematic model of the UR10e vendors package.

From the model presented above, by moving the 4-th and 6-th reference frames to $\mathbf{O}_5 = \mathbf{B}$ (the decoupling point), the kinematic reference distribution is transformed as follows as presented in the steps from Chapter 4. The decoupled kinematic model for non-spherical wrist cobots is displayed at Figure 7.11 with it bounded parameters from Table 7.2[4].

4: Being $r = \sqrt{a_6^2 + d_6^2}$, and $\beta = atan2\left(\frac{d_6}{a_6}\right)$.

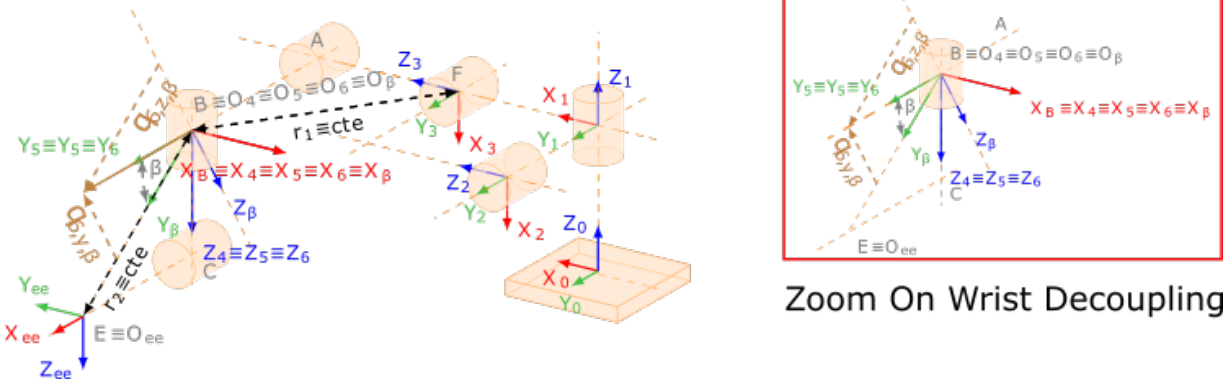

Figure 7.11: Decoupled kinematic model of the UR10e robot.

In the following subsubsection, the forward and inverse kinematics for the reference frame distribution of Figure 7.11 and Table 7.2 are addressed, as well as the particular singularity study.

Table 7.2: Parameters for the proposed decoupled FK model of UR10e

X [m]	Y [m]	Z [m]	θ_x [rad]	θ_y [rad]	θ_z [rad]
0	0	a_0	0	0	q_1
0	d_1	0	0	$q_2 + \pi/2$	0
0	$-d_2$	a_2	0	q_3	0
0	d_3	a_3	0	$q_4 + \pi/2$	0
0	0	0	0	0	q_5
0	0	0	0	q_6	0
Parameters of the O_β reference frame					
X [m]	Y [m]	Z [m]	θ_x [rad]	θ_y [rad]	θ_z [rad]
0	r	β	0	0	$\pi/2$

[a] This table does not follow regular Denavit Hartenberg (DH) convention as normally do.
[b] The X, Y, and Z coordinates represent the Cartesian translation between reference systems.
[c] The θ_x, θ_y, and θ_z rotations reprents the rotations along the corresponding subindexed axes.

7.2.1.1 Forward and Inverse Kinematics

In this subsection, the developments and theoretical formulation from Chapter 4 are employed to particularized the general purpose equations into the UR10e use case. The proposed decoupled kinematic model relies in the β auxiliary orientation from the Table 7.2 to obtain a vector that is always pointing to the end effector. Therefore, the correct usage of this auxiliary orientation angle will be displayed in the following section.

7.2.1.1.1 Forward Kinematics For the position and orientation forward kinematics, it is only required the traditional matrix postmultiplication between all the reference frames systems. In this manner, the position and orientation of the TCP can be straightforward computed as shown in previous equation 4.2[5]. By applying the aforementioned matrix postmultiplication, the whole model homogeneous transformation matrix can be computed as follows in equation 7.1.

5: Quick reminder of equation 4.2:

$$\mathbf{p}_E^0 = \mathbf{p}_B^0 + \mathbf{p}_{BE}^0 = \mathbf{p}_B^0 + \mathbf{R}_\beta^0 \mathbf{p}_{BE}^\beta$$

$$\mathbf{R}_E^0(\mathbf{q}) = \mathbf{R}_B^0(\mathbf{q}).$$

$$\mathbf{T}_6^0 = \mathbf{T}_1^0 \cdot \mathbf{T}_2^1 \cdots \mathbf{T}_5^4 \cdot \mathbf{T}_6^5 = \begin{bmatrix} (\mathbf{R}_6^0)_{3x3} & (\mathbf{p}_6^0)_{3x1} \\ (\mathbf{0})_{1x3} & 1 \end{bmatrix} \tag{7.1}$$

Since for the decouple model the desired position is the one from the decoupling point, it is held that $\mathbf{p}_6^0 = \mathbf{p}_B^0$, the arm position can be calculated as displayed in equation 7.2. In equation 7.2, all the parameters can be obtained from Table 7.2. In addition, the terms c_i and s_i represent the cosine and sine functions of the i-th joint, respectively. Being the $c_{i,\cdots,j}$ or $s_{i,\cdots,j}$ the addition of a cosine or a sine from the i-th to the j-th joint. Moreover, the parameter $d_0 = d_1 + d_2 + d_3$ corresponds to the arm-shoulder-wrist misalignments, and $Z_B^{0,*} = z_B^0 + a_0$ is the world-base frame Z axis translation whenever the world frame is not coincident with the robot base one.

$$\mathbf{p}_B^0 = \begin{bmatrix} x_B^0 \\ y_B^0 \\ z_B^{0,*} \end{bmatrix} = \begin{bmatrix} -(d_0) \cdot s_1 + (a_2 c_2 + a_3 c_{2,3}) \cdot c_1 \\ d_0 \cdot c_1 + (a_2 c_2 + a_3 c_{2,3}) \cdot s_1 \\ -a_2 s_2 - a_3 s_{2,3} \cdot c_1 \end{bmatrix} \tag{7.2}$$

Additionally, the rotation matrix obtained from $\mathbf{T}_6^0$ is computed as in equation 7.3. This rotation matrix results in a function dependant of all

the joints of the manipulator. However, since for the resolution of the IK only the reduced version for the arm orientation ($\mathbf{R}_3^0$) and the wrist orientation ($\mathbf{R}_6^3$) are employed, both symbolic expressions are evaluated instead, corresponding to equations 7.4 and 7.5[6], respectively in their most simplified format.

6: Remember that the following expression is held:

$$\mathbf{R}_6^0 = \mathbf{R}_3^0 \cdot \mathbf{R}_6^3$$

So it can be easily computed the $\mathbf{R}_6^0$ from equations 7.4 and 7.5.

$$\mathbf{R}_6^0 = \mathbf{R}_6^0(q_1, \cdots, q_6) = \begin{bmatrix} (\mathbf{R}_6^0)_{11} & (\mathbf{R}_6^0)_{12} & (\mathbf{R}_6^0)_{13} \\ (\mathbf{R}_6^0)_{21} & (\mathbf{R}_6^0)_{22} & (\mathbf{R}_6^0)_{23} \\ (\mathbf{R}_6^0)_{31} & (\mathbf{R}_6^0)_{32} & (\mathbf{R}_6^0)_{33} \end{bmatrix} \tag{7.3}$$

$$\mathbf{R}_3^0 = \begin{bmatrix} -s_{2,3}c_1 & -s_1 & c_{2,3}c_1 \\ -s_{2,3}s_1 & c_1 & c_{2,3}s_1 \\ -c_{2,3} & 0 & -s_{2,3} \end{bmatrix} \tag{7.4}$$

$$\mathbf{R}_6^3 = \begin{bmatrix} -c_4s_6 - c_5c_6s_4 & s_4s_5 & c_4c_6 - c_5s_4s_6 \\ c_6s_5 & c_5 & s_5s_6 \\ s_4s_6 - c_4c_5c_6 & c_4s_5 & -c_6s_4 - c_4c_5s_6 \end{bmatrix} \tag{7.5}$$

Once the position and orientation are computed in terms of the joints of the manipulator, the velocity computation can be addressed. For the velocity computation, the Jacobian matrix of the manipulator should be computed as displayed in equation 4.10. The symbolic evaluation of the Jacobian matrix for the UR10e robot is displayed below in equation 7.6:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} (\mathbf{J}_{11})_{3\times3} & \mathbf{0}_{3\times3} \\ (\mathbf{J}_{21})_{3\times3} & (\mathbf{J}_{22})_{3\times3} \end{bmatrix} \tag{7.6}$$

Since the jacobian computation is straightforward and the evaluation contains large terms, the symbolic evaluation has been excluded from this document. This expresion of the jacobian will be latter employed on Subsubsection 7.2.1.2.

7.2.1.1.2 Inverse Kinematics In order to solve the IK of the manipulator and find a suitable set of closed solutions, it is relevant to keep in mind that the solutions are related to the decoupling point[7]. Following the steps presented in Chapter 4, the first thing to do is to calculate the algebraic solutions for the arm positioning. For such a purpose, the following three relationships has been applied leading to their corresponding solutions for each joint[8]:

1. $(x_B^0)^2 + (y_B^0)^2 + (z_B^{0,*})^2$ for the solutions related to q_3 (equation 7.9).
2. Combining of $(x_B^0)^2 + (y_B^0)^2$ with $z_B^{0,*}$ to solve the solutions of q_2 (equation 7.8).
3. The study standalone of x_B^0 and y_B^0 for the solutions corresponding to q_1 (equation 7.7).

7: Similar to what happens with the FK, the consequence of this fact is due to the following relationships from equation 4.3:

$$\mathbf{p}_B^0 = \mathbf{p}_E^0 + \mathbf{R}_\beta^0(-\mathbf{p}_{BE}^\beta)$$
$$\mathbf{R}_B^0(\mathbf{q}) = \mathbf{R}_E^0(\mathbf{q}).$$

8: Remember that x_B^0, y_B^0 and z_B^0 are the positioning component of $\mathbf{x}_B^0$ from equation 4.4. This component corresponds to $\mathbf{p}_B^0$. Moreover, the component $z_B^{0,*} = z_B^0 - a_0$ to compensate the world-base reference frame displacement.
In addition, the following simplifications has been considered in equations from 7.7 to 7.9: $b_1 = a_2^2 + a_3^2 + d_1^2 + 2(d_1d_2 + d_1d_3) + (d_2 + d_3)^2$, $b_2 = 2a_2a_3$, $\lambda = (x_B^0)^2 + (y_B^0)^2 - d_0^2$, $\alpha_1 = a_2 + a_3c_3$, $\alpha_2^+ = a_3s_3^+$, and $\alpha_2^- = a_3s_3^-$, being s_3^+ or s_3^- the positive or negative solution of equation 7.9d, respectively.

$$q_{1,I} = atan2(y_B^0\sqrt{\lambda} - x_B^0d_0, y_B^0d_0 + x_B^0\sqrt{\lambda}) \tag{7.7a}$$
$$q_{1,II} = atan2(-y_B^0\sqrt{\lambda} - x_B^0d_0, y_B^0d_0 - x_B^0\sqrt{\lambda}) \tag{7.7b}$$

$$q_{2,I} = atan2(-z_B^{0,*}\alpha_1 - \alpha_2^+\sqrt{\lambda}, \alpha_1\sqrt{\lambda} - z_B^{0,*}\alpha_2^+) \tag{7.8a}$$
$$q_{2,II} = atan2(-z_B^{0,*}\alpha_1 + \alpha_2^+\sqrt{\lambda}, -\alpha_1\sqrt{\lambda} - z_B^{0,*}\alpha_2^+) \tag{7.8b}$$
$$q_{2,III} = atan2(-z_B^{0,*}\alpha_1 - \alpha_2^-\sqrt{\lambda}, \alpha_1\sqrt{\lambda} - z_B^{0,*}\alpha_2^-) \tag{7.8c}$$
$$q_{2,IV} = atan2(-z_B^{0,*}\alpha_1 + \alpha_2^-\sqrt{\lambda}, -\alpha_1\sqrt{\lambda} - z_B^{0,*}\alpha_2^-) \tag{7.8d}$$

$$q_{3,I} = atan2(s_3, c_3) \tag{7.9a}$$
$$q_{3,II} = -q_{3,I} = atan2(-s_3, c_3) \tag{7.9b}$$
$$c_3 = \frac{(x_B^0)^2 + (y_B^0)^2 + (z_B^{0,*})^2 - b_1}{b_0} \tag{7.9c}$$
$$s_3 = \pm\sqrt{1 - c_3^2} \tag{7.9d}$$

Though there are a total of 10 solutions divided in 2 solutions for q_I, 4 solutions in the case of q_{II}, and others 4 solutions in the case of q_{III}, not every combination is possible. Indeed, the only possible combinations corresponds to the i) shoulder-right/elbow-up, ii) shoulder-left/elbow-up, iii) shoulder-right/elbow-down, and iv) shoulder-left/elbow-down configurations. It means that only the four possible combinations are allowed:

1. Solution for i): $[q_{1,I}, q_{2,I}, q_{3,I}]$.
2. Solution for ii): $[q_{1,I}, q_{2,III}, q_{3,II}]$.
3. Solution for iii): $[q_{1,II}, q_{2,II}, q_{3,I}]$.
4. Solution for iv): $[q_{1,II}, q_{2,IV}, q_{3,II}]$.

It is also remarkable, that for the solutions of equations 7.7 to 7.9, when there is no arm-shoulder-wrist misaligment between the reference frames ($d_0 = d_1 = d_2 = d_3 = 0$), and the world-base offset is null ($a_0 = 0$), the solution matches is similar in form to the ones given by [109] for the anthropomorphic arm, so it can be seen as an additional verification of the adequacy of the obtained solution.

Moreover, to address the IK for the orientation, the equation 7.5 is required. Combining the several terms from different rows and columns of the expression that relates a generic ZYZ Euler rotation to the wrist joints, the solutions for the inverse kinematics of the orientation can be obtained. Therefore, applying these considerations for the positive or negative root of γ whether $q_5 \in (0, \pi)$ or $q_5 \in (-\pi, 0)$, respectively, these solutions can be found [9] as shown in 7.10.

9: Note that $\gamma = +\sqrt{s_{2,1}^2 + s_{2,3}^2}$ where $s_{m,n}$ represents the m-th row and n-th column of the genereic (Euler ZYZ) $\mathbf{R}_6^3$ matrix.

$$q_{4,I} = atan2(s_{1,2}, s_{3,2}); \text{ or } q_{4,II} = atan2(-s_{1,2}, -s_{3,2}) \tag{7.10a}$$
$$q_{5,I} = atan2\left(\gamma, s_{2,2}\right); \text{ or } q_{5,II} = atan2\left(-\gamma, s_{2,2}\right) \tag{7.10b}$$
$$q_{6,I} = atan2(s_{2,3}, s_{2,1}); \text{ or } q_{6,II} = atan2(-s_{2,3}, -s_{2,1}) \tag{7.10c}$$

In this occasion, the results can only be gather between the analogous solutions. It means that $q_{i,I}$ represent the same set of correlated solutions, and $q_{i,II}$ also replicates this behavior.

Lastly, since solving the IK for the case of the velocity (the inverse function of the Jacobian), the relevant results are strictly bounded to the singular configurations. Therefore, the following subsubsection addresses specifically the singularity study for the UR10e with the aim of obtaining an applicable set of closed solutions for the singular configurations of the robot.

7.2.1.2 Singularity Study and characterization

The characterization and study of the singularities arises from equation 7.6 where it can be seen that the results of the singularities cna only be a function of the $|(\mathbf{J}_{11})_{3\times3}|$ or the $|(\mathbf{J}_{22})_{3\times3}|$. More specifically, the manipulator will be on a singular configuration whether $|(\mathbf{J}_{11})_{3\times3}| = 0$ or $|(\mathbf{J}_{22})_{3\times3}| = 0$ as appreciated in equation 7.11.

$$|\mathbf{J}| = 0 \Rightarrow |(\mathbf{J}_{11}(q_1, q_2, q_3))_{3x3}| \cdot |(\mathbf{J}_{22}(q_4, q_5, q_6))_{3x3}| = 0 \tag{7.11}$$

Since the robot will be in a singular configuration whether the first or the second term of the right-hand side of equation 7.11, the singularities can be split into two groups: the arm singularities and the wrist singularities. The first kind are referred to the solutions of $|(\mathbf{J}_{11}(q_1, q_2, q_3))_{3x3}| = 0$. Therefore, they depend exclusively in the first three joints of the manipulator. Furthermore, the second type are related to the wrist singularities. Since they correspond to the solutions of $|(\mathbf{J}_{22}(q_4, q_5, q_6))_{3x3}| = 0$, they depend on the last three joints of the manipulator.

Starting with the arm singularities, for the particular use case of the UR10e robot, the solutions from equation 7.12 can be computed for the relation described above. In addition, Figure 7.12 represent a heat map where the zeros of the equations are coloured in red (for the solutions relative to q_3 strictly) and in black (for the solutions involving non clear dependency of q_2 and q_3). The zeros from the figure are found using mathematical methods and a grid of values for q_2 and q_3.

$$\begin{aligned}|\mathbf{J}_{11}| = f(q_2, q_3) = 0.21\cos(q_2)\sin(q_3) - 0.19\sin(q_2) + \\ + 0.19\cos^2(q_3)\sin(q_2) + 0.19\cos(q_2)\cos(q_3)\sin(q_3)\end{aligned} \tag{7.12}$$

From the two types of solutions in Figure 7.12, the red coloured ones corresponds to solutions of $\{-\pi, 0, \pi\}$. These solutions are referred to the solutions of the sine function ($f_{q_{3,sing}} = \sin(q_3)$). Extracting the sine of the third joint from equation 7.12 as a common factor, the remaining expression still depends on q_2 and q_3 as displayed in equation 7.13[10].

$$|\mathbf{J}_{11}| = f(q_3) \cdot h(q_2, q_3) = \sin(q_3) \cdot (0.19986\cos(q_2 + q_3) + 0.21456\cos(q_2)) \tag{7.13}$$

Any combination that nullifies $f(q_3)$ or $h(q_2, q_3)$ is considered a singular configuration of the arm. Furthermore, if the wrist singularities are studied in a similar way, after computing and simplifying the determinant of $|(\mathbf{J}_{22}(q_4, q_5, q_6))_{3x3}| = 0$, the solution from equation 7.14 is calculated.

10: Note that the correspondence between the functions is as follows:

$$f(q_3) = \sin(q_3)$$

$$h(q_2, q_3) = 0.19986\cos(q_2+q_3)+0.21456\cos(q_2)$$

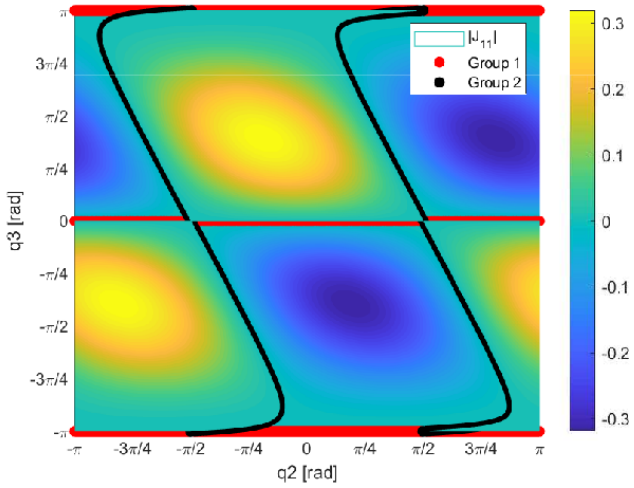

Figure 7.12: Graphical representation of the found zeros for the case of $|(\mathbf{J}_{11}(q_1, q_2, q_3))_{3x3}| = 0$ in the UR10e robot decoupled kinematics.

Analogous to the previous case, the Figure 7.13 depicts the solutions for this equation. In this case, the solutions found are simpler than the previous case, since $|\mathbf{J}_{22}| = g(q_5)$.

$$|\mathbf{J}_{22}| = g(q_5) = -\sin(q_5) \tag{7.14}$$

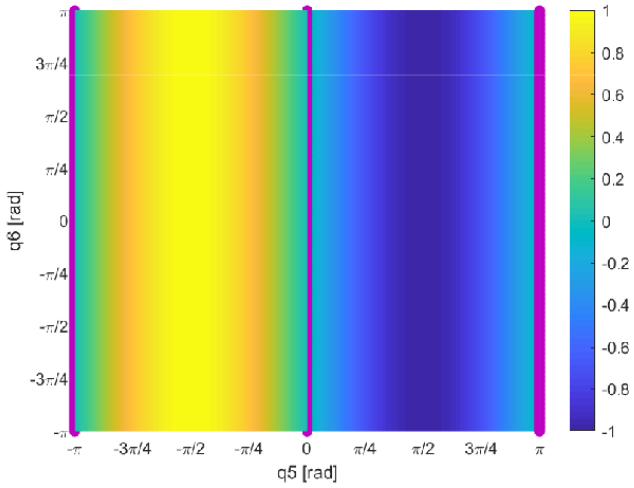

Figure 7.13: Graphical representation of the found zeros for the case of $|(\mathbf{J}_{22}(q_4, q_5, q_6))_{3x3}| = 0$ in the UR10e robot decoupled kinematics.

In this manner, the parametrization of the singular configurations correspond to the joint dependant functions that nullyfies the Jacobian matrix or, in other words, the solutions of $h(q_2, q_3)$, $f(q_3)$ and $g(q_5)$. To sum up and gather the singular configurations on a single expression, the equation 7.15 is utilized.

$$|\mathbf{J}| = 0 \iff \begin{cases} h(q_2, q_3) = 0.19986\cos(q_2 + q_3) + 0.21456\cos(q_2)) \\ f(q_3) = \sin(q_3) = 0 \\ g(q_5) = -\sin(q_5) = 0 \end{cases} \tag{7.15}$$

From the characterized solutions of equation 7.15, there are two types of solutions that matches with the ones of the literature from [109]. These

solutions corresponds to the functions f and g. However, the remaining kind does not apparently match with any know type of singularity. Therefore, in Chapter 8 a singularity study comparison is executed against the driver model to check whether they can be consider a new type of singularity or not. These solutions are the ones implemented in equation 6.4 to compute the distance to the singularity. In the case of no joint related to any singularity the function $f(|\mathbf{J}_{ii}|)_{q_i} = 0$, while for the h function the distance will be studied according the the second joint (q_2).

Lastly, it is also worth mentioning that even though the singularities of $\mathbf{J}_{11}$ and $\mathbf{J}_{22}$ may happen simultaneously, they are not coupled singularities. It means that even though there is a case where both determinant are in a singularity, each singular configuration is independent from the other.

7.2.2 Hardware Interface and RTT ROS controller

The specific Hardware Interface implemented on top of the hardware interface bridge for the specific use case of the UR10e robot is based on the original Universal Robots ROS Drivers for ROS Melodic [127]. This ROS driver is a real time ready ROS Controller, which means that it has real time capabilities but no real time management inside. To add the real time capabilities, the original Universal Robots ROS driver has been embedded into a RTT ROS Controller employing the software architecture explained prior in this chapter. This encapsulation has been made following the class diagram from Figure 7.14.

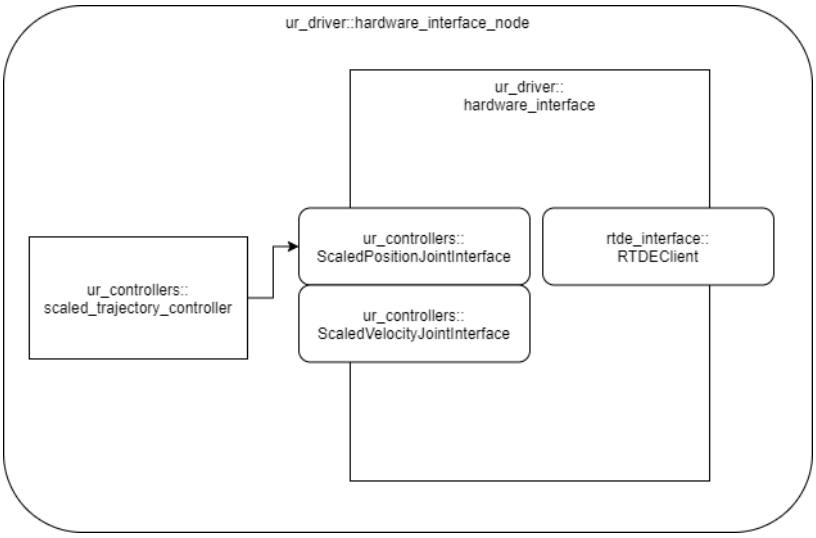

Figure 7.14: ROS Control class diagram for the UR10e robot developed hardware interface.

As it can be appreciated above in Figure 7.14, the proposed Hardware Interface inherits from a scaled_trajectory_controller to create a interface to command in position or velocity the UR10e. Then the commands are sent to the robot through the RTDE Client library[11]. This hardware interface is implemented on an Orocos component as displayed in Figure 7.21[12] thanks to the hardware interface bridge. More specifically the ros_control::update() component is in charge of coordinating the trajectory references from MoveIt with the downscaling and repulsive magnitude coming from the APF RTT Component, and computing then with the Ruckig library a suitable signal for the robot controller.

11: The RTDE Client library is the vendors official library to send commands to the robot through the Real Time Data Exchange interface through a standard TCP/IP connection. This interface is also employed to read the current status of the robot.

12: It has been placed at the end of the section in favour of the comprehension for the reader.

In parallel to this first component runs the RTT APF Orocos component. This component is responsible of computing the repulsive influence due to the obstacle and the singularity (in the case of the d-APF controller), as well as the limiting velocities due to obstacle and singularity risk. With the following coordination of ROS components it is achieved the load of the RTT ROS Controller into the RTT ROS Controller Manager as displayed in Figure 7.15.

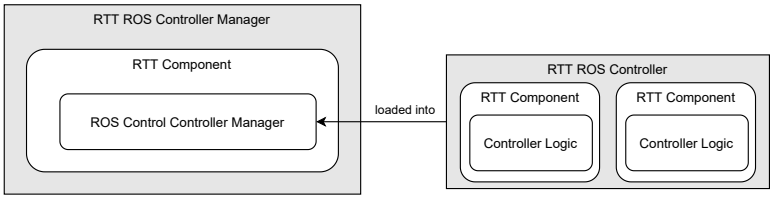

Figure 7.15: Block Diagram of the RTT ROS Control integration showing the RTT Controller Manager and the model of an RTT ROS Controller.

To implement each of the components above, several libraries and tools independent of ROS and Orocos have been employed for each controller parallel component. First, for the vision processing purpose the most well known vision library is employed, the OpenCV [136]. In addition, for the implementation of the matrix calculation, the Eigen3 library is employed [137]. Moreover, the Boost C++ libraries [138] have been utilized for managing general convenience utilities as the smart pointer, or containers, among other utilities. To ease the integration of the robot kinematics and dynamics the KDL library has been used [139]. And lastly, as mentioned earlier, to handle the online dynamics of the robot the Ruckig library has been used.

With all this software packages and libraries two software components have been developed: the apf_algorithms library and the rtt_ur_controllers package. The apf_algorithms package is a self-developed library that implements the different control algorithms to compute the references related to the collision and the singularity avoidance explained in Chapters 5 and 6. This library increases the overall flexibility of the software implementation, since changes can be directly done in the algorithms without modifying the behavior of the ROS Control controllers. This library is then utilized by the rtt_ur_controllers package to implement both Orocos based ROS control components from above inheriting from the ur_controllers of the original Universal Robots ROS drivers. This last package also relies in the kdl_parser to load the KDL kinematic model from the URDF files loaded into the ROS Parameter Server. In Figures 7.16 and 7.17, both software packages and dependencies can be appreciated for the DLS-APF controller and the d-APF one, respectively. In the case of the DLS-APF controller, an additional component is appreciated that corresponds to the wdls_algorithms library to implement the DLS kinematic model from the Orocos based packages. Moreover, it is not a package *per se* but the robot_state_publisher[13] has been modified for the d-APF controller, resulting in the decoupled_robot_state_publisher. The goal of this modification is to compensate the influence of modifying directly the URDF of the UR10e to implement a new reference frame distribution (what generates an eccentric rotation on the 6-th axis). In this way, the kinematic description of the robot will be automatically updated by the ROS Parameter Server when loading the robot.

13: The aim of this ROS node is to read the status of the different joints states of the manipulator to compute the transform of each joint related refernce systems.

Lastly, the Hardware Interface still requires a particular RobotHW interface to be simulated in Gazebo. This virtual RobotHW interface aims to

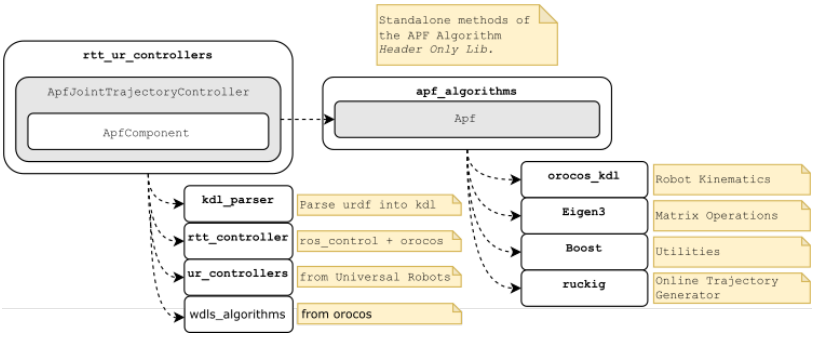

Figure 7.16: Software packages that contain the ROS controllers and their dependencies for the DLS-APF controller.

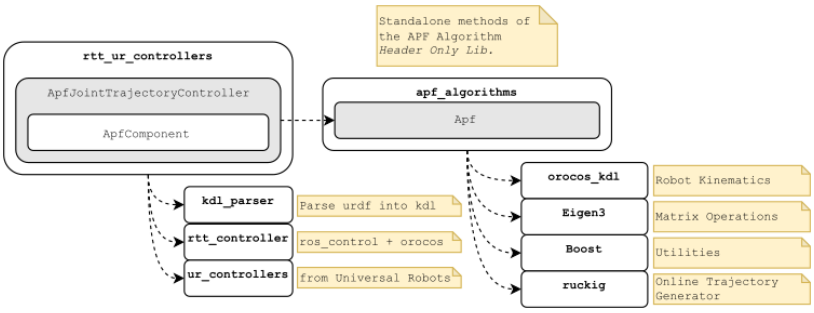

Figure 7.17: Software packages that contain the ROS controllers and their dependencies for the d-APF controller.

expose the right ports to which connect the scaled_joint_interface created in the ROS controller. To do so, the URRobotHWSim has been implemented using as reference model the generic DefaultRobotHWSim from Gazebo. Therefore, our version of the gazebo_ros_packages has also been implemented and compiled including the new simulated interface[14].

14: Bear in mind that if no new RobotHW interface is created, the standard ROS Control Hardware Interface can bu directly used, required no modified version of the gazebo_ros_packages and plugins.

7.2.3 Vision algorithm

Complementary to the kinematic modelling exposed in Subsection 7.2.1 and the Hardware Interface for the implementation of a RTT ROS Controller of Subsection 7.2.2, the proposed approach could not be complete without a proper scene segmentation to avoid just what is considered as an obstacle. The aim of the vision algorithm employed is to distinguish between what is part of the scene, which elements are workpiece to interact with, and what can be considered as an obstacle (typically a human operator) that must be avoided. The main contribution of the employed vision algorithm consists of a lightweight robot filter based on the virtual information of the URDF combined with the current configuration of the robot, followed by a dynamic/static image filter that segments the remaining components of the scene into the three aforementioned groups. Indeed, this is also a novel contribution which faces the fact that the robot can be either a moving or a stuck element of the scene that must be filtered prior to analyse the rest of the scene.

In order to address the vision processing from the depth maps given by the Intel Realsense D435, three different vision algorithms have been implemented. First, an algorithms for simulation purposes has been created. This segmentation algorithm is based in a simple HSV colour filter to filter an green sphere that can be static or dynamic without stressing the device where the simulation is running. This simple but effective filter allows to test in simulation the suitability and behavior of the controller before testing the algorithms in the real robot. Consequently,

this lightweight filter even though it is a simple one it suits perfect the aim of the simulation. Furthermore, two additional filtering algorithms have been implemented for the real scenario. Both approaches are based on a filter that combines the information of the URDF description or the transformation of the robot to filter the robot from the depths maps.

The first of the two approaches, is based on the already real time urdf filter from github [140]. This filter uses the information of the URDF and the ROS Parameter Server to create a virtual mask containing the information where the robot is expected to be. This virtual based mask is then applied to the real depth map to filter out the robot. To make it work properly in our implementation, some timing filtering parameter was added to avoid the robot of repelling itself while moving. Even though this filter works well as displayed in Figure 7.18, it makes no segmentation between the scene, the static workpieces, and the obstacles. This behavior treat everything in the surroundings of the robot as an obstacle evading the workpiece, so it does not suit the application requirement.

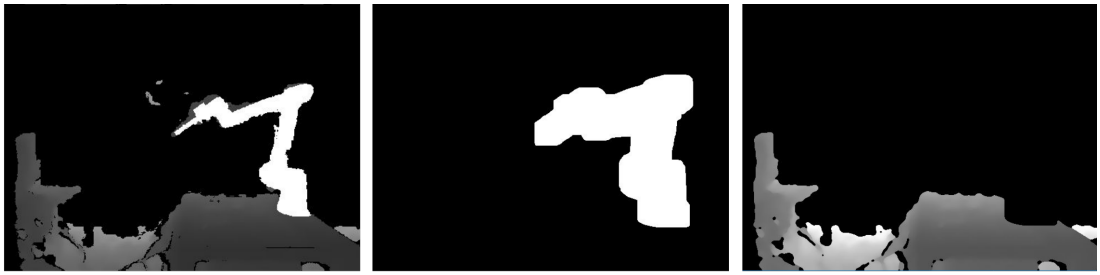

Figure 7.18: Example of the first realtime URDF filter:(a) Sample frame filtered by the URDF filter, where the gray *shadow* around the filtered parts of the robot is unwanted. (b) Example the mask generated during the postprocessing stage of the depth image filtering where the robot silhouette is dilated. (c) Frames that visualize the postprocessing stage of the depth image filtering process.

To overcome the aforementioned limitation of discerning from among the dynamic and the static elements of the scene, another virtual URDF segmentation filter has been proposed. In this occasion, the filter has been fully implemented with the aim of segmenting the robot from the depth map but the dynamic and the static elements too. This algorithm is supported by the colour image of the camera, so the depth map has been aligned to the colour frame. As it can be seen in Figure 7.19, the algorithm starts reading the colour image to create a model or update the model of the background reference image. Then, the depth image that arrives from the ROS camera driver nodes is read to virtually compute the position of the homogeneous transformation matrices (tf) of each link reference system and represented into this depth picture. By knowing the depth information of each link thanks to the previous positioning task, the robot can be filtered from the image. In this way, the remaining depth image contains the scene, workpiece, and the obstacle (as displayed in Figure 7.20). To perform the desired advanced segmentation desired into these three aforementioned groups, the background model is utilized as a mask to know the static elements of the depth map that belongs to the scene. Then, thanks to a image buffer filter, the static work pieces and the dynamic obstacles are distinguish and segmented. Thus, three different depth images are generated one with only the scene information, another one with the positioning data of the workpieces, and the last one containing the obstacles of the surroundings of the robot. These depth

maps are forwarded into the ROS network to be utilized by the controller or any other node.

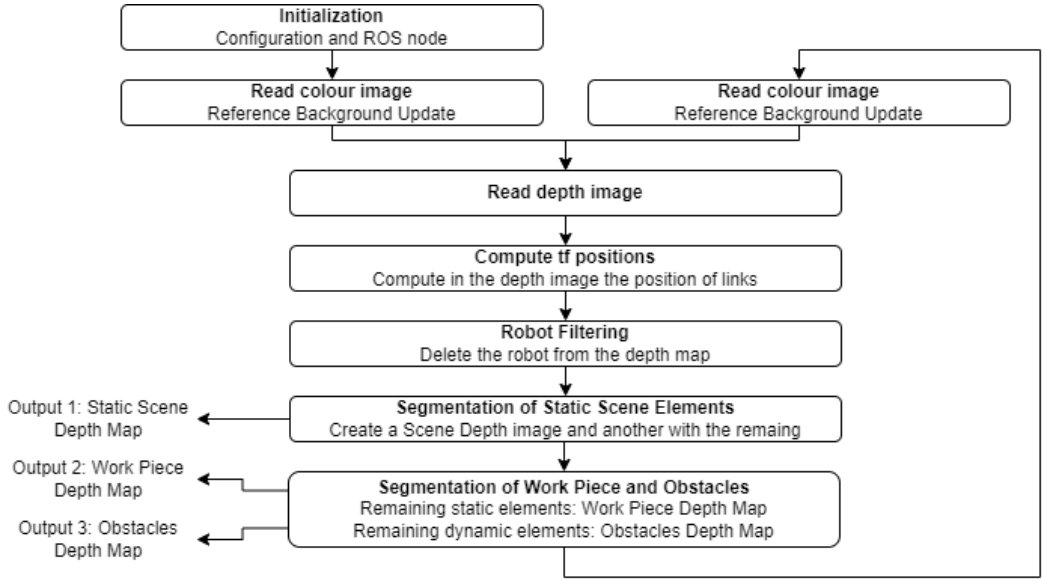

Figure 7.19: Flow chart of the final advanced scene segmentation algorithm. In the figure tf means transformation and is referred to the homogeneous transformation matrix that contains the information of each joint and link positioning of the robot.

The main problem both filters present is the strong dependency on a good camera positioning in the virtual environment according to the real world scenario. Therefore, a calibration package (camera_autopos) has been created to automatically obtain the position and orientation of the cameras with respect to a ArUco marker [141] thanks to a created diamond marker. With this relative positioning, the description of the URDF can be modified so the real and the simulated environment matches.

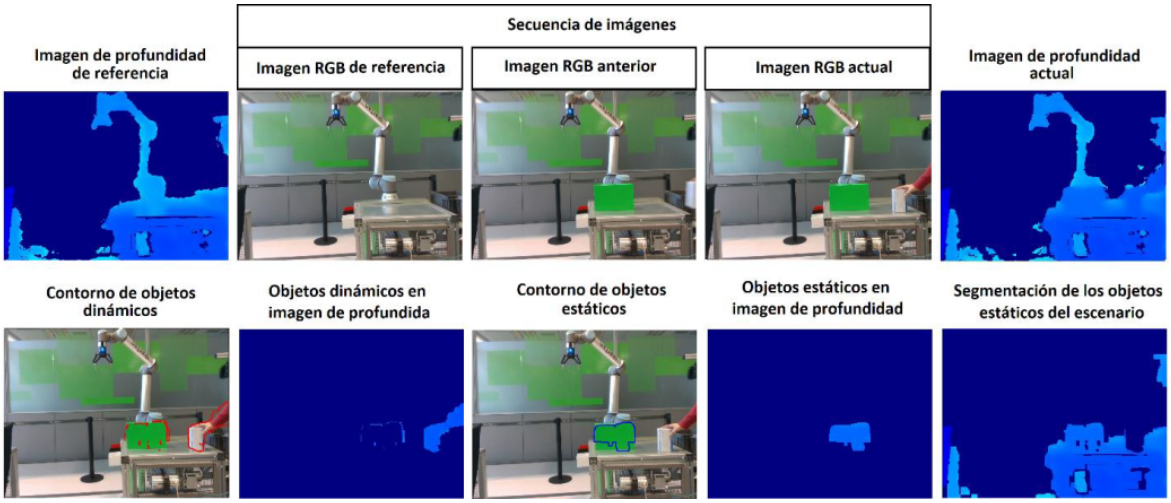

Figure 7.20: Final scene segmentation. The segmentation can be divided into static background, work pieces and dynamic obstacles to avoid.

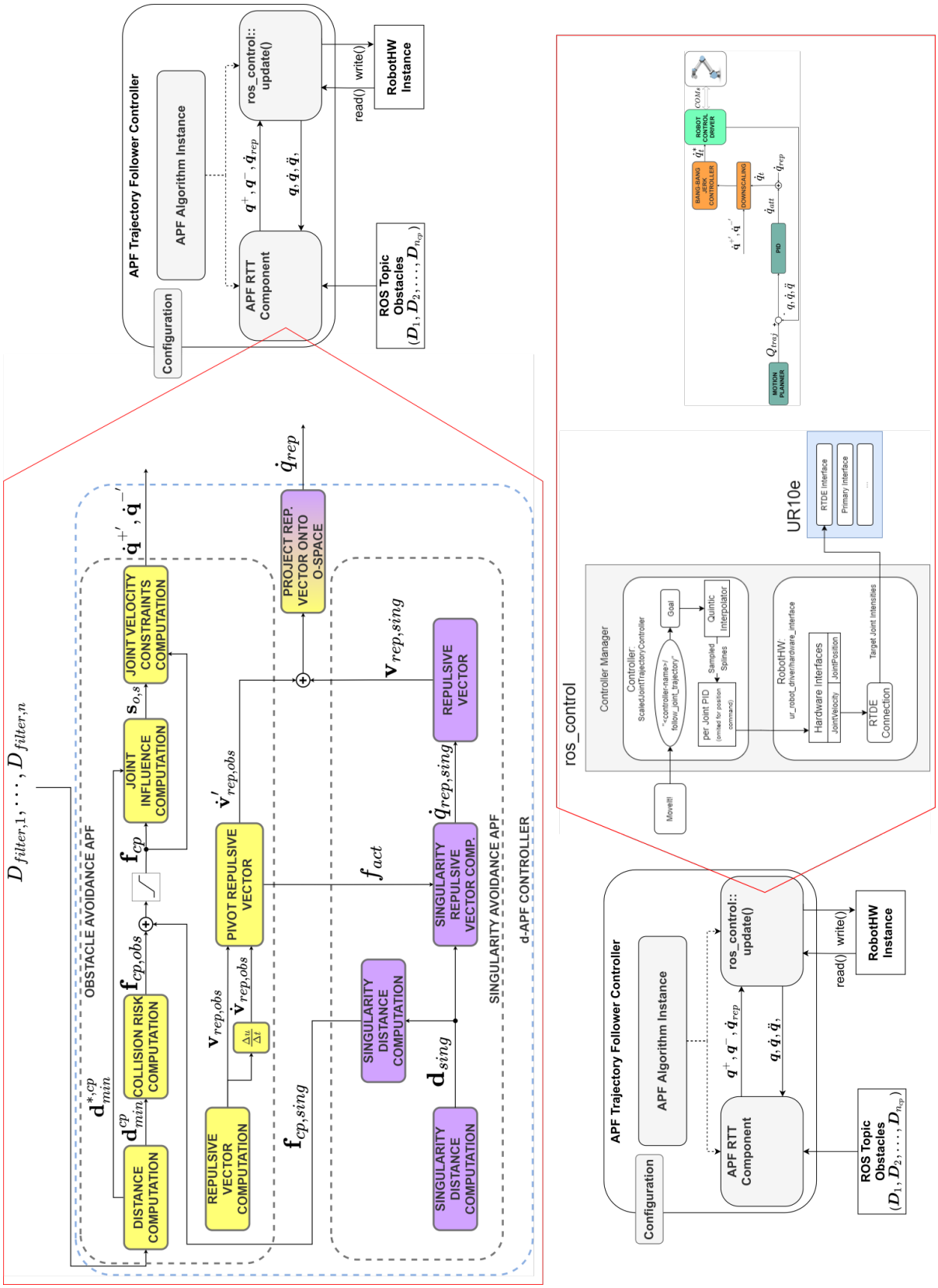

Figure 7.21: ROS Control encapsulation on Orocos for the UR10e. At the bottom and the top it has been included the corresponding components of the control loops implemented for the ros_control::update() and the RTT APF components, respectively. Note that this representation is for the d-APF controller, being a similar one for the DLS-APF controller without the control boxes referred to the singularity avoidance (check Figure 5.11 and the considerations from Chapter 5).

7.3 SUPSI robot particularization

The SUPSI robot particular implementation is based on the previous controller developed for the UR10e. Since it is a self-developed robot, the first task to do was integrate the robot into the ROS Control combined with Orocos architecture. Therefore, the implemented interface is a mirror of the one modified for the UR10e. Despite the similarities between both robots, it is worth mentioning that the structure of the SUPSI robot is distributed differently for the resting position than in the UR10e case. Because of that reason, the kinematic model of the SUPSI manipulator differs from the UR10e one. Therefore, it is considered also interesting to analyze this other structure to test and verify if the results and implementation made for the UR10e robot are also applycable to other non-spherical wrist manipulators. In the current subsections only the specific modifications with respect to the final version of the UR10e drivers are explained.

Similar to the UR10e case, two models have to be presented for the SUPSI robot: one for applying to the DLS-APF controller, and another model for the d-APF controller. The traditional kinematic model implemented in the robot by the SUPSI correspond to the one displayed in Figure 7.22 and Table 7.3. In this case, since the kinematic model and the structure is very similar to the UR10e robot, the table with all the parameters has been avoided as it is considered enought expalined with the anotations of the figure.

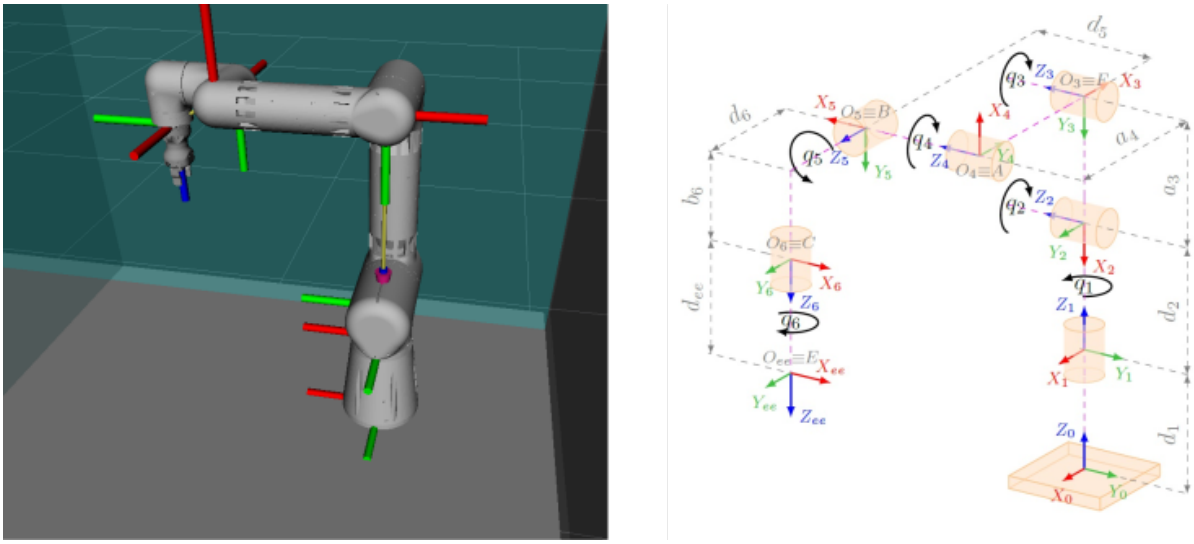

Figure 7.22: Kinematic model of the SUPSI vendors package displayed in its resting position. At the right-hand side the RViz simulation, and at the left-hand side the kinematic model.

To apply the proposed decoupled kinematics for non-spherical wrist cobot, the 4-th and the 6-th reference frames should be moved to the decoupling point (B). This modification leads to a similar kinematics to the one shown in Figure 7.23. This configuration corresponds to the parameter table of the decoupled kinematics gathered in Table 7.4[15].

15: Being $r = \sqrt{d_6^2 + (b_6 + d_{ee})^2}$, and $\beta = \left(\frac{(b_6+d_{ee})}{d_6}\right)$

Table 7.3: Parameters for the kinematic model of the SUPSI robot ROS package.

X [m]	Y [m]	Z [m]	θ_x [rad]	θ_y [rad]	θ_z [rad]
0	0	d_1	0	0	q_1
0	0	d_2	$\pi/2$	$\pi/2$	q_2
$-a_3$	0	0	0	0	$q_3 - \pi/2$
$-a_4$	0	0	0	0	$q_4 - \pi/2$
0	0	d_5	0	$-\pi/2$	$q_5 + \pi/2$
0	b_6	d_6	$-\pi/2$	π	q_6

[a] This table does not regular Denavit Hartenberg (DH) convention as normally do.
[b] The X, Y, and Z coordinates represent the Cartesian translation between reference systems.
[c] The θ_x, θ_y, and θ_z rotations reprents the rotations along the corresponding subindexed axes.

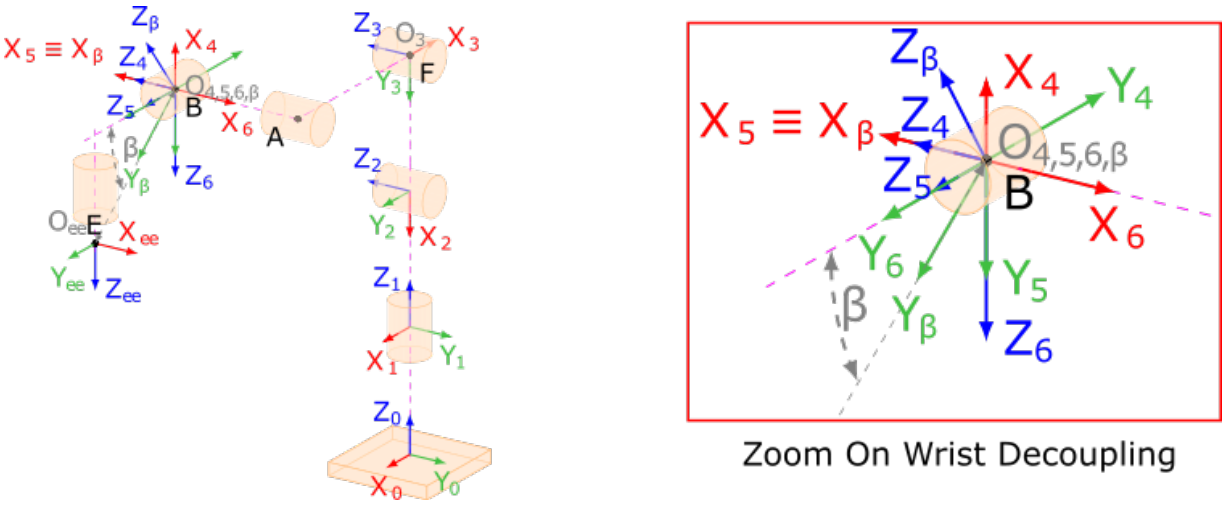

Figure 7.23: Decoupled kinematic model of the SUPSI robot in the resting position with a zoom on the wrist decoupling.

7.3.1 Kinematic behavioral study

As appreciated in the preamble of this section (Section 7.3), the robot structure of the SUPSI robot is practically identical to the one the UR10e has. In fact, they share the same non-spherical wrist structure. However, they are different between the 2nd and 3rd reference frames. In the case of the SUPSI robot, there is a $\pi/2$ additional rotation between these two joints. Therefore, the kinematic model will behave similar with slight differences due to the $\pi/2$ displacement between joints 2 and 3. Therefore, minimal anotations are given during the analysis, highlighting only the differences. In this subsection, first the FK and IK are studied, leading to a posterior singularity study and characterization due to the Jacobian analysis.

7.3.1.1 Forward and Inverse Kinematics

7.3.1.1.1 Forward kinematics For the case of the forward kinematics of the robot, the results obtained to compute the homogeneous transformation matrix of the decoupled model ($\mathbf{T}_6^0$) shows the following position and rotational equations (see from equation 7.16[16] to 7.18).

16: Note that $z_B^{0,*} = z_B^0 - d_1 - d_2$.

X [m]	Y [m]	Z [m]	θ_x [rad]	θ_y [rad]	θ_z [rad]
0	0	d_1	0	0	q_1
0	0	d_2	$\pi/2$	$\pi/2$	q_2
$-a_3$	0	0	0	0	$q_3 - \pi/2$
$-a_4$	0	d_5	0	0	$q_4 - \pi/2$
0	0	0	0	$-\pi/2$	$q_5 + pi/2$
0	0	0	$-\pi/2$	π	q_6
Parameters of the O_β reference frame					
X [m]	Y [m]	Z [m]	θ_x [rad]	θ_y [rad]	θ_z [rad]
0	r	$\pi/2 - \beta$	0	0	$\pi/2$

[a] This table does not follow regular Denavit Hartenberg (DH) convention as normally do.
[b] The X, Y, and Z coordinates represent the Cartesian translation between reference systems.
[c] The θ_x, θ_y, and θ_z rotations reprents the rotations along the corresponding subindexed axes.

Table 7.4: Parameters for the proposed decoupled FK model for the SUPSI robot ROS package

$$\mathbf{p}_B^0 = \begin{bmatrix} x_B^0 \\ y_B^0 \\ z_B^{0,*} \end{bmatrix} = \begin{bmatrix} -d_5 s_1 + c_1(-a_3 s_2 + a_4 c_{2,3}) \\ -d_5 c_1 - s_1(a_3 s_2 - a_4 c_{2,3}) \\ a_3 c_2 + a_4 s_{2,3} \end{bmatrix} \tag{7.16}$$

$$\mathbf{R}_3^0 = \begin{bmatrix} -c_{2,3}c_1 & s_{2,3}c_1 & s_1 \\ -c_{2,3}s_1 & s_{2,3}s_1 & -c_1 \\ -s2,3 & -c_{2,3} & 0 \end{bmatrix} \tag{7.17}$$

$$\mathbf{R}_6^3 = \begin{bmatrix} -c_4 s_6 + c_6 s_4 s_5 & c_5 s_4 & -c_4 c_6 - s_4 s_5 s_6 \\ -s_4 s_6 - c_4 s_5 c_6 & -c_4 c_5 & -s_4 c_6 + c_4 s_5 s_6 \\ -c_5 c_6 & s_5 & c_5 s_6 \end{bmatrix} \tag{7.18}$$

In addition, for the velocity, the calculated Jacobian matrix also presents the $\mathbf{0}_{3\times3}$ matrix that simplifies the study of the inverse kinematics.

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} (\mathbf{J}_{11})_{3\times3} & \mathbf{0}_{3\times3} \\ (\mathbf{J}_{21})_{3\times3} & (\mathbf{J}_{22})_{3\times3} \end{bmatrix} \tag{7.19}$$

7.3.1.1.2 Inverse kinematics Since the positioning and orienting equations are similar to the ones seen for the UR10e case, the closed set of solutions for the inverse kinematics of the SUPSI robot do not differ much according to the previous ones (equations 7.7, 7.8, 7.9 and 7.10). Moreover, as the IK of the positioning and orientation does not influence directly the kinematic singularities that has to be parameterized to implement the d-APF Controller, so it is not presented in this document.

7.3.1.2 Singularity Study and characterization

Regarding the IK for the velocity of the robot, the singular configurations correspond to the configurations where the determinant of the Jacobian matrix turns 0 ($|\mathbf{J}| = 0$). Once again, the solutions for the singularity analysis can be split into the solutions for the each of both cases $|(\mathbf{J}_{11})_{3x3}| = f_1(q_1, q_2, q_3) = 0$ or $|(\mathbf{J}_{22})_{3x3}| = f_2(q_4, q_5, q_6) = 0$. The solutions for each situation are presented in equation 7.20[17] and 7.21, respectively.

17: In this equation, the mathematical functions utilized to decompose $|J_{11}|$ correspond to:

$$f(q_3) = -\cos(q_3)$$

$$h(q_2, q_3) = a_3 a_4 (a_4 \cos(q_2 + q_3) - a_3 \sin(q_2)).$$

Moreover, these equations have been also represented to allow a more clear understanding of the zeros of both cases in Figures 7.24 and 7.25.

$$|\mathbf{J}_{11}| = f(q_3) \cdot h(q_2, q_3) = -\cos(q_3) \cdot (a_3 a_4 (a_4 \cos(q_2 + q_3) - a_3 \sin(q_2))) \tag{7.20}$$

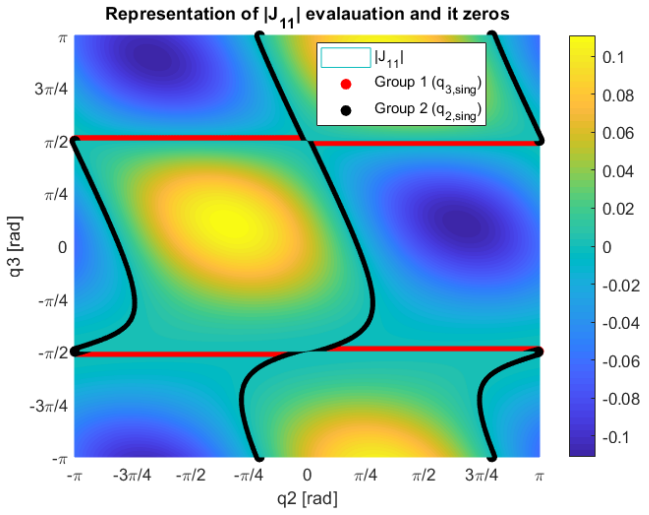

Figure 7.24: Graphical representation of the found zeros for the case of $|(\mathbf{J}_{11}(q_1, q_2, q_3))_{3x3}| = 0$ in the SUPSI robot decoupled kinematics.

Figure 7.24 displays a similar singularity distribution for the arm singularities group into two different solutions types: the red coloured ones and the black coloured ones. The first ones are bounded to $f(q_3)$ while the black coloured ones are linked to $h(q_2, q_3)$. Nevertheless, due to the structural change in the distribution of the joints of the robot in its resting position, the $\pi/2$ displacement relative to this modification is also present in the closed set of solutions for the singular configurations. This matching between models (the UR10e and the SUPSI robot one) is positive because it can be seen as a check for the good and stable behavior of the proposed decoupled kinematic model for non-spherical wrist robots.

$$|\mathbf{J}_{22}| = g(q_5) = -\cos(q_5) \tag{7.21}$$

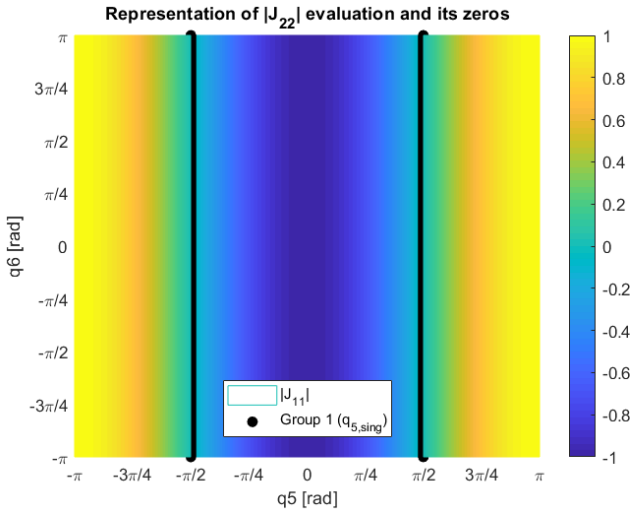

Figure 7.25: Graphical representation of the found zeros for the case of $|(\mathbf{J}_{22}(q_4, q_5, q_6))_{3x3}| = 0$ in the SUPSI robot decoupled kinematics.

With the solutions appreciated above the singularity characterization of the joints can be described in a general formulation that gathers all the cases as shown in equation 7.22. This parameterization of the singular configurations ends up in the equation 7.23 where the calculation of the vector of distances to singularities is displayed for the SUPSI robot.

$$|\mathbf{J}| = 0 \iff \begin{cases} h(q_2, q_3) = a_3 a_4 (a_4 \cos(q_2 + q_3) - a_3 \sin(q_2)) \\ f(q_3) = -\cos(q_3) \\ g(q_5) = -\cos(q_5) \end{cases} \tag{7.22}$$

$$\mathbf{d}_{sing} = \begin{bmatrix} 0 & f(|\mathbf{J}_{11}|)_{q_2} & f(|\mathbf{J}_{11}|)_{q_3} & 0 & f(|\mathbf{J}_{22}|)_{q_5} & 0 \end{bmatrix}^T \tag{7.23}$$

7.3.2 Hardware Interface and RTT ROS controller

The implementation of the Hardware Interface is also a mirror of the ROS Control implementation for the UR10e robot. Figure 7.26 displays the controller class diagram for the implemented Hardware Interface where, once again, it can be appreciated that the position and velocity joint interfaces inherits from a self-developed scaled_joint_trajectory interface. Unlike to the UR10e which relies on the RTDE library to communicate with the robot, for the SUPSI robot a library to encapsulate the real time TCP/IP communication has been developed, the so called supsi_hardware_interface::SupsiHardwareInterface.

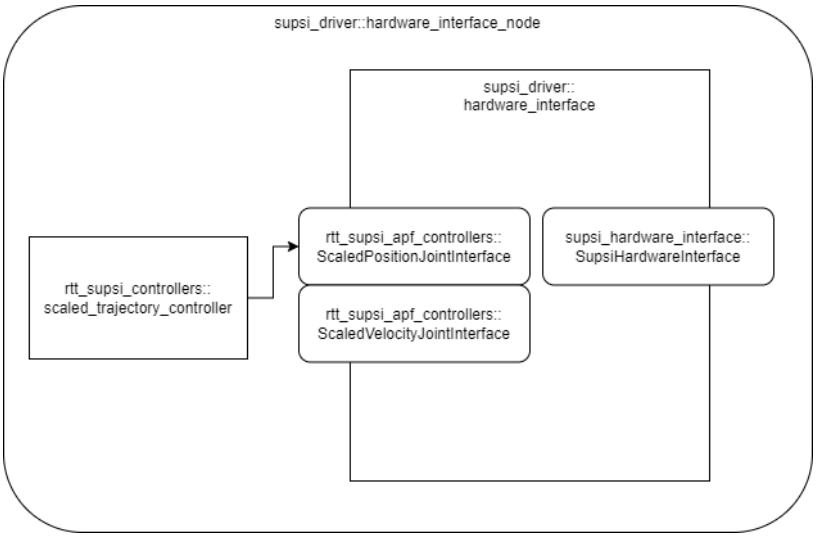

Figure 7.26: ROS Control class diagram for the SUPSI robot developed hardware interface.

One of the advantages of employing the SUPSI robot for implementing the controller is that since it is a self-developed, there is a lot of versatility for implementing the communication capabilities. The SUPSI robot comes with a KEBA controller where the SUPSI engineers and researchers have implemented the low level controllers of the robot, the safety measures, and the starting procedure and sequence, among other capabilities of the deliberative robot. Therefore, to implement an specific program to allow the communication with the robot, two additional automation tasks to send and received the desired and current joint states, respectively, was implemented in the KEBA controller.

This two programs were required from the KEBA controller side to allow sending the current position and velocity states to the ROS Controller, and receiving the target position and velocity to be applied as references in the low level controllers of the robot. In addition to both task, a circular buffer was also implemented to avoid loosing any received command. Moreover, several flags where also utilized to manage the movement in a new jogging mode that executes the commands received from the ROS side. Fact that also requires additional flags to coordinate this new working mode for the robot with its regular behavior to guarantee the trigger of the essential safety measurements already implemented. By doing so, the already integrated safety of the controller would never be violated guaranteeing the operator safety since the ROS Controller never bypasses the robot control box.

In the ROS side, for the implementation of the supsi_hardware_interface::SupsiHardwareInterface, a TCP/IP library for communication in Ubuntu was developed. This communication library was then encapsulated inside a self-developed Hardware Interface. On top of that, the RTT ROS Controller was implemented by inheriting from an adapted scaled_trajectory_controller interface for the SUPSI robot and the hardware interface bridge of the proposed control architecture. This considerations allows the implementation of the rtt_supsi_apf_controller for both controllers: the DLS-APF and the d-APF ones. The software package dependencies coincides with the one showed for the UR10e due to their mirror implementation.

Lastly, to guarantee the proper communication between the ROS and the robot control box sides a communication structure was defined in both sides to allow the understanding of the different data types. The SUPSI robot is a 6 DoF manipulator, so the minimum required information for the ROS controller are the identification number of the message, and the position and velocity of each joint. To ease the understanding of the compact structure of data sent from and to the robot control box, the Table 7.5 is included below.

Table 7.5: Required struct to communicate with the KEBA Controller.

Message field	Data size	ROS format	KEBA format
id	2 Bytes	int16	UINT
pos[6]	24 Bytes (6×4 Bytes)	float	DWORD
vel[6]	24 Bytes (6×4 Bytes)	float	DWORD

Taking into account this modifications and with the implementation displayed in Subsection 7.3.1, the Hardware Interface can be considered fully implemented for the SUPSI robot.

7.3.3 Vision algorithm

With respect to the vision algorithm, there are just a filter to detect the obstacle from the scene for the simulated scenario. In this case, the vision filter utilized corresponds to the exact same HSV green filter used for the UR10e simulated scenario (see Figure 7.27). In this way, a green sphere emulates the random and static behavior of the human operator depending on the type of simulation that is running. Once the green obstacle is filtered, the remaining depth map is published so the controller can compute the distance between the obstacle and the

different control points of the body of the robot. To emulate a human behavior is enough with this method, however, for a real scenario the filter should employ the proposed algorithm for the UR10e robot or any pretrained Neural Network based algorithm instead. In this manner, the vision filter is considered a black box that suits the objective of giving a segmented environment to the ROS Controller.

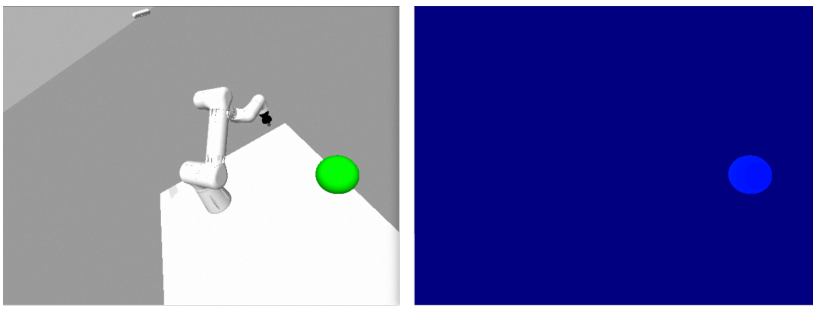

Figure 7.27: Green filter for the SUPSI robot vision algorithm in simulation. (a) Colour image of the Gazebo environment. (b) Filtered depth map after applying the green colour filter.

7.4 Conclusions to Chapter 7

In this chapter all the considerations regarding the physical and virtual implementation of the controllers have been presented. Firstly, the real time encapsulation of ROS Control into Orocos is addressed. This development leads to the implementation of a generic RTT Controller Manager to enable the ROS Control loop cycle and a generic hardware interface bridge to share data between Orocos components and ROS Controllers. In addition, the data structure to allow the coordination of the hybrid ROS-Orocos controllers with the remaining elements of the ROS network is exposed.

On top of the common software architecture, each DLS-APF and d-APF controllers (RTT ROS controllers, in general) have been implemented for each robot (the UR10e and the SUPSI robot). In the case of the SUPSI robot, additional considerations such as a self-defined message for the communications or a program for the KEBA controllers were required, since it is a self-developed robot. Moreover, the implementation of the d-APF controller requires a full kinematic behavioral study, also included in this chapter. From this study, similarities between both structures has been found highlighting positive aspects in the implementation of the decoupled kinematic model proposed for non-spherical wrist cobot. This analysis ends up in a suitable closed set of solutions for the inverse kinematics and the singular configurations of each robot.

Lastly, it is worth mentioning that the whole implementation has been made keeping the safety of the application since the programmed intrinsic safety measures can never be violated by the external RTT ROS controller.

Tests, Results and Conclusion

Performance Tests and Results 8

Until this point, all the theoretical developments and the implementation have been regarded from Chapter 4 to 7. Throughout all the presented developments, the conclusions of each chapter highlight a set of advantages that, a priori, were fulfilled by both main contribution: the novel decoupled kinematic model for non-spherical wrist cobot and the d-APF controller. To test the validity of the aforementioned statements about the advantages, Chapter 8 analyze the behavior of the main contributions of this work.

Due to the presented research method in Chapter 3 this chapter offers a combination of simulated test with real world tests. On the one hand, the aspects more related to the kinematics modeling or the first performance test are executed in simulation. On the contrary, the real time suitability of the proposed controller as well as the study of the trajectory tracking errors, among other results, are studied in a real scenario. In general, both robots, the UR10e robot and the SUPSI one, have been employed to test the performance of the controller in simulation, while only the UR10e robot has been utilized to test the performance on a real collaborative work cell.

Each of the proposed tests is described in its corresponding section, where then the collected data from the experimentation phases as well as a quick first analysis about the results are exposed. The first observed results correspond to an analysis of the singularity behavior between the proposed novel model and other typically employed models of the literature. Then, these analyzed kinematic models are implemented into its respective controller (DLS-APF and d-APF controllers), allowing a performance test in simulation for each of the RTT ROS controllers and robot. Subsequently, the suitability of the real time performance of the implemented controller is measured for the UR10e and both controller implemented (DLS-APF and d-APF controllers). The last results included are about the efficiency and error while tracking trajectories with the proposed controller based on several scenarios of static and dynamic elements, only tested on the UR10e robot. Finally, to conclude Chapter 8, a brief conclusion is presented summing up all the most relevant results of the chapter.

8.1 Kinematic behavior implications

In this section, the suitability of the kinematic behavior of the decoupled kinematic model for non-spherical wrist cobots is checked. This section uses the UR10e because since it is a commercial robot, there are more kinematic models examples to test against in the literature. More specifically, it is employed the ROS based from the Universal Robots ROS drivers [127] and the Denavit-Hartenberg (DH) from [111] as reference models. The aim of the test is to check whether it is expected the same

kinematic behavior or not, in a theoretical manner. Thus, the tests is a simulation based test.

To check the kinematic behavior equivalence between the proposed model (for this section, it is generically called the decoupled model) and the other two from the literature (called the coupled ones). This thesis has implemented a proof based on the matrix equivalency [118][1]. This method has been selected because matrix equivalency is similar to the SVD and allows to compare directly whether two matrices shares singular values matrices or not. The aim of the proposed test is to check whether the decoupled model shares the null space with the coupled models of the literature or, in other words, to validate if their singular configurations coincide. This proof is selected since two matrices are equivalent if they share the null space, what means in this particular use case that their singular configurations are the same [142]. Therefore, their decomposition into singular values should be the same [118]. However, if they are not equivalent, their kinematic behavior will be different, presenting different unstable configurations for the control.

The first part of the comparison relies in two generic Jacobian matrices for the decoupled and coupled models, $\mathbf{J}_D$ and $\mathbf{J}_C$, respectively. Both expressions can be appreciated in equation 8.1.

$$\mathbf{J}_D = \begin{bmatrix} (\mathbf{J}_{D,11})_{3\times3} & (\mathbf{0})_{3\times3} \\ (\mathbf{J}_{D,21})_{3\times3} & (\mathbf{J}_{D,22})_{3\times3} \end{bmatrix} \tag{8.1a}$$

$$\mathbf{J}_C = \begin{bmatrix} (\mathbf{J}_{C,11})_{3\times3} & (\mathbf{J}_{C,12})_{3\times3} \\ (\mathbf{J}_{C,21})_{3\times3} & (\mathbf{J}_{C,22})_{3\times3} \end{bmatrix} \tag{8.1b}$$

Computing the Jacobian determinant as square block matrices[2], the expression from below (equation 8.2) is obtained. Note that, since the utilized Jacobian is the geometrical one, this result is truth only according to the following assumptions: $\mathbf{J}_{D,22} = \mathbf{J}_{C,22} = \mathbf{J}_{22}$[3] , and $\mathbf{J}_{D,21} = \mathbf{J}_{C,21} = \mathbf{J}_{21}$[4].

$$|\mathbf{J}_D| = |\mathbf{J}_{D,11}||\mathbf{J}_{22}| \tag{8.2a}$$

$$|\mathbf{J}_C| = |\mathbf{J}_{C,11}\mathbf{J}_{22} - \mathbf{J}_{C,12}\mathbf{J}_{21}| \tag{8.2b}$$

This first theoretical analysis on the Jacobians determinant of the coupled and decoupled models highlights that they should not be equivalent. It might occur that with a proper selection of the reference frames distribution, the following conditions turns true: $(\mathbf{J}_{C,12})(\mathbf{J}_{21}) = \mathbf{0}$ and $|\mathbf{J}_{D,11}| = |\mathbf{J}_{C,11}|$. In case this scenario does not happen, the general consideration is that $|\mathbf{J}_D| \neq |\mathbf{J}_C|$[5].

This theoretical result has been checked with a simulation for each of the aforementioned model in a statistical way. For each characterized singularity[6] a set of 10000 random valid configurations have been tested. The aim is to check whether or not these three kinematic models shares the same null space. Therefore, Tables 8.1, 8.2 and 8.3 gathers the evaluation of the singularity for each of the singularity parameterizations, which corresponds to $f(q_3)$, $g(q_5)$, and $h(q_2, q_3)$, respectively.

1: In linear algebra, two rectangular m-by-n matrices $\mathbf{A}$ and $\mathbf{B}$ are called equivalent if

$$\mathbf{B} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{P}$$

where $\mathbf{Q}$ is an m-by-m invertible matrix and $\mathbf{P}$ is an n-by-n invertible matrix.
As shown in Figure 8.1, equivalent matrices represent the same linear transformation $V \rightarrow W$ under two different choices of a pair of bases of V and W, with P and Q being the change of basis matrices in V and M, respectively. Equivalent matrices is an equivalence relation of the space of rectangular matrices characterized by the following properties:

1. Equivalent matrices have the same rank.
2. Equivalent matrices are a linear combination of their rows and columns. Therefore, they might have or might not have the same determinants.

2: **Assumption 1:** A square block submatrix determinant can be computed as stated in the following equation:

$$\left|\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}\right| = |\mathbf{AD} - \mathbf{BC}|$$

3: **Assumption 2:** Let the jacobian matrix ($\mathbf{J}$) be an 6-by-6 block matrix generically defined as:

$$\mathbf{J} = \begin{bmatrix} (\mathbf{J}_{11})_{3\times3} & (\mathbf{J}_{12})_{3\times3} \\ (\mathbf{J}_{21})_{3\times3} & (\mathbf{J}_{22})_{3\times3} \end{bmatrix}$$

In general, for a jacobian matrix, it is true the following statement:

$$(\mathbf{J}_{21})_{3\times3} = \begin{bmatrix} \mathbf{z}_0 & \cdots & \mathbf{z}_{m-3} \end{bmatrix} \neq (\mathbf{J}_{12})_{3\times3}$$

where:

$$(\mathbf{J}_{12}) = \begin{bmatrix} \mathbf{z}_{n-2} \times \Delta_{p_{n-2}} & \cdots & \mathbf{z}_n \times \Delta_{p_n} \end{bmatrix}$$

$$\Delta_{p_i} = (\mathbf{p}_e - \mathbf{p}_i)$$

4: **Assumption 3:** Let $\mathbf{J}_D$ and $\mathbf{J}_C$ be two geometric jacobian matrices of different kinematic models (different reference system distribution). If and only if the geometric Jacobian is employed to describe the kinematic behavior of the robot and both models include the same rotations and share the orientation of their reference frames, then the following relations are also true:

$$(\mathbf{J}_{D,21})_{3\times3} = (\mathbf{J}_{C,21})_{3\times3} \text{ and}$$

$$(\mathbf{J}_{D,22})_{3\times3} = (\mathbf{J}_{C,22})_{3\times3}$$

This happens because the geometric jacobian is related to the angular velocity around X, Y, and Z axis. Thus, if two kinematic models present the same joint turns, the angular velocity should be equal (only held true if the world frame is same oriented).

Each of the presented Tables executes a singularity matching, a singularity checking of boundary singularity, and an statistical study distribution of the computed data for each tested configuration. The goal of the singularity matching test is to check the percentage of configurations that can be considered singular for each use case. The selected criterion to consider a configuration in a singularity has been chosen following the guidance of [105][7]. The boundary singularity test is to check whether the robot is in a boundary singularity or not, to understand the nature of the singularity. And lastly, the statistical analysis of the values obtained from each Jacobian determinant pretends to show the variability in the computed results.

Singularity Matching			
	Dec [%]	**ROS [%]**	**DH [%]**
Dec	100.00	100.00	100.00
ROS	–	100.00	100.00
DH	–	–	100.00
All models match [%]		**Boundary Sing. [%]**	
100.00		0.00	
Statistics			
	Mean (J)	**Std (J)**	**Max (J)**	**Min (J)**
Dec	$3.33 \cdot 10^{-20}$	$1.30 \cdot 10^{-17}$	$1.35 \cdot 10^{-16}$	$-1.05 \cdot 10^{-16}$
ROS	$7.94 \cdot 10^{-20}$	$1.66 \cdot 10^{-17}$	$1.49 \cdot 10^{-16}$	$-1.94 \cdot 10^{-16}$
DH	$2.61 \cdot 10^{-20}$	$1.31 \cdot 10^{-17}$	$1.15 \cdot 10^{-16}$	$-1.05 \cdot 10^{-16}$

Table 8.1: Statistics and singularity matching for the case $f(q_3)$.

Singularity Matching			
	Dec [%]	**ROS [%]**	**DH [%]**
Dec	100.00	100.00	100.00
ROS	–	100.00	100.00
DH	–	–	100.00
All models match [%]		**Boundary Sing. [%]**	
100.00		0.00	
Statistics			
	Mean (J)	**Std (J)**	**Max (J)**	**Min (J)**
Dec	$-4.08 \cdot 10^{-21}$	$6.43 \cdot 10^{-18}$	$4.66 \cdot 10^{-18}$	$-5.30 \cdot 10^{-17}$
ROS	$-1.40 \cdot 10^{-20}$	$4.89 \cdot 10^{-18}$	$4.90 \cdot 10^{-17}$	$-4.41 \cdot 10^{-17}$
DH	$-3.96 \cdot 10^{-20}$	$4.84 \cdot 10^{-18}$	$3.99 \cdot 10^{-17}$	$-4.72 \cdot 10^{-17}$

Table 8.2: Statistics and singularity matching for the case $g(q_5)$.

Singularity Matching			
	Dec [%]	**ROS [%]**	**DH [%]**
Dec	99.84	2.71	2.67
ROS	–	2.71	2.62
DH	–	–	2.67
Three models matching			
2.62			
Boundary Sing. [%]		**Shoulder Sing. [%]**	
0.00		0.00	
Statistics			
	Mean (J)	**Std (J)**	**Max (J)**	**Min (J)**
Dec	$-5.08 \cdot 10^{-07}$	$3.51 \cdot 10^{-05}$	$1.01 \cdot 10^{-04}$	$-1.01 \cdot 10^{-04}$
ROS	$-2.58 \cdot 10^{-04}$	0.0149	0.0418	−0.0419
DH	$-2.57 \cdot 10^{-04}$	0.0149	0.0420	−0.0418

Table 8.3: Statistics and singularity matching for the case $h(q_2, q3)$.

The first fact appreciated is that for the first two cases ($f(q_3)$ and $f(q_5)$) there is full matching of the testing singular configurations. On the contrary, for the case of singularities due to q_2 ($h(q_2, q_3)$) there is a mismatching between the three models. This behavior is appreciated in both sections of the tables, the Singularity Matching and the Statistics.

5: Note that this demonstration is also applicable for a 7 DoF by applying **Assumption 4** instead: A rectangular matrix which meets the condition of having an invertible block submatrix $\mathbf{A}$ (and similarly $\mathbf{D}$ is invertible), its determinant can be computed in the following way:

$$\left|\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}\right| = |\mathbf{A}| \cdot |\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}|$$

.

6: Kind reminder of equation 7.15:

$$|\mathbf{J}| = 0 \iff \begin{cases} h(q_2, q_3) = 0 \\ f(q_3) = 0 \\ g(q_5) = 0 \end{cases}$$

7: In this paper it is considered as a singular configuration each configuration that meets:

$$\Psi = \{\mathbf{q} \in \mathbb{R}^n : |\mathbf{J}(\mathbf{q})| \leq s_0\},$$

where s_0 is a close to zero threshold defined by the user.

Moreover it is also remarkable that none of them corresponds to a boundary singularity nor a shoulder singulary. So in a first glimpse, the proposed decoupled kinematic model for non-spherical wrist cobot behaves correctly and suit the desired purpose.

Focusing the attention again on the first two tables, these singularities are already known in the literature as shown in [109]. Therefore, it can be assumed that the proposed model adopts the singularities due to the structural distribution of the robot joints. It is worth noting that thanks to this matching, the decoupled kinematic model is considered well developed since the singularities due to constructive reason will remain in the kinematic model independently how the reference systems are distributed.

On the contrary, for the case of $h(q_2, q_3)$ this matching is not present in the Singularity Matching section of Table 8.3 nor the Statistics one. From the Statistics section it is appreciated that there is around a 3-rd order of magnitude difference according to the literature models. This relevant mismatch in the models is also supported by the fact that even between the DH and the ROS models, the computed singular configurations differs from one another. In this manner, this behavior highlights the importance of a good selection of reference systems while modelling the robot. A more favorable or simpler model can lead to more efficient computation, or even to a closed set of solutions for the singular configurations as seen in this work. Due to this observed strong dependency on the reference frame distribution, this thesis proposes a new type of singularity belonging to the internal singularities of the robot[8], the **model singularities**. These singular configurations are the ones directly linked to the reference frames distribution (i.e. $h(q_2, q_3)$), and are generated due to the mathematical coupling of trigonometric relationships.

8: Terminology from [109].

In addition to the results of the singularity analysis, the difference in the kinematic behavior can also be appreciated since equation 8.1. In this equation the coupled models present a full Jacobian structure while the decoupled models relies on a simplified zero block matrix structure ($\mathbf{0}_{3\times3}$). This fact is relevant because it means that at least the task space control of the robot looses the dependency of the fourth and fifth (q_4 and q_5) joints. Therefore, the wrist no longer affects the positioning of the arm, influencing exclusively the orientation and angular velocities of the decoupling point. Becoming, in this way, a trade off of the change in the control model of the TCP for the decoupling point instead.

To sum up, the proposed kinematic model modifies the kinematic behavior of the robot by simplifying it. This simplification can be seen in both: the closed set of solutions obtained for the robot (see Chapter 7), and the comparison of singular configurations exposed above. The results show that the decoupled kinematic model for non-spherical wrist manipulators bounds the singularities to a closed joint dependent characterization. In this way, the singular configurations are reduced to a limited set of configurations, enabling the chance of predicting them beforehand which, in turn, allows the utilization of advanced reactive control algorithms as the d-APF controller proposed, as opposed to traditional models.

8.2 RTT ROS Controllers performance tests in simulation

Once the suitability and behavior of the proposed kinematic model have been proven adequate, the first tests on the controller can be executed. As hinted in the preamble of this chapter, the very first proofs on the behavior of the controller were executed in simulation first, to validate the suitability of its implementation without risking any equipment integrity. In this first test, two parameters are tested: the timing performance of the d-APF controller against the DLS-APF one, and the timing performance and suitability of implantation of different types of collaborative manufacturing processes.

For such an aim, a collaborative battery cells disassembly task has been simulated for both robots in an environment as the one shown in Figure 8.2. The different processes to be executed in this collaborative battery disassembly process are described in the flow chart from Figure 8.3. The executed simulated tests consist of two different tests of 10 battery pack disassembly processes, in total 80 pick and place operations (sorting the related cells). From among this operations, 40 cycles are executed in the presence of an obstacle simulating the operator (static and dynamic) while the remaining 40 cycles are performed in the absence of obstacles. The presence of operator in both states, static and dynamic, is simulated with a green ball in Gazebo filtered with the HSV green filter as described in Subsection 7.2.3. An example of the simulated tests during execution for the SUPSI robot is shown in Figure 8.4 below.

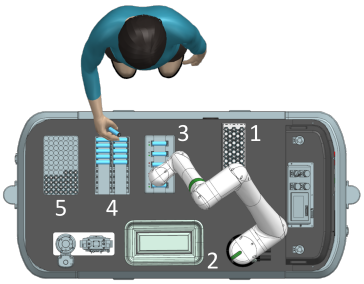

Figure 8.2: Cobot workcell layout with the numbered core process steps.

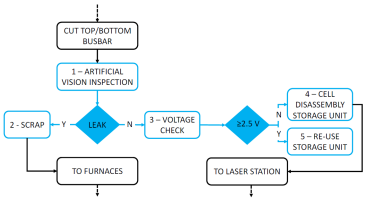

Figure 8.3: Battery cell sorting after battery pack disassembly process.

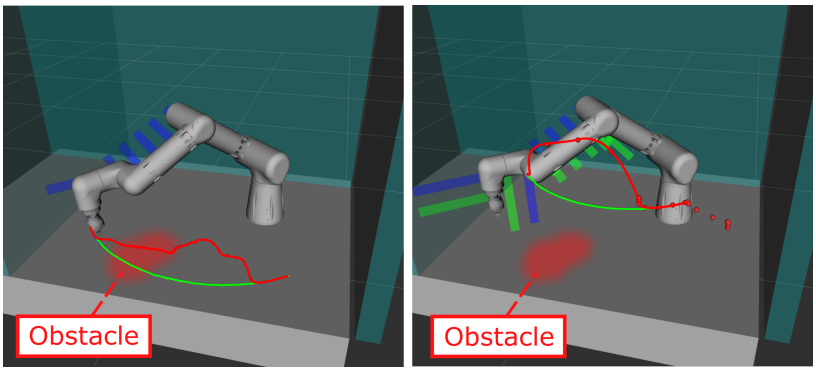

Figure 8.4: Collision avoidance during RTT ROS controllers simulated test with the SUPSI robot: Left: DLS and APFs controller case. Right: d-APF controller case where at each control point the relative obstacle (blue bars) and singularity risk (green bars) values are displayed.

In this manner, the experimental process results displayed in Table 8.4 measures: the mean execution time of the ros_control::update() component (controller), the mean timing performance of the APF RTT component (component)[9], the mean cycle time for each pick and place task with obstacle (task obs.), and in the absence of obstacle (task no obs.). Note that since the laboratory environment of the SUPSI robot is different to the UR10e one, the collaborative disassembly cell could not be reliably reproduced. For such reason, there are only displayed results of the Component and the Controller for the UR10e robot.

All these results presented above are summed up in Table 8.4[10] for the DLS-APF controller as Benchmark results, and the d-APF controller results as d-APF. In general terms, the controller components present a reduction on the requiered computational load, while the cycle times are reduced. So the proposed controller is behaving as expected. In addition,

9: Both terminology are obtained from Figure 7.21.

10: Further detail of the measured values to compute the results from Table 8.4 are displayed at the end of this section

the obtained results display similarities between the implementation of the controllers on each robot.

Table 8.4: Results summary of the RTT ROS Controllers simulated performance tests. The Component and Controller rows of the table are measured in milliseconds (ms) since they represent control cycle loop execution times. Moreover, the Task with and without obstacles rows represent the timing of a full pick and place cycle, therefore it is measured in seconds (s).

SUPSI robot study case						
	Static obstacle tests			**Dynamic obstacle tests**		
Parameter	**Benchmark**	**d-APF**	**Reduction**	**Benchmark**	**d-APF**	**Reduction**
Component	2.137ms	1.804ms	16%	2.183ms	1.804ms	17%
Controller	1.6562ms	1.794ms	−8%	1.555ms	1.626ms	−5%
Task no obs.	95.506s	54.036s	43%	93.998s	56.339s	40%
Task obs.	113.31s	70.287s	38%	99.124s	59.877s	39%
UR10e robot study case						
	Static obstacle tests			**Dynamic obstacle tests**		
Parameter	**Benchmark**	**d-APF**	**Reduction**	**Benchmark**	**d-APF**	**Reduction**
Component	1.3474ms	1.265ms	6%	1.3751ms	1.3239ms	3.7276%
Controller	2.0504ms	2.3839ms	−16%	2.0949ms	2.2558ms	−7%

Regarding the Component execution timings, it is worth noticing that the Component performance has been diminished from a 16-17% of timing reduction for the SUPSI robot to 3-6% of timing reduction in the UR10e. This about 10% trimming in the required computational time could be a consequence of the simulated scene. While, on the one hand, the SUPSI robot utilizes two cameras, the UR10e only employs one. Therefore, the computational stress that the simulation performance is suffering is less than in the first case, leading to a reduction of the required overall time to compute responses.

Despite this difference, it is relevant that in both scenarios the Component requires less time to compute a response than the DLS-APF controller. Moreover, in the SUPSI robot study case, the Component of the DLS-APF controller cannot compute a response in less than 2ms (the required control loop frequency). So it would not seem recommendable to apply to a hard real time controller since some control references will not considered an updated repulsive component due to the obstacle. Which will not lead into a collision, but it will increase the risk for the operator in the collaborative application.

The fact that the Controller component has to perform more operations to compute a response due to the singularity repulsive component produces an increase of the required time to compute a control response. Even though the reduction is similar in both scenarios, it is more worrying the fact that for the UR10e robot, the utilized time to compute a new control reference is over 2ms. Bearing in mind the good performance in the SUPSI robot, the fact that is a simulation test, and that in the following section (Section 8.3) the real time suitability test are applied to this robot, no drawbacks are highlighted yet for this behavior.

Relative to the pick and place tasks with and without obstacles the required time to perform an operation is also reduced in comparison to the DLS-APF controller. The first remarkable fact of the test on simulations highlights a overall reduction of around a 40% in the time needed to execute a full pick and place operation for both cases (static and dynamic obstacles). This improvement in the manufacturing performance of the simulated battery pack collaborative disassembly process can be due to

two factors. Firstly, since the d-APF controller performs more efficiently than the DLS-APF one, the computational load of the simulation engine is lower so the simulation can be executed faster in time. Moreover, the paths followed by the avoidance when using the proposed controller are smoother while avoiding simultaneously the collision and the singularity. Herein, the risk to be trapper into singular configurations is reduced and the movements executed by the manipulator are more straight forwarded, reducing the required time for each pick and place cycle.

The outperform of the d-APF controller over the DLS-APF control approach is also appreciated in comparisson to the standalone manually executed disassembly process. From empirical data measured on the shop-floor manual process, a standard disassembly average cycle for one battery lasts 454s. This performance makes a total of 60 batteries process by one operator during a single regular working shift of 8 hours. Introducing a single collaborative robot in the process generates at least 2 robot safety stops due to collisions (statistical data taken from over 6 months of data measurement). Every time one of these safety stops happens, it takes roughly 20s to be operational again since the operator has to rearm the robot and restart the program. In general, introducing a cobot to this manufacturing process will increase the disassembly time of a battery pack to 494s, meaning a 9% of productivity loss. Therefore, introducing the advanced robot controller proposed in this thesis can deal with this timing increase by reducing the non-critical safety stops due to avoiding simultaneously the collision with dynamic obstacles and singularities. Using the data from Table 8.4, it can be stated that a full battery pack disassembly cycle requires around 439s to be completed. This behavior implies a reduction of 11% in the required time to execute a disassembly cycle in comparison to the original collaborative operation (494s) due to partially optimized trajectories and the partial absence of production stops.

Summarizing the results obtained in the simulated test, the most relevant issue to highlight is that the controller behaves as expected, reducing the required computational load to compute safe responses to avoid obstacles in the case of the proposed d-APF controller. This improvement is also appreciated in the collaborative application test where the required time to disassembly a full pack has been roughly decreased an 11%. Lastly, even though the Controller component works suitable for the SUPSI robot in the application, the following section will test the performance of the proposed controller and its real time suitability in the case of the UR10e robot.

For further details in how the values from Table 8.4 are obtained, the different testing results are gathered below in the following figures from Figure 8.5 to 8.9. In Figure 8.5 the timing execution for the pick and place tasks executed by the SUPSI robot with and without obstacle are appreciated. Consequently the Component and Controller timing performances are also presented for each controller and use case. This way, the result for the static obstacle tests for the SUPSI and the UR10e robot are collected in Figure 8.6 and 8.7, respectively. While the case for the dynamic obstacle avoidance is presented in Figure 8.8 for the SUPSI robot and in Figure 8.9 for the UR10e.

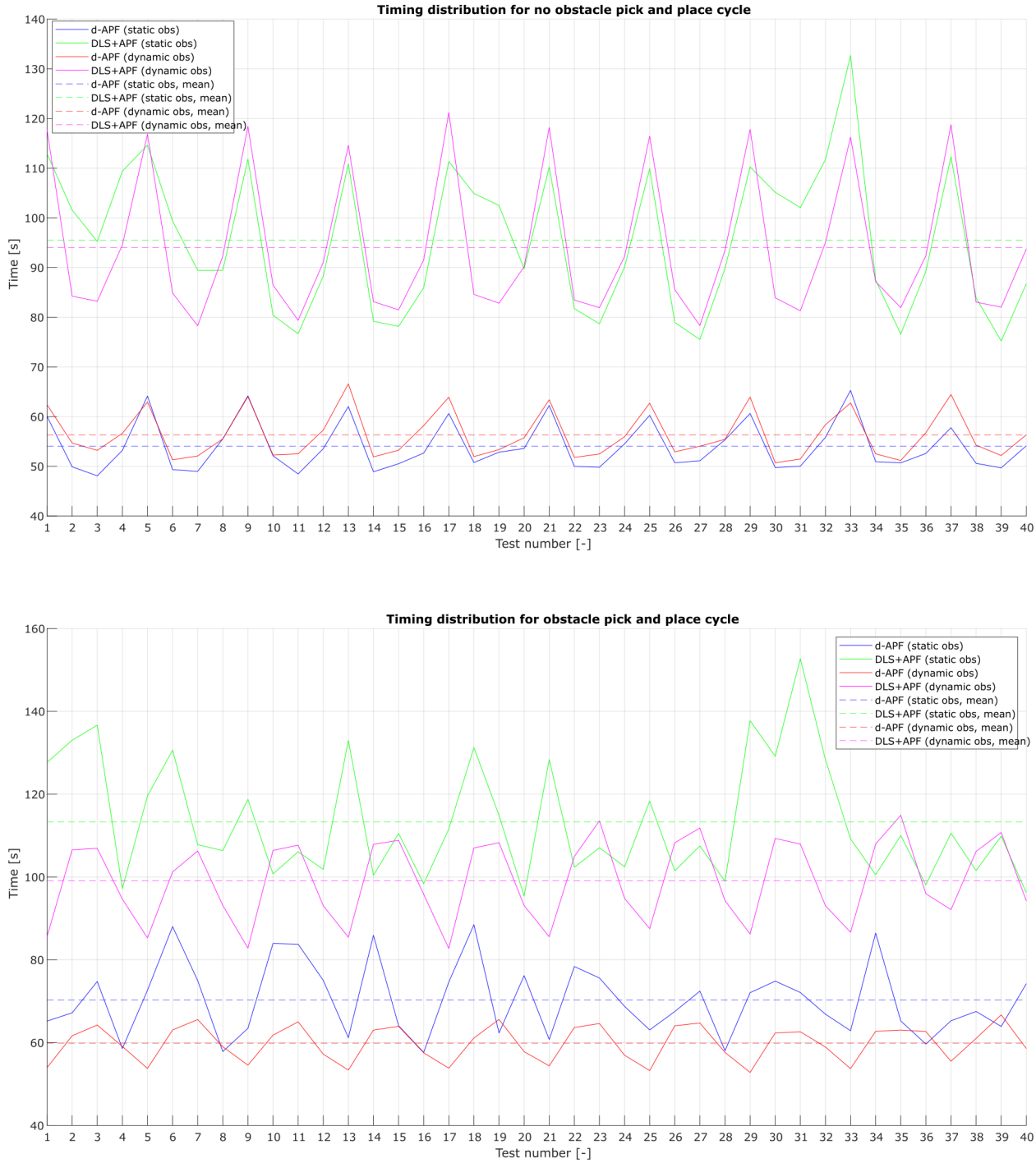

Figure 8.5: Results of the task execution with and without obstacle in the RTT ROS controllers simulated test: At the left hand side the timing distribution for no obstacle pick and place cyles, and at the righ hand side the timing distribution for the obstacle avoidance cycles.

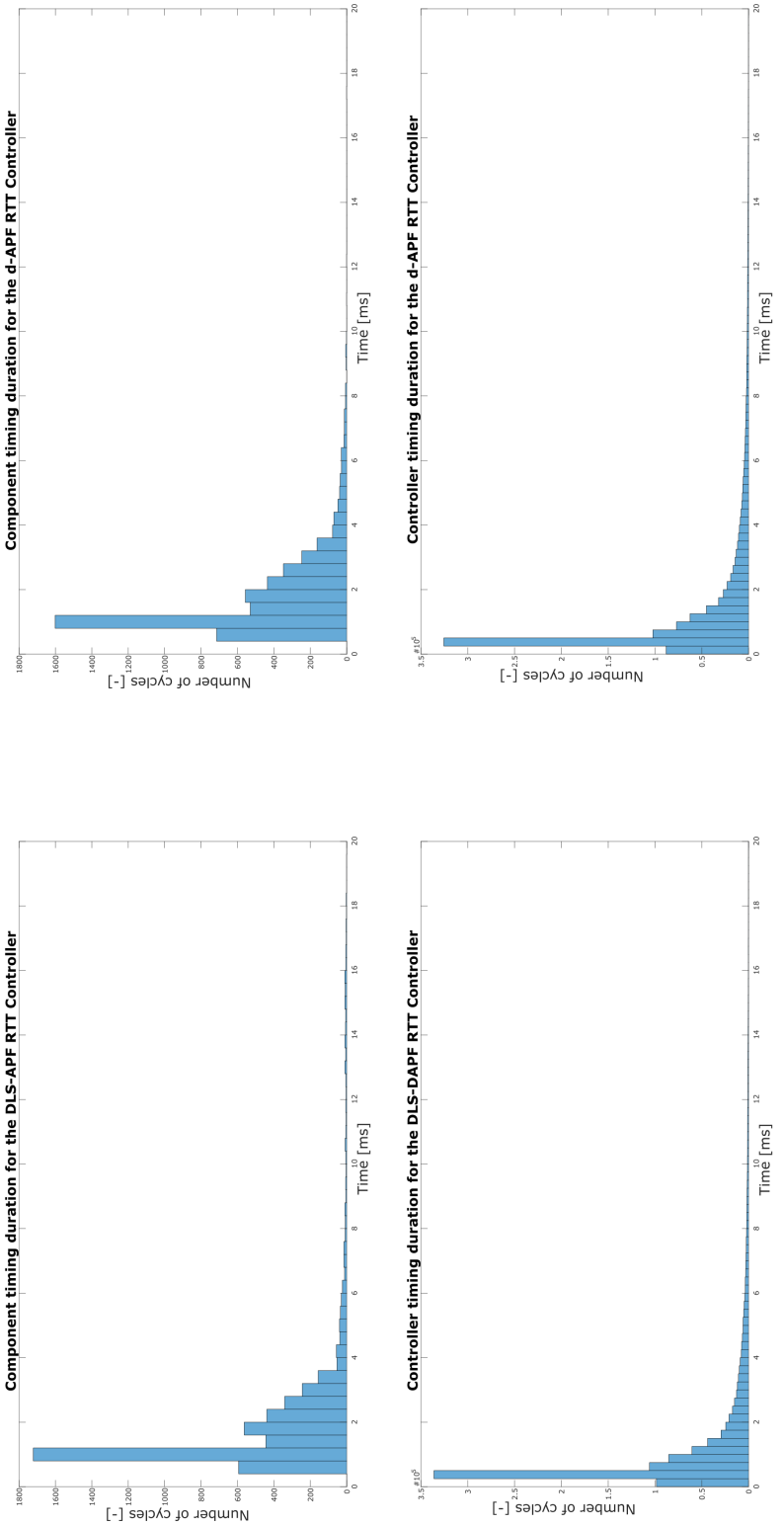

Figure 8.6: Results of the timing performance of the Component and the Controller in the RTT ROS controllers simulated test for the SUPSI robot (static obs.): On the top left corner Component timing performance of the DLS-DAPF controller, on the top right corner the Component timing performance of the d-APF controller, on the bottom left corner the Controller timing performance for the DLS-DAPF controller, and on the bottom right corner the Controller timing performance for the d-APF controller.

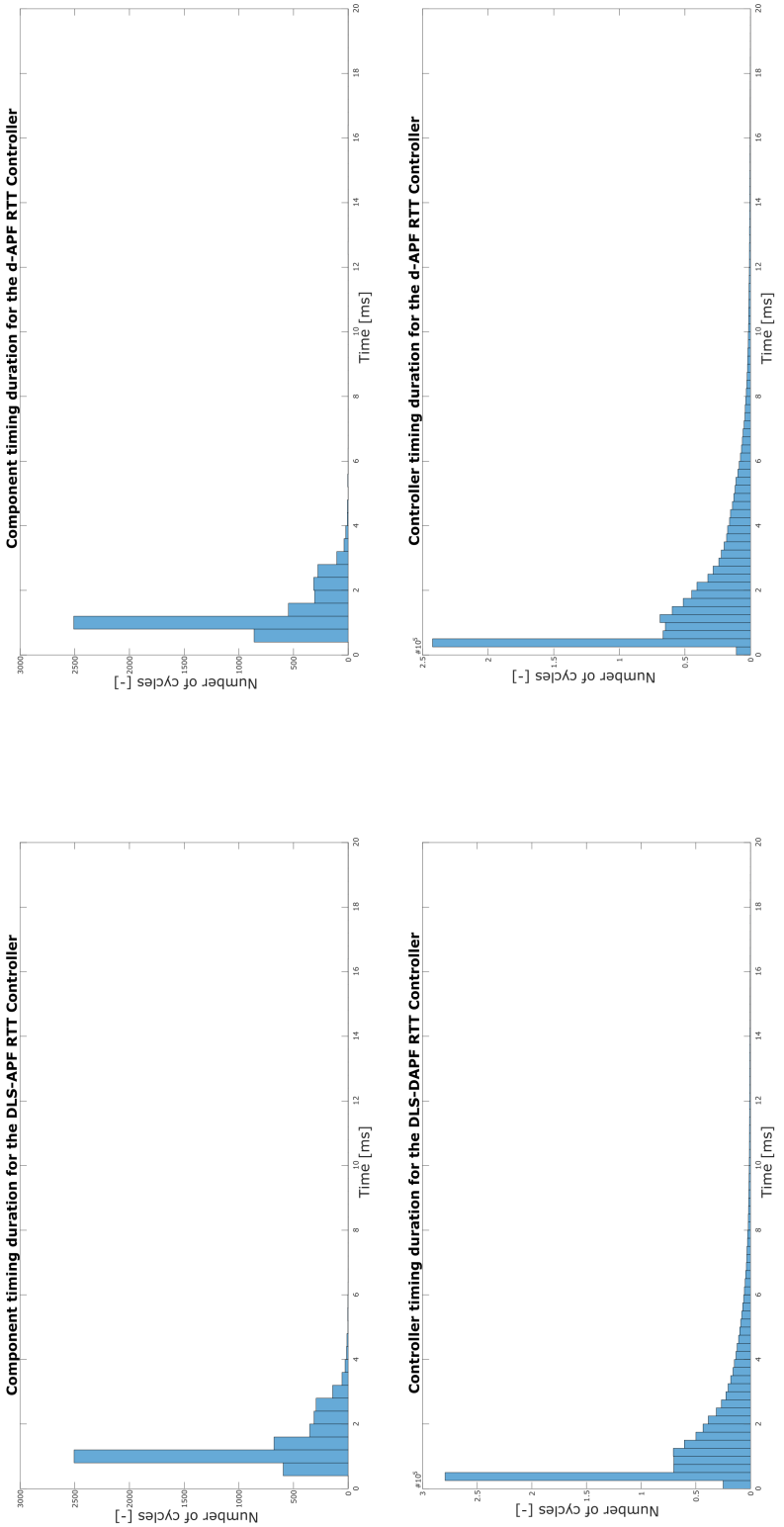

Figure 8.7: Results of the timing performance of the Component and the Controller in the RTT ROS controllers simulated test for the UR10e robot (static obs.): On the top left corner Component timing performance of the DLS-DAPF controller, on the top right corner the Component timing performance of the d-APF controller, on the bottom left corner the Controller timing performance for the DLS-DAPF controller, and on the bottom right corner the Controller timing performance for the d-APF controller.

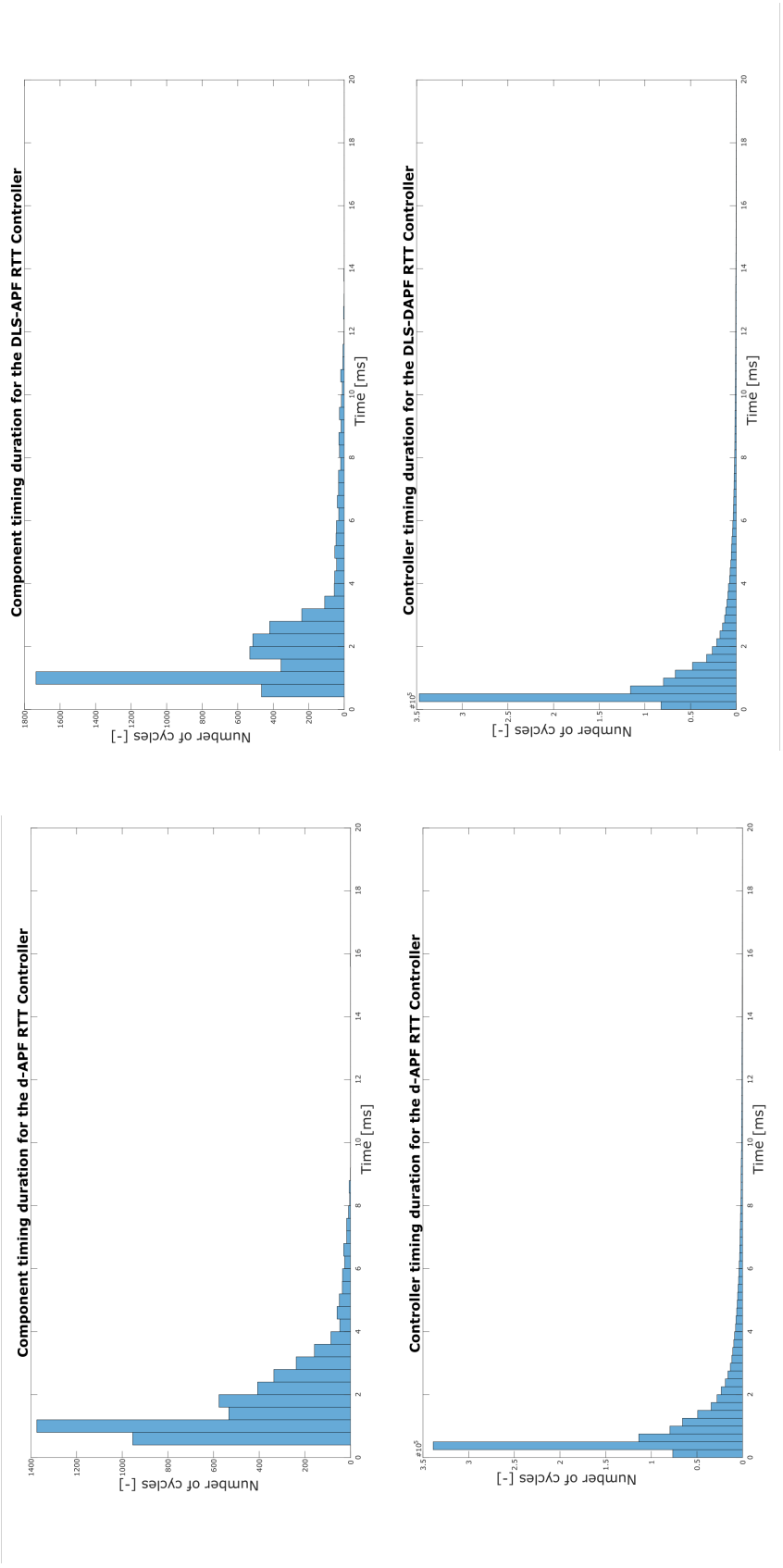

Figure 8.8: Results of the timing performance of the Component and the Controller in the RTT ROS controllers simulated test for the SUPSI robot (static obs.): On the top left corner Component timing performance of the DLS-DAPF controller, on the top right corner the Component timing performance of the d-APF controller, on the bottom left corner the Controller timing performance for the DLS-DAPF controller, and on the bottom right corner the Controller timing performance for the d-APF controller.

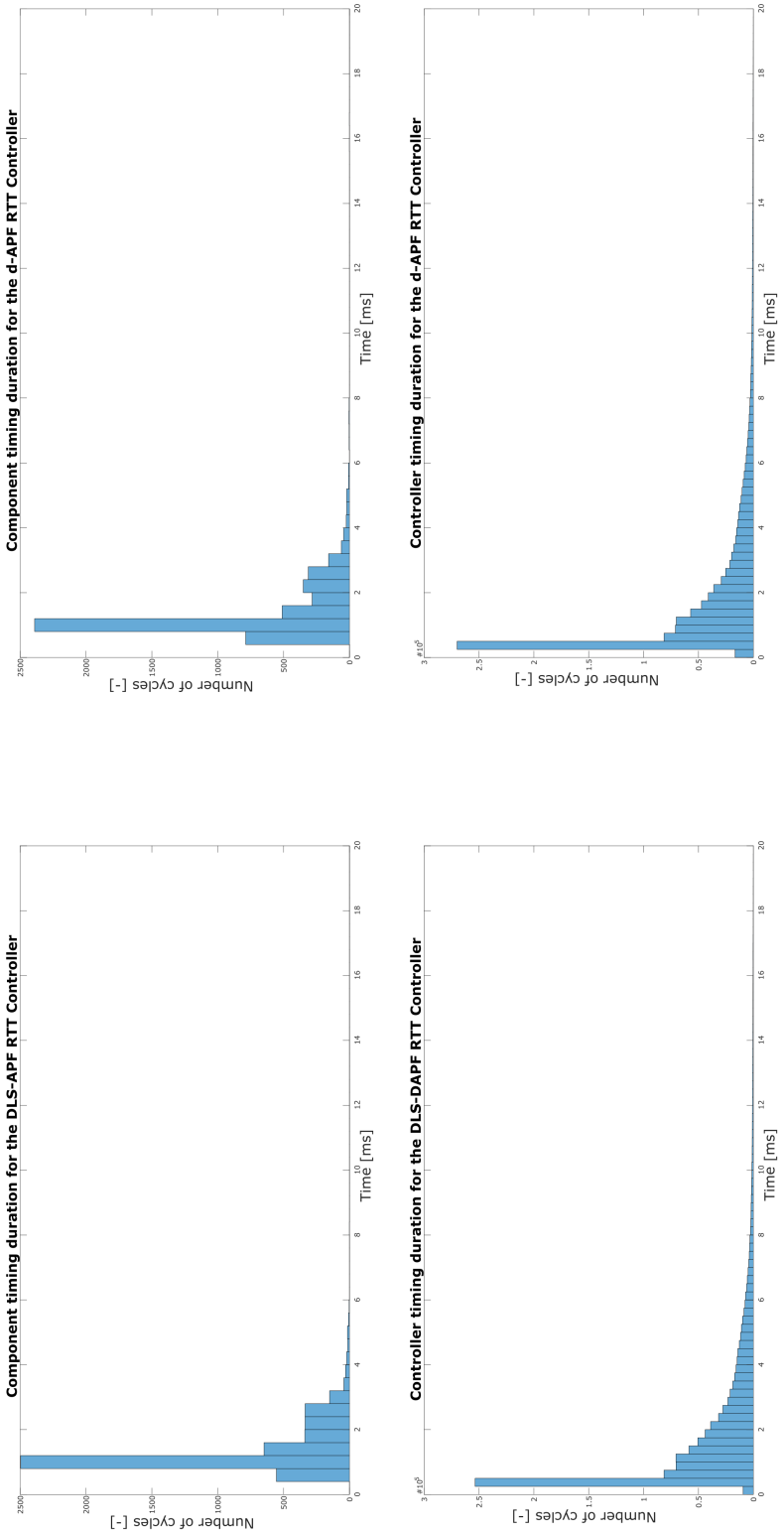

Figure 8.9: Results of the timing performance of the Component and the Controller in the RTT ROS controllers simulated test for the UR10e robot (static obs.): On the top left corner Component timing performance of the DLS-DAPF controller, on the top right corner the Component timing performance of the d-APF controller, on the bottom left corner the Controller timing performance for the DLS-DAPF controller, and on the bottom right corner the Controller timing performance for the d-APF controller.

8.3 Real time suitability tests

The real time suitability test aims for testing if the controller can send updated references to the robot in the required control loop cycle frequency of 500Hz[11]. As seen in the previous section, the simulated case of the UR10e robot, for the controller timing performance presents a timing issue. Therefore, this tests is also employed to clear out the doubts about the application of the proposed controller in a real scenario, since the misbehavior could be due to internal non-manageable processes of the simulation computer.

11: Frequency corresponding to a loop cycle of 2ms.

The executed test pretends to test the performance of the APF component and the RTT ROS controller component for each of both RTT ROS Controller implemented. As the main objective of the test is to check the timing performance of the controller, in this occasion no specific task is performed during the test. However, it is required for the user to force the actuation of the repulsive components of the APF randomly, as well as the execution of random trajectories for the robot. By introducing this random errors into the proof, the timing efficiency of the controller can be tested even in the scenario of non-handled disturbances. Hence, once again, 1000000 cycles have been executed to measure the timing performance of the Controller component, while 5000 sampled are collected for the case of the APF computational component.

The raw collected data are presented in Figure 8.10 where it can be appreciated a histogram for the Component and the Controller[12] for the DLS-DAPF controller at the left side of the picture and the d-APF controller data at the right hand side. Furthermore, the performance of each Orocos component has been analyze through a box plot in order to better appreciate the distribution of the read data (see Figure 8.11). In addition to the raw data, the Table 8.5 is included below where the maximum (Max), minimum (Min), standard deviation (Std) and mean (Mean) are presented for each controller component (Component and Controller)[13].

12: This adopted names match with the ones of the previous section.

13: For each ROS Control-Orocos component, the Benchmarking results are once again referred to the DLS-DAPF Controller, and the d-APF ones are related to the proposed controller in this Ph.D.

	Component		Controller	
	Benchmark	**d-APF**	**Benchmark**	**d-APF**
Mean	0.4457ms	0.3676ms	0.1010ms	0.1063ms
Std	0.1584ms	0.0828ms	0.0222ms	0.0230ms
Max	2.1414ms	1.9837ms	1.4349ms	1.3383ms
Min	0.2248ms	0.1795ms	0.0225ms	0.0233ms

Table 8.5: Results summary of the RTT ROS Controllers real time suitability tests. The Component and Controller rows of the table are measured in milliseconds (ms) since they represent control cycle loop execution times. Moreover, the Task with and without obstacles rows represent the timing of a full pick and place cycle, therefore it is measured in seconds (s)

First of all, after analyzing the results obtained from the real time execution of the controller in the UR10e robot (summary exposed in Table 8.5), it seems clear that the misbehavior from the previous section was due to non-controlled processes in the simulation computer. Regarding the APF computation component, the proposed d-APF algorithm outperforms a rough 17% over the benchmark controller. Consequently, the efficiency of the controller component also behaves as expected, being a little heavier in the computation of the output. The increase in the required computational time is aligned with the simulation results being nearly a 5% of increment.

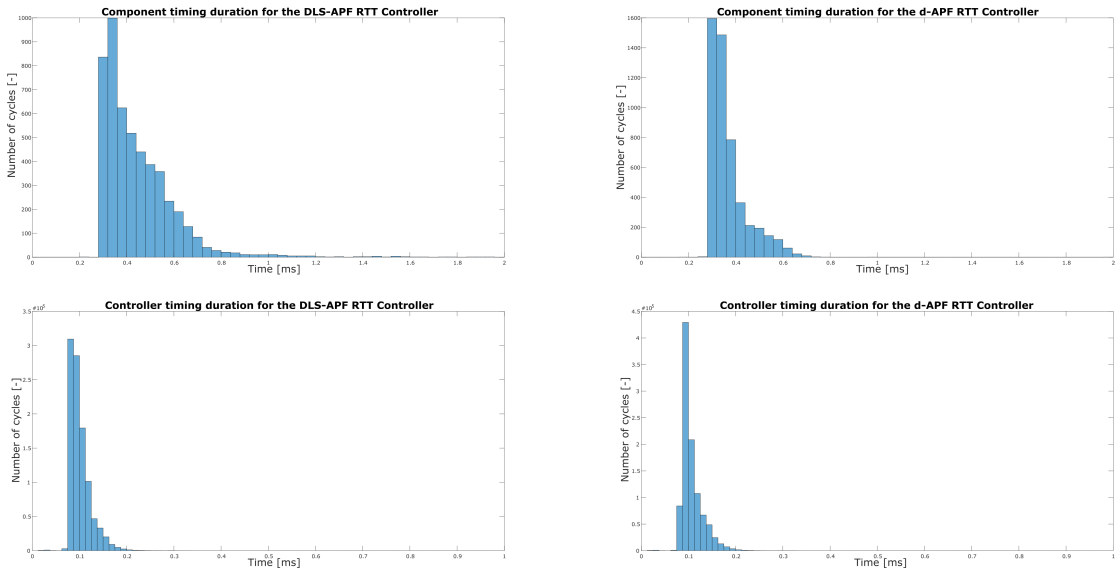

Figure 8.10: Real time suitability tests timing histograms: On the top left corner Component timing performance of the DLS-DAPF controller, on the top right corner the Component timing performance of the d-APF controller, on the bottom left corner the Controller timing performance for the DLS-DAPF controller, and on the bottom right corner the Controller timing performance for the d-APF controller.

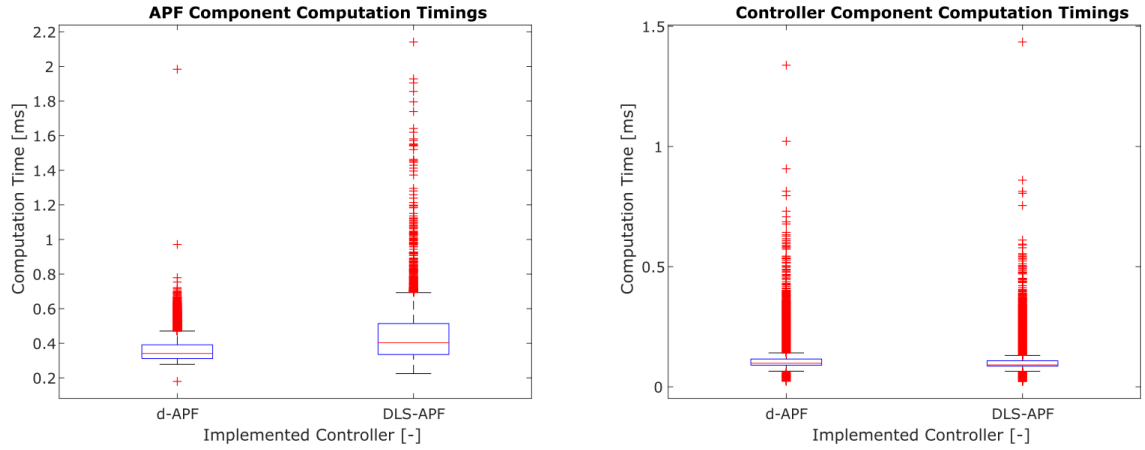

Figure 8.11: Real time suitability tests timing boxplots: At the left hand side the results for the APF Component computation timings, and at the right hand side the results for the RTT Controller Component timings.

More importantly, in contrast to the DLS-APF controller, the proposed d-APF controller never violates the limit of 2ms for assuring an updated reference when sending commands to the robot. However, as seen in Figure 8.11, this violation is not recursively, so loosing the APF computational component one cycle out of 1000000 cycles, is not a big deal[14]. Therefore, even though the d-APF controller outperforms the DLS-APF one, both are suitable for its usage in soft real time controllers.

14: This conclusion is also supported by the fact that it has been employed the PREEMPT_RT kernel patch. It is a soft real time patch, so little variability in the execution is accepted.

8.4 Holding position performance

The holding position performance test aims to check whether is suitable the proposed controller to maintain a configuration without risking the operator. Therefore, this tests executes a holding position operation

where the operator forces the robot to be pushed away to avoid the collision. The repulsive capabilities of the designed algorithm has been proven for each RTT ROS Controller (the DLS-APF, and the d-APF ones) with and without pivoting vector on. In this manner, the experimental phase of this test consists of three steps as shown in Figure 8.12: i) let the robot hold the desired position, ii) get closer to the robot to start the repulsive components, and iii) leave the robot to be attracted to the held position.

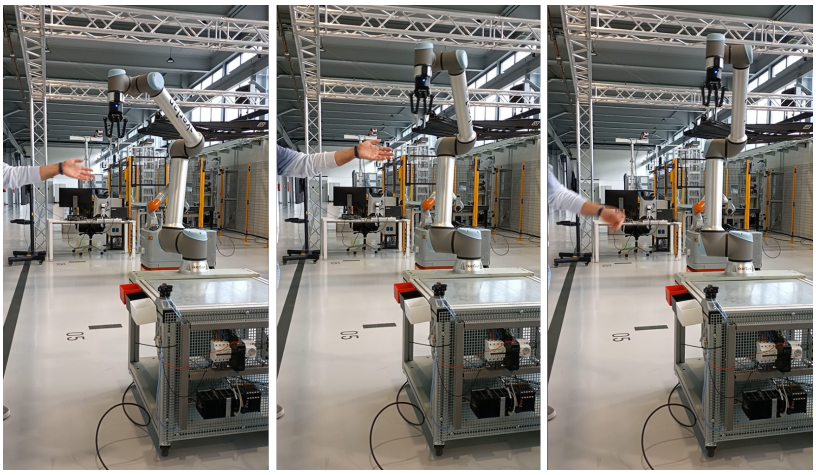

Figure 8.12: Sequence of a non pivoting static collision avoidance (d-APF Controller): The first frame is the initial state of the repulsion, the second frame is a point in between, and the last one represents when the repulsive vector is null because the obstacle is far again.

Following the described process and registering the feedback of the joint states topic, the behavior of the robot has been recorded. Even though several tests have been executed, for the shake of clarity, Figure 8.13 only displays the results for one trajectory for each of the four possible use cases explained in the paragraph above. In addition to this results, Figure 8.14 shows the virtual behavior in the ROS Parameter Server through the RViz visualization of the moment where the robot is repelling the collision.

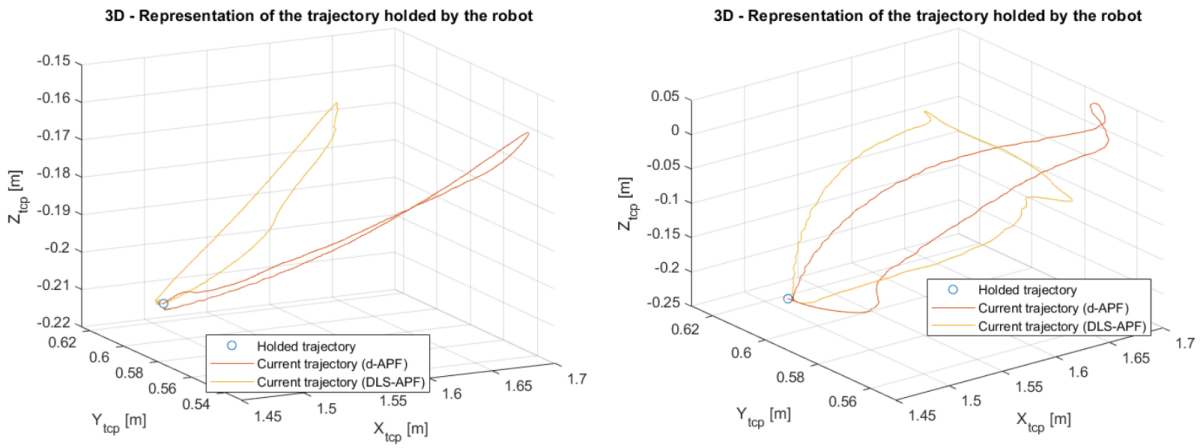

Figure 8.13: Collision and singularity avoidance test for holding a reference: At the left hand side the results without the pivoting vector on, and at the right hand side the results with the pivoting vector on.

The results obtain from the experimentation show that after the repulsive vector has stopped actuating, the robot recover the original position in each situation. On the one hand, the test executed with the pivoting vector off, displays a more agresive avoidance strategy. Indeed, the performance of the collision avoidance of the d-APF controller seems softer and quicker

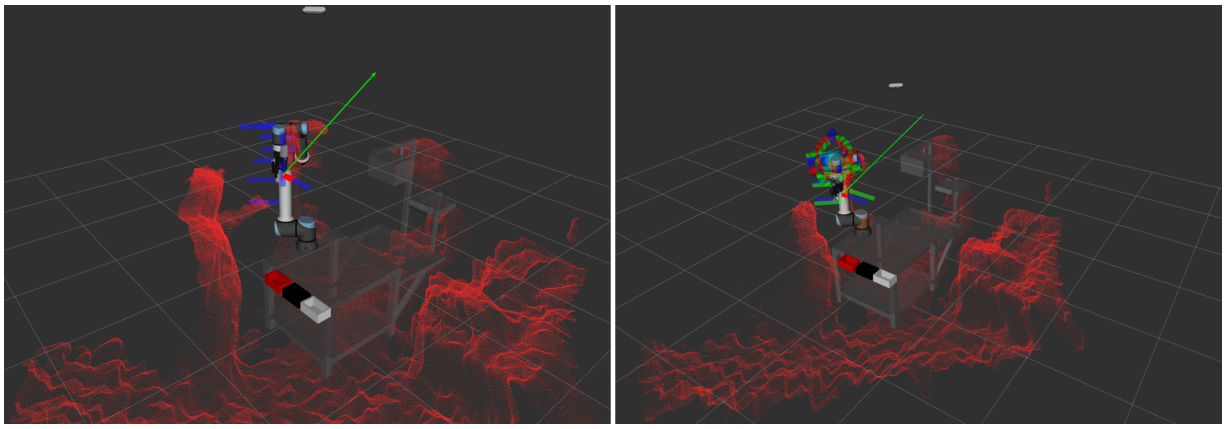

Figure 8.14: Virtualization environment of RViz during the collision avoidance: at the left hand side the response for the DLS-APF is shown, while at the right hand side the response is the corresponding one due to the d-APF controller. The green vector is referred to the collision avoidance repulsive vector, while the blue and green bars represent the collision and singularity risk indexes implemented.

than the response of the benchmarking one. It means that in the case of the first controller, the robot reacts quicker and safer to a collision, maintaining in both situations the desired position after the risk has passed.

In the case of the pivoting active responses, a similar behavior can be observed. However, the effect of the pivoting vector is clearly appreciated at the right hand side of Figure 8.13 where a wider trajectory is exectued to avoid the collision. In this occasion, the d-APF controller responds quicker to the obstacle, and even, it can be appreciated the influence of the singularity repulsive vector in the middle of the trajectory with the little arc in the top of the trajectory. On the contrary, the DLS-APF controller response is not only slower but wider too. The fact that the trajectory is wider is a drawback because it could end up in undesired collateral collision with other components of the scene, damaging them.

The wider and more jerky response of the DLS-APF controller is related to both: the different utilized kinematic model and the singularity handling technique. On the one hand, the more simplified kinematic model proposed for the non-spherical wrist manipulators in this work makes a smoother response whenever the obstacle comes closer to the robot. In addition, the less aggressiveness of the response is also appreciated for the proposed controller in the case of the pivoting vector. In the case of the DLS-APF controller, prior to returning to the position the robot is closed to a singularity obtaining a jerkier trajectory. In contrast, the d-APF controller displays a little loop before turning back to the rest position which is due to the avoidance of the singularity during the obstacle avoidance in order to prevent the arm to be fully stretched.

For these reasons, in general, the behavior of the DLS-APF controller is less desirable because it can react late to a risk in the environment or add additional risks to the control response. Therefore, the proposed d-APF controller seems to present a more smoother behavior in the vicinity of a collision. As it can be observed, the proposed controller avoids the singular configuration while presenting a less aggressive response that can increase the confidence of the human-robot partner. Thus, the presented behavior of avoiding the singularity instead of working around the singular configurations handling the velocities, seem

more efficient to be applied in regular industrial shared environments. The following section tests the behavior of both controller for a less controlled environment where the robot performs a part of the pick and place cycle.

8.5 Trajectory tracking performance

The trajectory tracking performance tests pretends to study the behavior of the proposed d-APF controller tested against the traditional DLS-APF controller while performing the tracking of trajectories. In order to obtain reliable results in the trajectory tracking with the proposed controller enabled two types of tests are executed. First, a classical error position trajectory tracking study is perform to check whether or not the robot can follow a trajectory without any collision or singularity disturbance. A repulsive scenario is then experimented to compare the behavior of each controller according while tracking a trajectory instead of a static spot. The following subsections explains each experimental phase, the goal, and metrics employed, analyzing at the end the obtained results.

8.5.1 Without repulsion

The first check executed consist of a trajectory tracking error typical study. Since a robot is a machine that in most situations moves from one point to another to transport different types of workpieces, this test pretends to test the accuracy of executing pick and place task with the implemented controllers. In order to obtain reliable results, the test executed in different situations the same trajectory at two different speeds: at 30% of the maximum speed of the manipulator, and at full speed (100%). The different scenarios studied correspond to the previous four tested for the holding positions tests: the d-APF controller with and without the pivoting vector on, and the same cases for the DLS-APF controller. The results collected for each different velocity in the non-repulsive trajectory tracking test are presented in Figure 8.15.

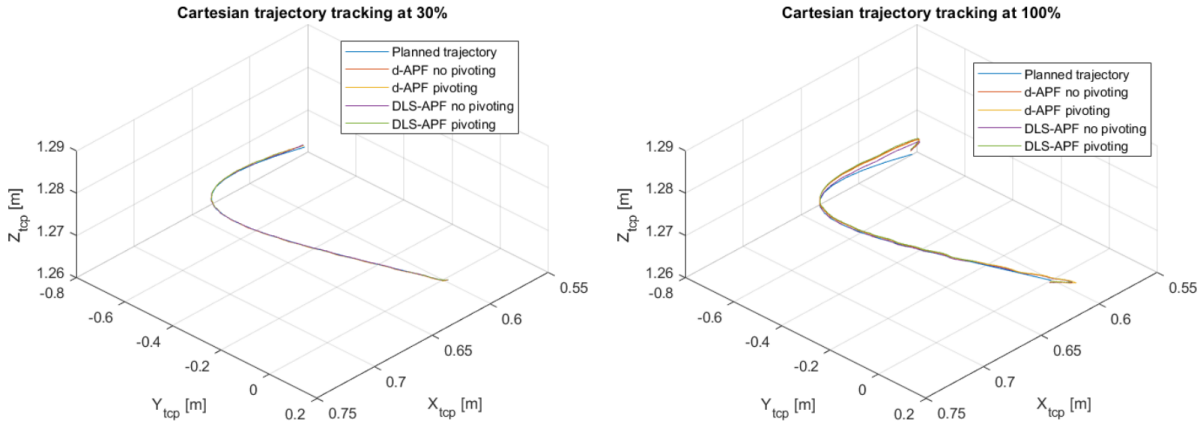

Figure 8.15: Trajectory tracking performance without repulsion: At the left side it can be found the trajectory tracking executed at 30% of the maximum speed of the manipulator, while the rigth image represents the same trajectory tracking at a 100% speed.

At a first sight at Figure 8.15, whenever the lower the velocity is, the better trajectory tracking is performed. For the case of a 30% of the maximum speed of the manipulator there is not much to say since it seems to track well the trajectory. Moreover, in both situations the robot is able to reach the desired target configuration, so the observed behavior seems suitable for its application to real collaborative applications where the robot should carry the load from one place to another. Despite that success, in the case of full speed, the biggest trajectory tracking error are at the beginning and ending of the trajectory. This behavior means that during the acceleration and deceleration processes the controller cannot follow the planned velocity slope, resulting in a little deviation from the planned trajectory. This means that even though the trajectory is deviated during the acceleration and deceleration steps, the robot arrives to the desired configuration to execute the pick and place task. In addition, to perform a better analysis and not just some appreciations from the global trajectory tracking task, Figures 8.16 and 8.17 represent the error while tracking the trajectory for the slowest and the fastest velocities, respectively.

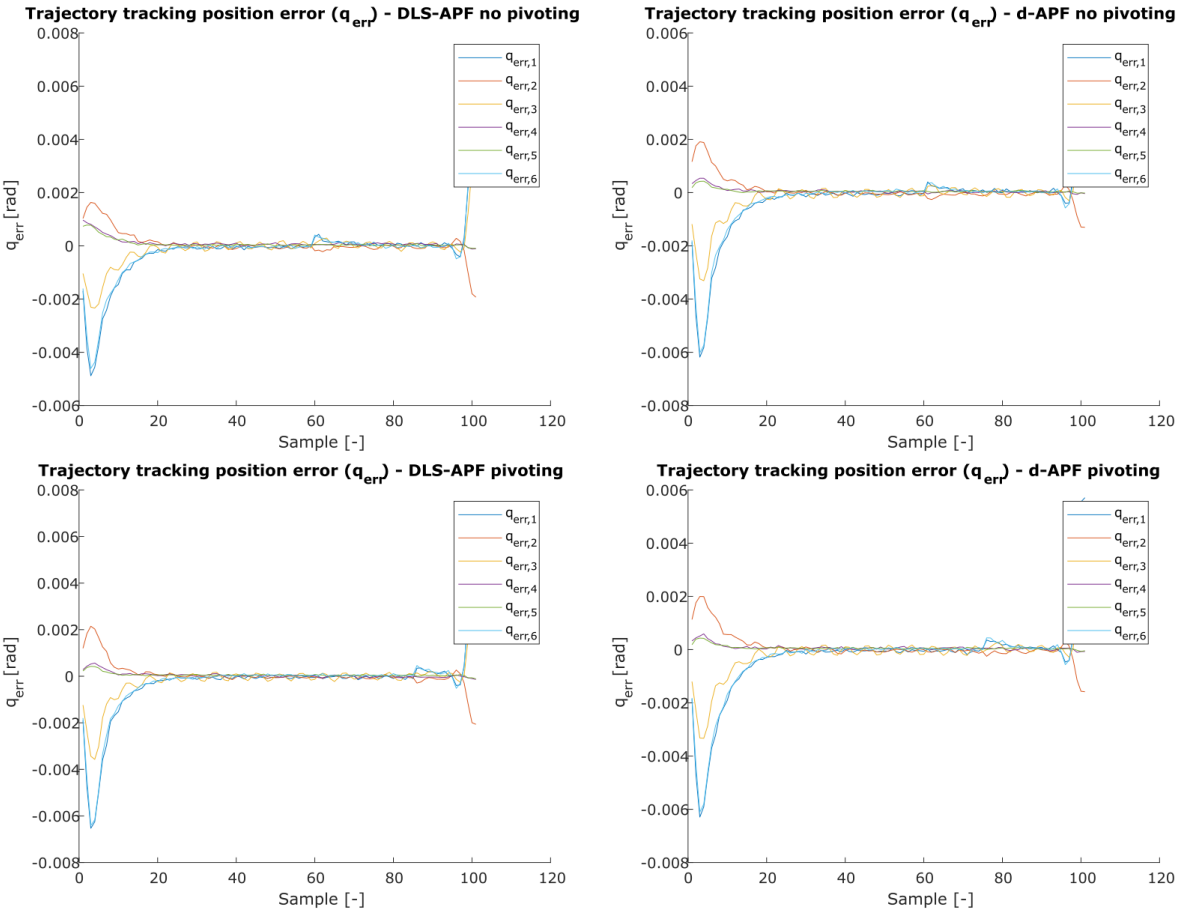

Figure 8.16: Error on position tracking at a 30% speed of the manipulator: Top figures are related to the error in the case of no pivoting on in the controller, while the botton figures correspond to the pivoting on scenario. Likewise, the left hand side figures are referred to the DLS-APF controller as well as the rigth hand side pictures are realted to the d-APF controller.

Regarding the results displayed in Figure 8.16 (referred to the tests at 30% of the maximum speed of the robot), the error visualized for each of the study cases is similar to the prior commented. Firstly, it is worth mentioning that once the robot has come through the initial steps of the movement, the error while tracking the trajectory is practically zero

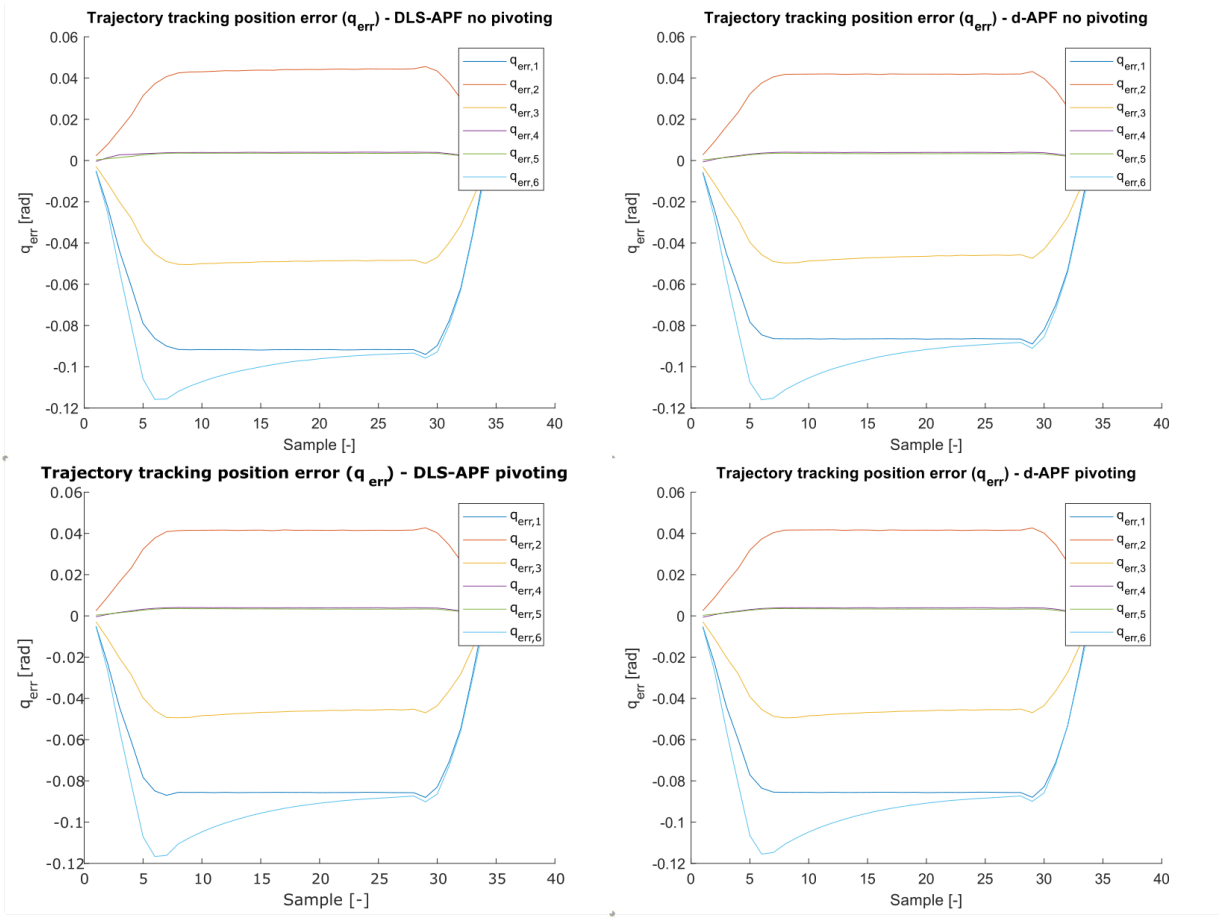

Figure 8.17: Error on position tracking at a 100% speed of the manipulator: Top figures are related to the error in the case of no pivoting on in the controller, while the botton figures correspond to the pivoting on scenario. Likewise, the left hand side figures are referred to the DLS-APF controller as well as the rigth hand side pictures are realted to the d-APF controller.

(mean tracking error per axis around $0.001rad$), so the cobot follows the trajectory in the middle stages of the task. In addition, even though during the acceleration and deceleration phases of the trajectory tracking the position error increases, at the beginning and end steps of the task the error tends to zero for the rest states in between pick and place operations. Therefore, for a 30% of the velocity of the manipulator, both controllers (the benchmarking and the d-APF ones) performs correctly trajectory tracking task without remarkable differences from one another.

Nevertheless, this behavior of compensating the position tracking error is not appreciated at Figure 8.17 where the position error collected data for the 100% velocity test are shown. In this occasion, the acceleration error is hold for the whole trajectory in every joint of the robot, being reduced then during the deceleration process when the robot approaches to the desired final state. In this situation, the average error in the tracking of the trajectory is between 0.003-0.08rads. Even though the robot arrives with low error to the desired configuration, the error during the execution of the trajectory at full speed is considerable.

Since the position error is maintained over time, it is clear that the origin of the error comes from a non tracked velocity profile during the acceleration and deceleration steps of the robot movement. As the presented results in Figure 8.17 a clear offset pattern, it is reasonable to

assume that the error comes from the implemented control architecture instead from the low level PID controllers of the robot control box. Indeed, it is believed that this error comes from the internal working behavior of our trajectory tracking which is embedded into the MoveIt API suite.

In a regular MoveIt trajectory tracking configurations, the allowed joint limits are narrowed to configure the trajectory tracking supervisor to be strict and stop the robot movement whenever one of those close limits is violated. However, in the proposed implementation, to allow the robot to be pushed away from the trajectory, this limits have been considerably widened. This requirement to make the proposed controller work configures the MoveIt trajectory supervisor to behave less restrictively. Therefore, the trajectory tracking for the controller in the regular ROS native communication with MoveIt is decayed.

From among the results obtained in the analysis of the standalone trajectory tracking tests, the most relevant fact is the deteriorated behavior of the MoveIt supervisor for the trajectory tracking. It is true that in both situations the robots always arrives to the desired state. However, the fact that the trajectory tracking presents a bigger error than the robot motor maximum precision (which corresponds to $\pm 2^\circ$ for the position and $\pm 11^\circ/s$ for the velocity) is not a good performance indicator. Therefore, the implemented controller can be utilized for basic pick and place task, but it could not be applied to high precision trajectory tracking task due to its unreliable behavior. Nevertheless, this behavior can be corrected through the implementation of a complementary trajectory supervisor to decouple the trajectory execution from MoveIt. Due to this reasons, the controller is considered suitable for some type of collaborative applications.

8.5.2 During obstacles repulsion

The previous section analyzes the performance of tracking an obstacle-free trajectory. However, in a shared environment this ideal scenario is hardly met. Therefore, this subsection studies the performance of the trajectory tracking for pick and place operations while the robot is avoiding the collision. Nevertheless, since the previous section displays a greater position error while tracking the trajectory for both controllers, this study is restricted just to the case of the 30% of the maximum speed of the UR10e. In this way, this last testing section executes two trajectory tracking tests one for the cases where the pivoting vector is on and off, respectively, for each of the implemented controllers. The executed analysis pretends to obtain a first reliable conclusion about the viability of implementing the developed control architecture on real collaborative scenarios. The sequence of movement executed while avoiding the collisions is appreciated in Figure 8.18 for the case of two repulsive responses during the trajectory execution with the d-APF controller.

The measured data for each test are represented in Figure 8.19 where the case for the switched off pivoting vector is observed at the left hand side, and the remaining case is at the right of the picture. In the case of no pivoting vector, the robot is subjected to only one repulsion process,

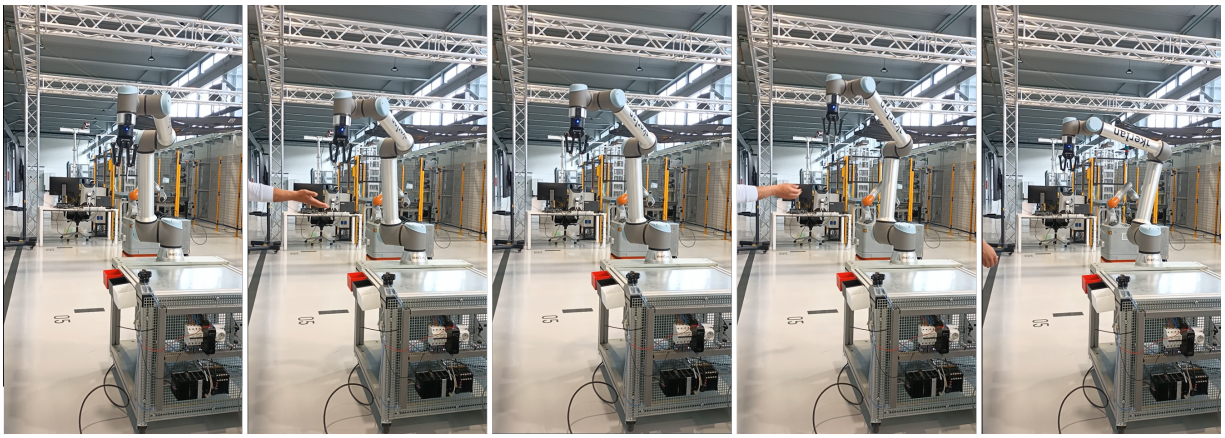

Figure 8.18: Sequence of a pivoting dynamic collision avoidance (d-APF Controller): i) Initial position, ii) First repulsion, iii) Recovery the trajectory tracking, iv) Second repulsion, and v) Final position.

while for the pivoting vector on case the robot suffers two repulsive actions at different parts of the trajectory.

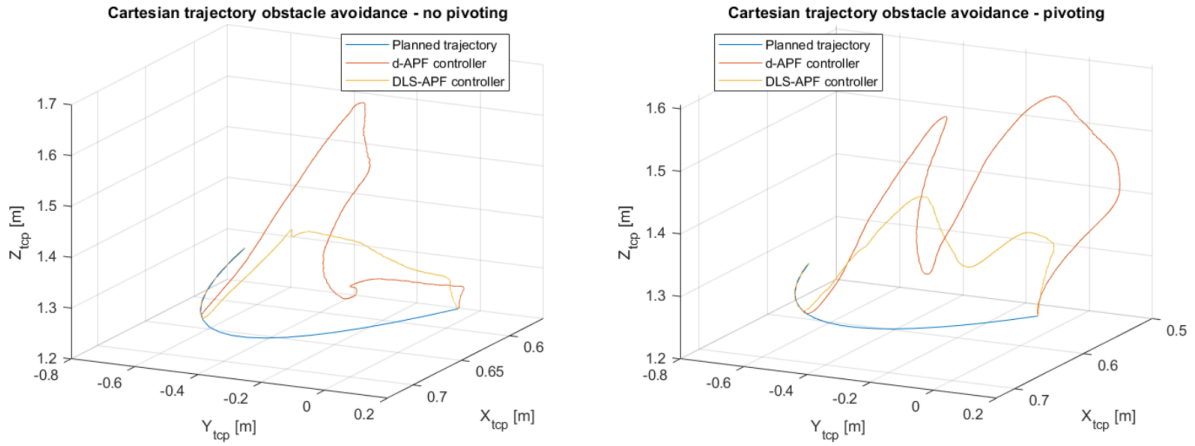

Figure 8.19: Collision and singularity avoidance test during trajectory tracking: At the left hand side the results without the pivoting vector on, and at the right hand side the results with the pivoting vector on.

In this last test similar results as the ones observed for the case of the position holding tests are appreciated. When the pivoting vector is not activated, the response is more abrupt, while when it is on the response corresponds to a wider modification of the trajectory. Furthermore, the response for the case of the d-APF controller is also smoother than the one for the reference controller (the DLS-APF one). This smoother reaction is due to the employed kinematic model, which due to the simplification of the kinematic model, more natural responses according to traditional path plannings can be computed. This behavior is also appreciated in the responses of both controller since the d-APF controller presents a response that pushes away the manipulator further than the reference model. This could be also an indication of a faster response of the proposed controller that helps improving the safety for the operator because the robot reaches more distant configurations from the operator.

Moreover, these slower reaction of the DLS-APF controller with respect to the d-APF controller can also be influenced by the singularity handling technique selected. On the one hand, the reference controller utilizes the wDLS kinematic model continuously to compute a suitable response

for the repulsive vector of the APF. Meanwhile, the proposed control solution uses a simpler kinematic model which computing each new control reference is more efficient. This continuous limitation of the velocity of the manipulator instead of letting the manipulator moving free and avoid the singularities while avoiding the collisions, directly affects the safety and performance of the application. Since the response if more narrow and slow, it will take more time to avoid any thread of the surroundings. Therefore the risk to collide with an operator or being trapped into a singular configuration increases. What makes colliding (with the robot arm or its load) of with the operator more possible when the DLS-APF controller is working.

Since the d-APF controller responds smoother and faster to collision pulling away the robot, the safety motor torques are less likely to be triggered while avoiding obstacles from the environment. Therefore, the implementation of the proposed control solution is considered more suitable for real collaborative scenarios, obtaining a control system that reacts safer and quicker to dangers of the environment.

8.6 Conclusions to Chapter 8

Positive results are extracted from the different tests executed. On the one hand, when analyzing the proposed decoupled model for non-spherical wrist cobot, it has been appreciated that the proposed model keeps the structural singularities of the robot. In addition, the strong influence of the reference system distribution is demonstrated, leading to a new type of singularities. These singular configuration belong to the already known internal singularities of the manipulators and are called model singularities. Thus, they are bounded, due to the rotations and the distribution of the reference frames, to the mathematical model of the robot and the coupling of the different trigonometric functions.

The simplified kinematic model for manipulators combined with a real time software architecture (detailed in Chapter 7) leads to the development of a real time suitable controller. The real time performance of the ROS-Orocos from the RTT ROS Controller running in parallel has been tested for both simulated and real controller. From these tests, an improvement of nearly a 17% is obtained for the component that computes the APF repulsive velocity. On the contrary, this improvement comes along with a reduction of a 5% of the efficiency for the Controller component to compute a response. However, since both required times are considerably below the 2ms of execution time, the overall real time performance is considered suitable for its application to real time control loops. In fact, the proposed architecture can run on a real time OS under the frequency of 1KHz, so this control architecture can be transported to other commercial cobot with faster interfaces.

Lastly, the performance of trajectory tracking and position holding tests shows that the d-APF controller implemented responds smoother to the possible collisions than the DLS-APF controller. However, the implemented control architecture presents the drawback of a reduction in the MoveIt supervisor strictness. However, the desired target configuration is achieved even the the manipulator is operating at full speed. Moreover,

the responses observed are smoother and faster achieving a position for the robot manipulator where the robot is further from the operator with the d-APF controller. In this manner, the robot does not only avoids blockages due to singularities, but also creates a safer environment for the operator. For all these reasons above, the proposed controller is considered more suitable for its application to industrial collaborative environment than the reference one.

Conclusion 9

9.1 General Conclusion

In general terms, this work proposed a suitable and novel control architecture to guarantee safety during collaborative applications without disregarding the efficiency of the manufacturing process. For such an aim, a d-APF controller has been proposed to react dynamically to the collisions while evading the singular configurations. To do so, firstly it was required to have a characterized set of solutions for the singular configurations. Therefore, the first two contributions came up, a novel kinematic model for non-spherical wrist robots and the singularity study. From the first one, a more simplified kinematic model has been obtained that allows more efficient computing an analytical set of closed solutions for the IK. Moreover, regarding the singularity study, a closed set of solutions for singular configurations has been extracted, allowed by the Jacobian simplification of the prior novel kinematic model. These solutions present the particularity of being a joint-dependant parametrization of the configurations where the robot kinematic model can become singular. These results have been latter applied to the proposed d-APF controller, which is by itself another relevant contribution of this work. This novel controller allows the simultaneous avoidance of collisions and blockages at the singular configurations, what enables keeping manufacturing efficiency while maintaining safety the operator. In this way, the proposed control approach outerform the traditional controllers such as the DLS-APF controller not only in terms of specific control loop cycle performance but also according to the efficiency of a whole set of operations in collaborative manufacturing environments.

In order to achieve this behavior, as regarded in Chapter 4, it is proposed a novel decoupled kinematic model for non-spherical wrist cobot. This model simplifies the robot kinematics in position and velocities to overcome the limitations of traditional kinematics models for non-spherical wrist robots. The main drawback of traditional models consists of not being able to compute a closed set of either a set of IK solutions nor singular configurations. By applying the novel wrist spherification technique to a 6 or greater DoF non-spherical wrist, a quasi-spherical wrist is obtained. In other words, it is obtained a non-spherical wrist that behaves as an spherical one. This formal contribution to the robot modeling theory has been latter particularized for a UR10e and the SUPSI robot on Chapter 7. Thanks to this implementation on each robot, a closed set of solutions for both the inverse kinematics and the singularities are obtained. Moreover, the calculated solutions for the singular configurations describe a characterization of the singular configurations dependant exclusively of q_2, q_3 and q_5 joints.

In turn, the obtained joint dependant characterization enables the implementation of the other contribution of the thesis, the d-APF controller (presented in Chapter 6). This controller consists of an advanced control algorithms to repel simultaneously the robot from the collision and

the singular configurations. In this way, the obtained joint dependent characterization is utilized to compute the distance to the singularity through a singularity risk index to limit the joint velocities and a repulsive component to push the robot away from the singular configurations. This novel controller has been latter tested against the reference one (the DLS-d-APF controller presented in Chapter 5). In this manner, the reference controller from [18] has been modified to introduce a wDLS kinematic model while computing the responses to dynamic obstacles in the environment. This modification allows the reference controller to generate a smoother response to the obstacle in the vicinities of the singular configurations. Becoming a more challenging controller to test against the novel d-APF controller.

The implementation of each controller for both robots is regarded in Chapter 7 where the novel control software architecture to encapsulate ROS Control based Controller into Orocos is first developed. This software architecture allows the implementation of real time capable ROS controllers without worrying about the programming steps of the real time synchronization of different threads. To do so, a RTT Controller Manager to embedded the ROS Control loop into a real time Orocos component and a hardware interface bridge to expose the data between the Orocos component and the ROS network are implemented. On top of this generic control architecture, any RTT ROS Controller can be implemented to run in real time. More specifically, the reference DLS-APF controller and proposed d-APF are implemented. On the one hand, for the DLS-APF implements the wDLS kinematic model inside an Orocos component thanks to the Kinematics and Dynamics Library (KDL). On the other hand, the novel kinematic decoupled model for non-spherical wrist cobots is implemented inside the d-APF controller. This has been possible through a combination of the joint dependent characterized solutions as a measure of closeness to singularities and the simplified Jacobian. In this manner, a more efficient control algorithm to avoid simultaneously singularities and collisions has been obtained.

The different tests performed to check the suitability of the proposed kinematic model and controller to real time simultaneous collision and singularity avoidance shows a plausible improvement on the efficiency of collaborative industrial operations. On one side, the d-APF controller outruns the DLS-APF controller in terms of timing performance to compute a new repulsive signal. In this sense, the proposed controller is able to compute a new response around a 17% time faster than the reference controller. This component runs in parallel to the controller component from the Orocos generated architecture. On the contrary, regarding this last controller component, it runs a rough 5% slower than the controller component from the reference controller. However, since the maximum required time to compute a new control signal is much lower than $2ms$ for both situations (even in the worst case scenario), the controller assures performing the computation of a new control reference in the desired 500Hz frequency of the robot controller. For this reason, even when the required time for the controller increases to compute a d-APF control signal, this additional time is not representative as the controller guarantees the operation in real time for the specific UR10e and SUPSI robot controller.

Moreover, proposed controller (d-APF) outperforms the reference one

(DLS-APF) also in terms of simultaneous collision and singularity avoidance. On the one hand, the smoother response of the d-APF controller is the first indicator of a better performance of the kinematic model. In this sense, the proposed controller relies on a simplified Jacobian which does not limit continuously the velocity of the manipulator. Allowing a more safer response since the distance with the operator is better kept. In addition, the allowance of handling singular configurations instead of repelling the manipulator from them of the reference controller, triggers configurations where the arm is fully stretched. These configurations are less likely desired, since the manipulator looses reaction capabilities to foresighted events becoming closer to blockages due to singularities. In this manner, this behavior leads to a safer and more efficient response for the d-APF controller on cobot where the robot maximize the distance with the operator avoiding blocking configurations. Moreover, this smoother reaction can also alleviate the confidence the operator has in a collaborative machine, such as the collaborative manipulator, what could also lead to an increase in manufacturing efficiency. So in general, the proposed d-APF controller behaves better and smoother around singular configurations than the reference one.

Furthermore, focusing the attention to the capabilities to perform pick and place operations, the d-APF controller also displays a better performance than the DLS-APF one. The required time to complete a collaborative task is nearly an 11% reduced when using the d-APF controller than the reference one. This behavior is due to the safety stops that are not avoided when using traditional singualrity handling and collision avoidance controller as the wDLS ones where the singularity is displaced from one configurations to other. In this way, the process to disassembly in simulation a full battery pack takes 494s as opposed to the 439s that the controller requires when using the d-APF controller. In addition to this reduction in the required time, it has also been studied the capabilities to perform the tracking of trajectories. In this sense, the proposed approach also performs a better trajectory tracking behavior then the one from DLS-APF solution. In both scenarios the robot reaches the desired target position with and without obstacle avoidance. However, during the trajectory, a significant error is introduced into the tracking of the trajectory due to the obstacle avoidance capabilities of the controller. This error is a reasonable one since pulling the robot far away from its original trajectory on purpose to avoid a collision increases considerably the position and velocity errors of the controller. As this behavior increases the safety for the operator, these errors are acceptable, narrowing down the applications of the controller to task where the trajectory tracking is not required but it is needed a precise placing or picking positions.

Hence, the proposed controller performs an adequate perception of the robot surroundings, works efficiently in a frequency of 500Hz, making it safe for the operator and other devices. It also increases the efficiency of fully executed manual tasks as seen in the simulated tests, avoiding non-critical safety stops that waste manufacturing time. Moreover, it is even modular in capabilities that could be integrated for the robot. Therefore, all the design requirements from Chapter 6 are gathered in the implemented d-APF controller. So it can be stated that its application is suitable for certain types of collaborative scenarios that do not involve high precision robotic tasks.

9.2 Research Outcomes

The following list shows the contributions generated during the doctorate.

- D. Rodríguez-Guerra, G. Sorrosal, I. Cabanes, and C. Calleja, "Human-Robot Interaction Review: Challenges and Solutions for Modern Industrial Environments". In *IEEE Access* 9 (2021) pp. 108557-108578, DOI: 10.1109/ACCESS.2021.3099287.
- D. Rodríguez-Guerra, G. Sorrosal, I. Cabanes, C. Calleja, and A. González "Identificación de los retos habilitantes para los escenarios industriales colaborativos". In *XVI Simposio CEA de Control Inteligente* (2021).
- D. Rodríguez-Guerra, G. Sorrosal, I. Cabanes, Aitziber Mancisidor, and C. Calleja, "Singularity Parametrization With a Novel Kinematic Decoupled Model for Non-Spherical Wrist Robots". In *ASME Journal of Mechanism and Robotics* (May 2023) 16/5: pp. 051003, DOI: 10.1115/1.4062586.
- D. Rodríguez-Guerra, G. Sorrosal, I. Cabanes, Aitziber Mancisidor, and C. Calleja, "Decoupled kinematics for non-spherical wrist manipulators". In *Proceedings of the Sixth Iberian Robotics Conference (ROBOT2023)*, Coimbra, Portugal, 22-24 November 2023, pp. XX-XX., DOI (Accepted).
- D. Rodríguez-Guerra, A. Mosca, A. Valente, I. Cabanes, and E. Carpanzano, "An advanced dual APF-Based controller for efficient simultaneous collision and singularity avoidance for human-robot collaborative assembly processes". In *CIRP Annals* (2023) 72/1: pp. 5-8, DOI: 10.1016/j.cirp.2023.04.037.
- D. Rodríguez-Guera, A. Mosca, G. Sorrosal, I. Cabanes, A. Valente, "Performance analysis of a dual APF-Based robotic controller on real collaborative scenarios". In *Journal of Intelligent and Robotic Systems (JINT)* (2023) VOL/NUMBER: pp. XX-XX, DOI (Work in progress).
- G. Sorrosal, C. Calleja, D. Rodriguez-Guerra, A. González, E. Nieto, and A. Pujana, "RIC, Robot Intelligent Control", In "ORVE - Oficina de Registro virtual" (2023), Registered Software.
- G. Sorrosal, C. Calleja, D. Rodriguez-Guerra, A. González, and A. Pujana, "DECC, Dynamic Efficient Collaboration Controller", In "ORVE - Oficina de Registro virtual" (2024), Registered Software (Work in progress).

9.3 Future Work

In order to keep up with the improvements developed in this work, there are still some open research fields that can contribute to improve the behavior of APF based robot controllers in general, and to the self developed d-APF controller for non-spherical wrist cobots, in particular. The following paragraphs regards different ideas to keep going with the development of the obtained contributions.

Firstly, to avoid the seen misbehavior when the cobot operates at high speeds, a self-developed trajectory supervisor should be designed and implemented. The ideal behavior of this new supervisor would allow the

integration of MoveIt planning capabilities; however, it should decouple MoveIt trajectory tracking capabilities from the ROS controllers. In this manner, a more versatile trajectory tracking controller could be obtained, making the proposed approach suitable for high precision tasks too.

Moreover, it would be interesting to add some specific robot dynamics to the tuning of the d-APF controller instead of relying on the Ruckig library. The main problem with the ruckig library is that it requires the physical limit of the actuators of the robot which is not always given by the manufacturer. Therefore, it is interesting to model manually the dynamics for the new proposed model for non-spherical wrist manipulators, enabling a more specific dynamic behavior to responds to obstacles in the environment. This manual description of the behavior of the manipulator can lead to enabling of techniques such as Artificial Intelligence based techniques to optimise the behavior of each repulsive component of the d-APF. In this way, the influence of the collision and singularity avoidance components will be better integrated and tuned to obtain a smoother response that increase the trust the operator has in the robot. This positive effect in the human-robot trust could lead to additional increase of manufacturing efficiency.

Finally, any other improvement regarded other aspects of the collaborative operation such as upgrades on the vision processing algorithms or other cognitive capabilities are also welcome. In particular, it would be very interesting to improve the vision algorithm with some kind of fusion sensor technique or Neural Network (NN) based algorithm to increase the efficiency and reliability of the scene segmentation. This way, with a better understanding of the surrounding of the robot, collision false positives could be avoided as well as enabling other type of approaching strategies to interact with different workpieces.

Appendix

Bibliography

The following references are displayed in citation order.

[1] Kenneth Karlsson and Marie Jonsson. 'Overview of SAAB in commercial aeronautics. Clean Sky 2 and ITD AIRFRAME Research and SAAB Aerostructures automation projects and strategy'. In: 2019 (cited on pages 4, 7, 9).

[2] Abdelfetah Hentout et al. 'Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017'. In: *Advanced Robotics* 33 (15-16 2019), pp. 764–799. DOI: 10.1080/01691864.2019.1636714 (cited on pages 4, 7–9, 11).

[3] Iñaki Maurtua et al. 'Human-robot collaboration in industrial applications: Safety, interaction and trust'. In: *International Journal of Advanced Robotic Systems* 14 (4 2017), pp. 1–10. DOI: 10.1177/1729881417716010 (cited on page 4).

[4] S. Robla-Gomez et al. 'Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments'. In: *IEEE Access* 5 (2017), pp. 26754–26773. DOI: 10.1109/ACCESS.2017.2773127 (cited on page 4).

[5] Valeria Villani et al. 'Survey on Human-Robot Interaction for Robot Programming in Industrial Applications'. In: *IFAC-PapersOnLine* 51 (11 2018), pp. 66–71. DOI: 10.1016/j.ifacol.2018.08.236 (cited on pages 4, 11).

[6] Federico Vicentini. 'Terminology in safety of collaborative robotics'. In: *Robotics and Computer-Integrated Manufacturing* 63 (January 2019 2020). DOI: 10.1016/j.rcim.2019.101921 (cited on page 4).

[7] Akira Kanazawa, Jun Kinugawa, and Kazuhiro Kosuge. 'Adaptive Motion Planning for a Collaborative Robot Based on Prediction Uncertainty to Enhance Human Safety and Work Efficiency'. In: *IEEE Transactions on Robotics* 35 (4 2019), pp. 817–832. DOI: 10.1109/TRO.2019.2911800 (cited on pages 4, 8, 11).

[8] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. 'Soft-tissue injury in robotics'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2010), pp. 3426–3433. DOI: 10.1109/ROBOT.2010.5509854 (cited on page 4).

[9] Alessandro De Luca et al. 'Collision detection and reaction: A contribution to safe physical human-robot interaction'. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* (2008), pp. 3356–3363. DOI: 10.1109/BioRob.2012.6290917 (cited on pages 4, 8, 11).

[10] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. 'Robot collisions: A survey on detection, isolation, and identification'. In: *IEEE Transactions on Robotics* 33 (6 2017), pp. 1292–1312. DOI: 10.1109/TRO.2017.2723903 (cited on page 4).

[11] ISO. *ISO/TS 15066:2016(en) Robots and robotic devices - Collaborative robots*. 2016. URL: https://www.iso.org/obp/ui/#iso:std:iso:ts:15066:ed-1:v1:en (visited on 04/24/2023) (cited on page 4).

[12] A. Valente et al. 'Deliberative robotics – a novel interactive control framework enhancing human-robot collaboration'. In: *CIRP Annals* 71.1 (2022), pp. 21–24. DOI: https://doi.org/10.1016/j.cirp.2022.03.045 (cited on pages 4, 11).

[13] Andrea Bauer, Dirk Wollherr, and Martin Buss. 'Human-robot collaboration: A survey'. In: *International Journal of Humanoid Robotics* 5 (1 2008), pp. 47–66. DOI: 10.1142/S0219843608001303 (cited on page 4).

[14] Przemyslaw A. Lasota, Terrence Fong, and Julie A. Shah. 'A Survey of Methods for Safe Human-Robot Interaction'. In: *Foundations and Trends in Robotics* 5 (3 2017), pp. 261–349. DOI: 10.1561/2300000052 (cited on page 4).

[15] Ali Ahmad Malik, Tariq Masood, and Arne Bilberg. 'Virtual reality in manufacturing: immersive and collaborative artificial-reality in design of human-robot workspace'. In: *International Journal of Computer Integrated Manufacturing* 33 (1 2020), pp. 22–37. DOI: 10.1080/0951192X.2019.1690685 (cited on page 4).

[16] Matteo Ragaglia, Andrea Maria Zanchettin, and Paolo Rocco. 'Safety-aware trajectory scaling for Human-Robot Collaboration with prediction of human occupancy'. In: *Proceedings of the 17th International Conference on Advanced Robotics, ICAR 2015* (2015), pp. 85–90. DOI: 10.1109/ICAR.2015.7251438 (cited on page 4).

[17] Nikolaos Nikolakis, Vasilis Maratos, and Sotiris Makris. 'A cyber physical system (CPS) approach for safe human-robot collaboration in a shared workplace'. In: *Robotics and Computer-Integrated Manufacturing* 56 (June 2017 2019), pp. 233–243. DOI: 10.1016/j.rcim.2018.10.003 (cited on page 4).

[18] Fabrizio Flacco et al. 'A Depth Space Approach for Evaluating Distance to Objects: with Application to Human-Robot Collision Avoidance'. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 80 (2015), pp. 7–22. DOI: 10.1007/s10846-014-0146-2 (cited on pages 5, 8, 10, 15, 22, 45, 47, 52, 55, 56, 128).

[19] Ignacio Trojaola et al. 'An Innovative MIMO Iterative Learning Control Approach for the Position Control of a Hydraulic Press'. In: *IEEE Access* 9 (2021), pp. 146850–146867. DOI: 10.1109/ACCESS.2021.3123668 (cited on page 7).

[20] Jorge Rodriguez et al. 'Fault-Tolerant Control Study and Classification : Case Study of a Hydraulic-Press Model Simulated in Real-Time'. In: *ICFTCFD 2018 : 20th International Conference on Fault-Tolerant Control and Fault Detection* (2018) (cited on pages 7, 9).

[21] Jorge Rodriguez-Guerra et al. 'A Methodology for Real-Time HiL Validation of Hydraulic-Press Controllers Based on Novel Modeling Techniques'. In: *IEEE Access* 7 (2019), pp. 110541–110553. DOI: 10.1109/ACCESS.2019.2934170 (cited on page 7).

[22] Fabio Pini, Francesco Leali, and Matteo Ansaloni. 'A systematic approach to the engineering design of a HRC workcell for bio-medical product assembly'. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* 2015-Octob (2015). DOI: 10.1109/ETFA.2015.7301655 (cited on pages 7, 9).

[23] Quality Inspection Org. *Benefits of Semi-Automation*. https://qualityinspection.org/semi-automation/. Last accessed 2023-08-27. 2017 (cited on page 7).

[24] Universal Robots. *¿Cuáles son las diferencias entre un cobot y un robot industrial?* https://blog.universal-robots.com/es/cobots-vs-robots-industriales. Last accessed 2023-08-27. 2018 (cited on page 7).

[25] Jenay M Beer, Arthur D Fisk, and Wendy A Rogers. 'Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction'. In: *Journal of Human-Robot Interaction* 3 (2 2014), p. 74. DOI: 10.5898/jhri.3.2.beer (cited on pages 7, 9).

[26] Panagiota Tsarouchi, Sotiris Makris, and George Chryssolouris. 'Human–robot interaction review and challenges on task planning and programming'. In: *International Journal of Computer Integrated Manufacturing* 29 (8 2016), pp. 916–931. DOI: 10.1080/0951192X.2015.1130251 (cited on pages 7–9, 11).

[27] Mohamad Bdiwi, Marko Pfeifer, and Andreas Sterzing. 'A new strategy for ensuring human safety during various levels of interaction with industrial robots'. In: *CIRP Annals - Manufacturing Technology* 66 (1 2017), pp. 453–456. DOI: 10.1016/j.cirp.2017.04.009 (cited on page 7).

[28] Flacco Fabrizio and Alessandro De Luca. 'Real-time computation of distance to dynamic obstacles with multiple depth sensors'. In: *IEEE Robotics and Automation Letters* 2 (1 2017), pp. 56–63. DOI: 10.1109/LRA.2016.2535859 (cited on pages 8, 14).

[29] 'Dual arm robot in cooperation with humans for flexible assembly'. In: *CIRP Annals - Manufacturing Technology* 66 (1 2017), pp. 13–16. DOI: 10.1016/j.cirp.2017.04.097 (cited on pages 8, 10).

[30] Ramy Meziane, Martin J.D. Otis, and Hassan Ezzaidi. 'Human-robot collaboration while sharing production activities in dynamic environment: SPADER system'. In: *Robotics and Computer-Integrated Manufacturing* 48 (December 2015 2017), pp. 243–253. DOI: 10.1016/j.rcim.2017.04.010 (cited on pages 8, 10).

[31] Diego Rodriguez-Guerra et al. 'Human-Robot Interaction Review: Challenges and Solutions for Modern Industrial Environments'. In: *IEEE Access* 9 (2021), pp. 108557–108578. DOI: 10.1109/ACCESS.2021.3099287 (cited on pages 8, 10, 14).

[32] Emanuele Magrini et al. 'Human-robot coexistence and interaction in open industrial cells'. In: *Robotics and Computer - Integrated Manufacturing* 61 (June 2018 2020), p. 101846. DOI: 10.1016/j.rcim.2019.101846 (cited on page 8).

[33] Nima Najmaei and Mehrdad R. Kermani. 'On Superquadric Human Modeling and Risk Assessment for Safe Planning of Human-Safe Robotic Systems'. In: *Journal of Mechanisms and Robotics* 2.4 (Sept. 2010). 041008. DOI: 10.1115/1.4002345 (cited on page 8).

[34] Andrea Maria Zanchettin et al. 'Towards an optimal avoidance strategy for collaborative robots'. In: *Robotics and Computer-Integrated Manufacturing* 59 (January 2019), pp. 47–55. DOI: 10.1016/j.rcim.2019.01.015 (cited on pages 8, 12, 14).

[35] Luca Bascetta et al. 'Towards safe human-robot interaction in robotic cells: An approach based on visual tracking and intention estimation'. In: *IEEE International Conference on Intelligent Robots and Systems* (2011), pp. 2971–2978. DOI: 10.1109/IROS.2011.6048287 (cited on pages 8, 10, 14).

[36] Alexander Winkler and Jozef Suchý. 'Dynamic collision avoidance of industrial cooperating robots using virtual force fields'. In: *IFAC Proceedings Volumes (IFAC-PapersOnline)* 45 (22 2012), pp. 265–270. DOI: 10.3182/20120905-3-HR-2030.00019 (cited on pages 8, 14).

[37] Universal Robots. *UR: Products*. 2019. URL: https://www.universal-robots.com/es/productos/ (cited on pages 8, 15, 46).

[38] OMRON. *Robots colaborativos*. 2019. URL: https://industrial.omron.es/es/products/collaborative-robots?utm_source=newsletter&utm_medium=mail&utm_campaign=2019-07-newsletter&elqTrackId=a3af48d51a9d4aa3be12c0ec5c058997&elq=d2e2fc397f7b4928aab86e7270824e20&elqaid=2966&elqat=1&elqCampaignId=3070 (cited on pages 8, 15).

[39] FANUC. *FANUC - CR and CRX/i Collaborative Robots Brochure*. 2023. URL: https://www.fanuc.eu/es/es/robots/p%c3%a1gina-filtro-robots/%d1%80obots-colaborativos (cited on pages 8, 15).

[40] Marc G. Carmichael, Dikai Liu, and Kenneth J. Waldron. 'A framework for singularity-robust manipulator control during physical human-robot interaction'. In: *International Journal of Robotics Research* 36 (5-7 2017), pp. 861–876. DOI: 10.1177/0278364917698748 (cited on pages 8, 15).

[41] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning and Control*. Vol. 48. Cambridge University Press, 2017 (cited on pages 9, 15–17, 19, 31, 37).

[42] Przemyslaw A. Lasota and Julie A. Shah. 'Analyzing the effects of human-aware motion planning on close-proximity human-robot collaboration'. In: *Human Factors* 57 (1 2015), pp. 21–33. DOI: 10.1177/0018720814565188 (cited on page 9).

[43] George Charalambous, Sarah Fletcher, and Philip Webb. 'The Development of a Scale to Evaluate Trust in Industrial Human-robot Collaboration'. In: *International Journal of Social Robotics* 8 (2 2016), pp. 193–209. DOI: 10.1007/s12369-015-0333-8 (cited on pages 9, 14).

[44] Luca Gualtieri et al. 'The Opportunities and Challenges of SME Manufacturing Automation: Safety and Ergonomics in Human–Robot Collaboration'. In: *Industry 4.0 for SMEs: Challenges, Opportunities and Requirements*. Ed. by Dominik T. Matt, Vladimír Modrák, and Helmut Zsifkovits. Cham: Springer International Publishing, 2020, pp. 105–144. DOI: 10.1007/978-3-030-25425-4_4 (cited on page 9).

[45] Abdelfetah Hentout et al. 'Key challenges and open issues of industrial collaborative robotics'. In: *RO-MAN 2018: Workshop on Human-Robot Interaction: From Service to Industry* (August 2018) (cited on pages 10, 11, 19).

[46] Jérémie Guiochet, Mathilde Machin, and Hélène Waeselynck. 'Safety-critical advanced robots: A survey'. In: *Robotics and Autonomous Systems* 94 (2017), pp. 43–52. DOI: 10.1016/j.robot.2017.04.004 (cited on page 10).

[47] Iina Aaltonen, Timo Salmi, and Ilari Marstio. 'Refining levels of collaboration to support the design and evaluation of human-robot interaction in the manufacturing industry'. In: vol. 72. 2018, pp. 93–98. DOI: 10.1016/j.procir.2018.02.026 (cited on page 10).

[48] Alessandro De Luca et al. 'Collision detection and safe reaction with the DLR-III lightweight manipulator arm'. In: *IEEE International Conference on Intelligent Robots and Systems* (2006), pp. 1623–1630. DOI: 10.1109/IROS.2006.282053 (cited on page 10).

[49] Benjamin Navarro et al. 'An ISO10218-compliant adaptive damping controller for safe physical human-robot interaction'. In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), pp. 3043–3048. DOI: 10.1109/ICRA.2016.7487468 (cited on page 10).

[50] Alexandros Kouris, Fotios Dimeas, and Nikos Aspragathos. 'Contact distinction in human-robot cooperation with admittance control'. In: *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings* (2017), pp. 1951–1956. DOI: 10.1109/SMC.2016.7844525 (cited on page 10).

[51] Biao Jia et al. 'Manipulating Highly Deformable Materials Using a Visual Feedback Dictionary'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (Section IV 2018), pp. 239–246. DOI: 10.1109/ICRA.2018.8461264 (cited on page 10).

[52] Azhar Aulia Saputra, Chin Wei Hong, and Naoyuki Kubota. 'Real-time grasp affordance detection of unknown object for robot-human interaction'. In: *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* 2019-Octob (2019), pp. 3093–3098. DOI: 10.1109/SMC.2019.8914202 (cited on page 10).

[53] J. Ernesto Solanes et al. 'Human–robot collaboration for safe object transportation using force feedback'. In: *Robotics and Autonomous Systems* 107 (2018), pp. 196–208. DOI: 10.1016/j.robot.2018.06.003 (cited on page 10).

[54] Rahaf Rahal et al. 'Caring about the Human Operator: Haptic Shared Control for Enhanced User Comfort in Robotic Telemanipulation'. In: *IEEE Transactions on Haptics* 13 (1 2020), pp. 197–203. DOI: 10.1109/TOH.2020.2969662 (cited on page 10).

[55] Nicola Maria Ceriani et al. 'Reactive Task Adaptation Based on Hierarchical Constraints Classification for Safe Industrial Robots'. In: *IEEE/ASME Transactions on Mechatronics* 20 (6 2015), pp. 2935–2949. DOI: 10.1109/TMECH.2015.2415462 (cited on page 10).

[56] Loris Roveda et al. 'Human-Robot Cooperative Interaction Control for the Installation of Heavy and Bulky Components'. In: *Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018* (2019), pp. 339–344. DOI: 10.1109/SMC.2018.00067 (cited on pages 10, 13).

[57] Riccardo Caccavale et al. 'Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction'. In: *Autonomous Robots* 43 (6 2019), pp. 1291–1307. DOI: 10.1007/s10514-018-9706-9 (cited on page 10).

[58] Lennart Bochmann et al. 'Human-robot Collaboration in Decentralized Manufacturing Systems: An Approach for Simulation-based Evaluation of Future Intelligent Production'. In: *Procedia CIRP* 62 (2017), pp. 624–629. DOI: 10.1016/j.procir.2016.06.021 (cited on page 10).

[59] J. Krüger, T. K. Lien, and A. Verl. 'Cooperation of human and machines in assembly lines'. In: *CIRP Annals - Manufacturing Technology* 58 (2 2009), pp. 628–646. DOI: 10.1016/j.cirp.2009.09.009 (cited on page 11).

[60] Riccardo Schiavi, Antonio Bicchi, and Fabrizio Flacco. 'Integration of active and passive compliance control for safe human-robot coexistence'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2009), pp. 259–264. DOI: 10.1109/ROBOT.2009.5152571 (cited on page 11).

[61] Ali Ahmad Malik and Arne Bilberg. 'Developing a reference model for human–robot interaction'. In: *International Journal on Interactive Design and Manufacturing* 13 (4 2019), pp. 1541–1547. DOI: 10.1007/s12008-019-00591-6 (cited on page 11).

[62] Jim Mainprice and Dmitry Berenson. 'Human-robot collaborative manipulation planning using early prediction of human motion'. In: *IEEE International Conference on Intelligent Robots and Systems* (May 2013), pp. 299–306. DOI: 10.1109/IROS.2013.6696368 (cited on pages 11, 14).

[63] Fabrizio Flacco et al. 'A depth space approach to human-robot collision avoidance'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2012), pp. 338–345. DOI: 10.1109/ICRA.2012.6225245 (cited on pages 12, 22, 51, 52, 55).

[64] Andrea Bonci et al. 'Human-Robot Perception in Industrial Environments: A Survey'. In: *Sensors* 21.5 (2021). DOI: 10.3390/s21051571 (cited on pages 12, 49).

[65] H.D. Cheng et al. 'Color image segmentation: advances and prospects'. In: *Pattern Recognition* 34.12 (2001), pp. 2259–2281. DOI: https://doi.org/10.1016/S0031-3203(00)00149-7 (cited on page 12).

[66] Islam I. Fouad, Sherine Rady, and Mostafa G. M. Mostafa. 'Efficient image segmentation of RGB-D images'. In: *2017 12th International Conference on Computer Engineering and Systems (ICCES)*. 2017, pp. 353–358. DOI: 10.1109/ICCES.2017.8275331 (cited on pages 12, 13).

[67] Michael Danielczuk et al. 'Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data'. In: May 2019, pp. 7283–7290. DOI: 10.1109/ICRA.2019.8793744 (cited on page 12).

[68] Yanzhu Zhang, Xiaoyan Wang, and Biao Qu. 'Three-Frame Difference Algorithm Research Based on Mathematical Morphology'. In: *Procedia Engineering* 29 (2012). 2012 International Workshop on Information and Electronics Engineering, pp. 2705–2709. DOI: https://doi.org/10.1016/j.proeng.2012.01.376 (cited on pages 12, 13).

[69] Mingliang Chen et al. 'Spatiotemporal Background Subtraction Using Minimum Spanning Tree and Optical Flow'. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 521–534 (cited on pages 12, 13).

[70] Tibor Trnovszký, Peter Sýkora, and Róbert Hudec. 'Comparison of Background Subtraction Methods on Near Infra-Red Spectrum Video Sequences'. In: *Procedia Engineering* 192 (2017). 12th international scientific conference of young scientists on sustainable, modern and safe transport, pp. 887–892. DOI: https://doi.org/10.1016/j.proeng.2017.06.153 (cited on pages 12, 13).

[71] Jaeyong Ju et al. 'Recognition of Human Group Activity for Video Analytics'. In: *Advances in Multimedia Information Processing – PCM 2015*. Ed. by Yo-Sung Ho et al. Cham: Springer International Publishing, 2015, pp. 161–169 (cited on pages 12, 13).

[72] Yerania Campos, Humberto Sossa, and Gonzalo Pajares. 'Spatio-temporal analysis for obstacle detection in agricultural videos'. In: *Applied Soft Computing* 45 (2016), pp. 86–97. DOI: https://doi.org/10.1016/j.asoc.2016.03.016 (cited on pages 12, 13).

[73] Jamal Atman and Gert F. Trommer. 'Laser-camera based 3D reconstruction of indoor environments'. In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2018, pp. 254–260. DOI: 10.1109/PLANS.2018.8373388 (cited on page 12).

[74] Yuan Been Chen and Oscal T.C. Chen. 'Image segmentation method using thresholds automatically determined from picture contents'. In: *Eurasip Journal on Image and Video Processing* 2009 (2009). DOI: 10.1155/2009/140492 (cited on page 12).

[75] Sojung Park and Minhwan Kim. 'Extracting Moving / Static Objects of Interest in Video'. In: *Advances in Multimedia Information Processing - PCM 2006*. Ed. by Yueting Zhuang et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 722–729 (cited on pages 12, 13).

[76] Xiuzhi Li and Chuanluo Xu. 'Moving object detection in dynamic scenes based on optical flow and superpixels'. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2015, pp. 84–89. DOI: 10.1109/ROBIO.2015.7414628 (cited on page 13).

[77] Andrew B. Godbehere, Akihiro Matsukawa, and Ken Goldberg. 'Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation'. In: *2012 American Control Conference (ACC)*. 2012, pp. 4305–4312. DOI: 10.1109/ACC.2012.6315174 (cited on page 13).

[78] Sundaram Muthu et al. 'Motion Segmentation of RGB-D Sequences: Combining Semantic and Motion Information Using Statistical Inference'. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 5557–5570. DOI: 10.1109/TIP.2020.2984893 (cited on page 13).

[79] Mohammad Safeea, Pedro Neto, and Richard Bearee. 'On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case'. In: *Robotics and Autonomous Systems* 119 (2019), pp. 278–288. DOI: 10.1016/j.robot.2019.07.013 (cited on page 14).

[80] M. Jirgl, Z. Bradac, and P. Fiedler. 'Human-in-the-Loop Issue in Context of the Cyber-Physical Systems'. In: *IFAC-PapersOnLine* 51 (6 2018), pp. 225–230. DOI: 10.1016/j.ifacol.2018.07.158 (cited on page 14).

[81] Constantin-Bala Zamfirescu et al. 'Human-centred Assembly: A Case Study for an Anthropocentric Cyber-physical System'. In: *Procedia Technology* 15 (2014), pp. 90–98. DOI: 10.1016/j.protcy.2014.09.038 (cited on page 14).

[82] Azfar Khalid et al. 'Security framework for industrial collaborative robotic cyber-physical systems'. In: *Computers in Industry* 97 (2018), pp. 132–145. DOI: 10.1016/j.compind.2018.02.009 (cited on page 14).

[83] B Sadrfaridpour, J Burke, and Y Wang. 'Human and Robot Collaborative Assembly Manufacturing: Trust Dynamics and Control'. In: *Proceedings of the Workshop on Human-Robot Collaboration Industrial Manufacturing. RSS* (2014), pp. 1–6 (cited on page 14).

[84] Angelo Campomaggiore et al. 'A fuzzy inference approach to control robot speed in human-robot shared workspaces'. In: *ICINCO 2019 - Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics* 2 (Icinco 2019), pp. 78–87. DOI: 10.5220/0007838700780087 (cited on page 14).

[85] Matteo Ragaglia, Andrea Maria Zanchettin, and Paolo Rocco. 'Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements'. In: *Mechatronics* 55 (December 2017 2018), pp. 267–281. DOI: 10.1016/j.mechatronics.2017.12.009 (cited on page 14).

[86] Stefan B. Liu and Matthias Althoff. 'Online Verification of Impact-Force-Limiting Control for Physical Human-Robot Interaction'. In: Institute of Electrical and Electronics Engineers Inc., 2021, pp. 777–783. DOI: 10.1109/IROS51168.2021.9636610 (cited on page 14).

[87] Mark Nicholas Finean, Wolfgang Merkt, and Ioannis Havoutis. 'Simultaneous Scene Reconstruction and Whole-Body Motion Planning for Safe Operation in Dynamic Environments'. In: Institute of Electrical and Electronics Engineers Inc., 2021, pp. 3710–3717. DOI: 10.1109/IROS51168.2021.9636860 (cited on page 14).

[88] Carlos Morato et al. 'Toward safe human robot collaboration by using multiple kinects based real-time human tracking'. In: *Journal of Computing and Information Science in Engineering* 14 (1 2014), pp. 1–9. DOI: 10.1115/1.4025810 (cited on page 14).

[89] G. J. Garcia et al. 'Visual servoing path tracking for safe human-robot interaction'. In: *IEEE 2009 International Conference on Mechatronics, ICM 2009* (April 2009). DOI: 10.1109/ICMECH.2009.4957114 (cited on pages 14, 15).

[90] Muhammad Usman Ashraf et al. 'Proactive Intention-based Safety through Human Location Anticipation in HRI Workspace'. In: *International Journal of Advanced Computer Science and Applications* 8 (4 2017), pp. 378–384. DOI: 10.14569/ijacsa.2017.080451 (cited on page 14).

[91] Daniel Sidobre and Kevin Desormeaux. 'Smooth Cubic Polynomial Trajectories for Human-Robot Interactions'. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 95 (3-4 2019), pp. 851–869. DOI: 10.1007/s10846-018-0936-z (cited on page 14).

[92] Amit Kumar Bedaka, Joel Vidal, and Chyi Yeu Lin. 'Automatic robot path integration using three-dimensional vision and offline programming'. In: *International Journal of Advanced Manufacturing Technology* 102 (5-8 2019), pp. 1935–1950. DOI: 10.1007/s00170-018-03282-w (cited on pages 14, 15).

[93] Sascha Kaden and Ulrike Thomas. 'Maximizing robot manipulability along paths in collision-free motion planning'. In: *2019 19th International Conference on Advanced Robotics, ICAR 2019* (2019), pp. 105–110. DOI: 10.1109/ICAR46387.2019.8981591 (cited on pages 14, 17).

[94] J. Sverdrup-Thygeson et al. 'Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks'. In: *1st Annual IEEE Conference on Control Technology and Applications, CCTA 2017* 2017-Janua (2017), pp. 142–149. DOI: 10.1109/CCTA.2017.8062454 (cited on page 14).

[95] Matteo Saveriano and Dongheui Lee. 'Distance based dynamical system modulation for reactive avoidance of moving obstacles'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (May 2014), pp. 5618–5623. DOI: 10.1109/ICRA.2014.6907685 (cited on page 15).

[96] Oussama Khatib. 'Real-Time Obstacle Avoidance For Manipulators and Mobile Robots'. In: (1986), pp. 500–505 (cited on page 15).

[97] Wenrui Wang et al. 'An obstacle avoidance method for redundant manipulators based on artificial potential field'. In: *Proceedings of 2018 IEEE International Conference on Mechatronics and Automation, ICMA 2018* (2018), pp. 2151–2156. DOI: 10.1109/ICMA.2018.8484459 (cited on page 15).

[98] Jean Bosco Mbede, Xinhan Huang, and Min Wang. 'Fuzzy motion planning among dynamic obstacles using artificial potential fields for robot manipulators'. In: *Robotics and Autonomous Systems* 32 (1 2000), pp. 61–72. DOI: 10.1016/S0921-8890(00)00073-7 (cited on page 15).

[99] Wenrui Wang et al. 'An improved artificial potential field method of trajectory planning and obstacle avoidance for redundant manipulators'. In: *International Journal of Advanced Robotic Systems* 15 (5 2018), pp. 1–13. DOI: 10.1177/1729881418799562 (cited on page 15).

[100] Haibo Zhou et al. 'Trajectory optimization of pickup manipulator in obstacle environment based on improved artificial potential field method'. In: *Applied Sciences (Switzerland)* 10 (3 2020). DOI: 10.3390/app10030935 (cited on page 15).

[101] Bakir Lacevic and Paolo Rocco. 'Kinetostatic danger field - A novel safety assessment for human-robot interaction'. In: *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings* (2010), pp. 2169–2174. DOI: 10.1109/IROS.2010.5649124 (cited on page 15).

[102] Dae Hyung Park et al. 'Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields'. In: *2008 8th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2008* (2008), pp. 91–98. DOI: 10.1109/ICHR.2008.4755937 (cited on page 15).

[103] Zhihao Liu et al. 'Dynamic risk assessment and active response strategy for industrial human-robot collaboration'. In: *Computers and Industrial Engineering* 141 (January 2020). DOI: 10.1016/j.cie.2020.106302 (cited on page 15).

[104] Yuyang Du et al. 'Robotic manufacturing systems: A survey on technologies to improve the cognitive level in HRI'. In: *Procedia CIRP* 107 (2022). Leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022, pp. 1497–1502. DOI: https://doi.org/10.1016/j.procir.2022.05.181 (cited on page 15).

[105] Zhi Hao Kang, Ching An Cheng, and Han Pang Huang. 'A singularity handling algorithm based on operational space control for six-degree-of-freedom anthropomorphic manipulators'. In: *International Journal of Advanced Robotic Systems* 16 (3 2019), pp. 1–16. DOI: 10.1177/1729881419858910 (cited on pages 15, 17, 20, 21, 64, 105).

[106] Tim Chen and J. C.Y. Chen. 'A New Viewpoint on Control Algorithms for Anthropomorphic Robotic Arms'. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 99 (3-4 2020), pp. 647–658. DOI: 10.1007/s10846-020-01149-5 (cited on page 15).

[107] KUKA. *LBR iiwa*. URL: https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industrial/lbr-iiwa (cited on page 15).

[108] Bruno Siciliano and Oussama Khatib. *Handbook of Robotics*. Springer, 2008 (cited on pages 15–17, 19, 31, 37).

[109] Bruno Siciliano et al. *Robotics: Modeling, Planning and Control*. 1st. Springer-Verlag London, 2009 (cited on pages 15–17, 19, 31, 32, 37, 84, 86, 106).

[110] Xin-Jun Liu, Chao Wu, and Jinsong Wang. 'A New Approach for Singularity Analysis and Closeness Measurement to Singularities of Parallel Manipulators'. In: *Journal of Mechanisms and Robotics* 4.4 (Aug. 2012). 041001. DOI: 10.1115/1.4007004 (cited on page 15).

[111] Mohammad H. FarzanehKaloorazi and Ilian A. Bonev. 'Singularities of the typical collaborative robot arm'. In: *Proceedings of the ASME Design Engineering Technical Conference* 5B-2018 (2018), pp. 1–7. DOI: 10.1115/DETC2018-86305 (cited on pages 15, 103).

[112] Zijia Li, Mathias Brandstötter, and Michael Hofbaur. 'Analysis of kinematic singularities for a serial redundant manipulator with 7 DOF'. In: *ARK 2018 - 16th International Symposium on Advances in Robot Kinematics* (January 2018), pp. 179–186. DOI: 10.1007/978-3-319-93188-3_21 (cited on pages 16, 18).

[113] Liguo Huo and Luc Baron. 'The joint-limits and singularity avoidance in robotic welding'. In: *Industrial Robot* 35 (5 2008), pp. 456–464. DOI: 10.1108/01439910810893626 (cited on page 16).

[114] Mantian Li et al. 'Design and analysis of a whole-body controller for a velocity controlled robot mobile manipulator'. In: *Science China Information Sciences* 63 (7 2020), pp. 1–15. DOI: 10.1007/s11432-019-2741-6 (cited on page 16).

[115] Signe Moe et al. 'Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results'. In: *Frontiers Robotics AI* 3 (APR 2016), pp. 1–18. DOI: 10.3389/frobt.2016.00016 (cited on pages 16, 17, 19).

[116] Xuhao Wang et al. 'Singularity analysis and treatment for a 7R 6-DOF painting robot with non-spherical wrist'. In: *Mechanism and Machine Theory* 126 (June 2018), pp. 92–107. DOI: 10.1016/j.mechmachtheory.2018.03.018 (cited on pages 16, 18, 19).

[117] Stefano Chiaverini. 'Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators'. In: *IEEE Transactions on Robotics and Automation* 13 (3 1997), pp. 398–410. DOI: 10.1109/70.585902 (cited on pages 17, 57, 59).

[118] Glyn James and Phil Dyke. *Advanced Modern Engineering Mathematics*. Ed. by Pearson Education Limited. Fifth. Pearson (cited on pages 17, 19, 104).

[119] Khaled Elashry and Ruairi Glynn. 'Robotic Fabrication in Architecture, Art and Design 2014'. In: *Robotic Fabrication in Architecture, Art and Design 2014* (March 2014). DOI: 10.1007/978-3-319-04663-1 (cited on pages 17, 32).

[120] E. Kreyszig. *Advanced Engineering Mathematics*. John Wiley & Sons, 2010 (cited on page 18).

[121] Filip Marić et al. 'A Riemannian Metric for Geometry-Aware Singularity Avoidance by Articulated Robots'. In: (Mar. 2021). DOI: 10.1016/j.robot.2021.103865 (cited on page 18).

[122] Vince Kurtz, Patrick M. Wensing, and Hai Lin. 'Control Barrier Functions for Singularity Avoidance in Passivity-Based Manipulator Control'. In: (Sept. 2021) (cited on page 18).

[123] Wankun Sirichotiyakul, Volkan Patoglu, and Aykut C. Satici. 'Efficient Singularity-Free Workspace Approximations Using Sum-of-Squares Programming'. In: *Journal of Mechanisms and Robotics* 12.6 (May 2020). 061004. DOI: 10.1115/1.4046997 (cited on page 18).

[124] Morgan Quigley et al. 'ROS: an open-source Robot Operating System'. In: *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. 2009 (cited on page 24).

[125] David Coleman et al. *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*. 2014 (cited on page 24).

[126] N. Koenig and A. Howard. 'Design and use paradigms for Gazebo, an open-source multi-robot simulator'. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727 (cited on page 24).

[127] Universal Robots and FZI. *Universal_Robots_ROS_Driver repository package*. 2021 (cited on pages 25, 87, 103).

[128] Konstantinos Chatzilygeroudis et al. *iiwa_ros: A ROS Stack for KUKA's IIWA robots using the Fast Research Interface*. 2019 (cited on page 25).

[129] Federico Reghenzani, Giuseppe Massari, and William Fornaciari. 'The Real-Time Linux Kernel: A Survey on PREEMPT_RT'. In: 52.1 (2019). DOI: 10.1145/3297714 (cited on page 26).

[130] Cuong Trinh Duc. 'Geometric Perspective on Kinematics and Singularities of Spatial Mechanisms'. In: *Mathematics Ph.D. for the Università degli studi di Genova* (2018). DOI: 10.15167/TRINH-DUC-CUONG_PHD2018-05-04 (cited on page 32).

[131] Keith L. Doty, Claudio Melchiorri, and Claudio Bonivento. 'A Theory of Generalized Inverses Applied to Robotics'. In: *The International Journal of Robotics Research* 12.1 (1993), pp. 1–19. DOI: 10.1177/027836499301200101 (cited on pages 56, 57, 59).

[132] 'Minimal Properties of Generalized Inverses'. In: *Generalized Inverses: Theory and Applications*. New York, NY: Springer New York, 2003, pp. 104–151. DOI: 10.1007/0-387-21634-0_5 (cited on page 56).

[133] pantor. *pantor/ruckig*. original-date: 2020-12-20T11:39:42Z. May 2021. URL: https://github.com/pantor/ruckig (visited on 05/05/2021) (cited on page 57).

[134] Sachin Chitta et al. 'ros_control: A generic and simple control framework for ROS'. In: *The Journal of Open Source Software* (2017). DOI: 10.21105/joss.00456 (cited on pages 71, 73).

[135] OROCOS Project Org. *What is OROCOS?* 2021. URL: https://docs.orocos.org/index.html (cited on pages 71, 72, 75).

[136] G. Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000) (cited on page 88).

[137] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010 (cited on page 88).

[138] Boost. *Boost C++ Libraries*. http://www.boost.org/. Last accessed 2015-06-30. 2015 (cited on page 88).

[139] R. Smits. *KDL: Kinematics and Dynamics Library*. http://www.orocos.org/kdl (cited on page 88).

[140] blodow (GitHub). *Realtime URDF filter*. https://github.com/blodow/realtime_urdf_filter. Last accessed 2023-08-23. 2014 (cited on page 90).

[141] S. Garrido-Jurado et al. 'Automatic generation and detection of highly reliable fiducial markers under occlusion'. In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292. DOI: https://doi.org/10.1016/j.patcog.2014.01.005 (cited on page 91).

[142] Fernando De Terán, Froilán M. Dopico, and D. Steven Mackey. 'Spectral equivalence of matrix polynomials and the Index Sum Theorem'. In: *Linear Algebra and its Applications* 459 (2014), pp. 264–333. DOI: https://doi.org/10.1016/j.laa.2014.07.007 (cited on page 104).

Notation

The following listing describes all the symbols employed along these Ph.D. Thesis document. For any doubt in the meaning of the equation expressions, please check the followings:

$\boldsymbol{\tau}$ Joint torques vector.

$\ddot{\mathbf{q}}$ Joint efforts vector.

$\ddot{\mathbf{q}}_{att}$ Joint acceleration vector of the attractive component of the APF.

$\ddot{\mathbf{q}}_{rep,obs}$ Joint accelerations vector of the repulsive component of the APF due to the obstacle.

$\ddot{\mathbf{q}}_{rep,sing}$ Joint accelerations vector of the repulsive component of the APF due to the singularity.

$\ddot{\mathbf{q}}_{traj}$ Joint acceleration vector command of the computed trajectory.

$\ddot{\mathbf{q}}_t$ Joint accelerations vector of the total component of the APF due to the attractive and repulsive components.

$\ddot{\mathbf{q}}_t^*$ Compensated joint accelerations vector of the total component of the APF due to the attractive and repulsive components.

$\ddot{\mathbf{q}}$ Joint accelerations vector.

$\dot{\mathbf{q}}^{+'}$ Upper joint velocity limit vector.

$\dot{\mathbf{q}}^{+}$ Upper joint velocity limit vector (given by the mechano-electrical limit of the design to be safe).

$\dot{\mathbf{q}}^{-'}$ Lower joint velocity limit vector.

$\dot{\mathbf{q}}^{-}$ Lower joint velocity limit vector (given by the mechano-electrical limit of the design to be safe).

$\dot{\mathbf{q}}_{att}$ Joint velocities vector of the attractive component of the APF.

$\dot{\mathbf{q}}_{rep,obs}$ Joint velocities vector of the repulsive component of the APF due to the obstacle.

$\dot{\mathbf{q}}_{rep,sing}$ Joint velocities vector of the repulsive component of the APF due to the singularity.

$\dot{\mathbf{q}}_{traj}$ Joint velocities vector command of the computed trajectory.

$\dot{\mathbf{q}}_t$ Joint velocities vector of the total component of the APF due to the attractive and repulsive components.

$\dot{\mathbf{q}}_t^*$ Compensated joint velocities vector of the total component of the APF due to the attractive and repulsive components.

$\dot{\mathbf{q}}$ Joint velocities vector.

$\dot{\mathbf{q}}_0$ Joint velocities vector for auxiliary goals such as the singularity handling or the collision avoidance.

$\dot{\mathbf{x}}_B^0$ Cartesian linear and angular velocities vector of the decoupling point according to the world reference frame.

$\dot{\mathbf{x}}_e$ Cartesian linear and angular velocities vector of the end effector.

λ Damping coefficient of DLS inverse kinematic model.

$\mu, \mu(\mathbf{q})$ Manipulability.

$\prod$ Product symbol (as capital pi notation).

ρ Maximum distance parameter of the APF controller.

$\sigma_{highest}$ The biggest singular value of the Jacobian matrix determinant.

$\sigma_i, \sigma_i(\mathbf{J})$ i-th singular value of the Jacobian matrix determinant.

σ_{lowest} The smallest singular value of the Jacobian matrix determinant.

$\mathbf{H}_r^c$ Homogeneous transformation matrix between the camera reference frame and the optical frame.

$\mathbf{O}_i$ Origin of the i-th reference frame of the robot manipulator.

$\mathbf{P}$ The projection matrix.

$\mathbf{p}_c$ Optical frame position of the camera.

$\mathbf{p}_d$ Depth space position representation of point d.

$\mathbf{p}_r$ Camera reference frame position.

$\mathbf{p}_\beta^0$ Relative position vector between the decopling point and the end effector expressed in the β reference system.

$\mathbf{p}_{BE}^0$ Relative position vector between the decopling point and the end effector expressed in the world reference system.

$\mathbf{p}_B^0$ Position vector of the decoupling point expressed in the world reference system.

$\mathbf{p}_E^0$ Position vector of the end effector expressed in the world reference system.

$\mathbf{q}_{att}$ Joint positions vector of the attractive component of the APF.

$\mathbf{q}_{rep,obs}$ Joint positions vector of the repulsive component of the APF due to the obstacle.

$\mathbf{q}_{rep,sing}$ Joint positions vector of the repulsive component of the APF due to the singularity.

$\mathbf{Q}_{traj}$ Compact expression for the commands of the computed trajectory.

$\mathbf{q}_{traj}$ Joint positions vector command of the computed trajectory.

$\mathbf{q}_t$ Joint positions vector of the total component of the APF due to the attractive and repulsive components.

$\mathbf{q}_t^*$ Compensated joint positions vector of the total component of the APF due to the attractive and repulsive components.

$\mathbf{R}_r^c$ Rotation matrix of $\mathbf{H}_r^c$.

$\mathbf{R}_\beta^0$ Rotation matrix of the β-th reference system viewed from the 0-th (world) reference system.

$\mathbf{R}_B^0$ Rotation matrix of the B-th reference system viewed from the 0-th (world) reference system.

$\mathbf{R}_E^0$ Rotation matrix of the E-th reference system viewed from the 0-th (world) reference system.

$\mathbf{R}_i^j$ Rotation matrix of the i-th reference system viewed from the j-th reference system.

$\mathbf{t}_r^c$ Translation matrix of $\mathbf{H}_r^c$.

$\mathbf{T}_i^j$ Homogeneous transformation matrix of the i-th point according to the j-th reference frame.

$\mathbf{X}_i$ X direction of the principal axis for the i-th reference frame of the robot manipulator.

$\mathbf{Y}_i$ Y direction of the principal axis for the i-th reference frame of the robot manipulator.

$\mathbf{Z}_i$ Z direction of the principal axis for the i-th reference frame of the robot manipulator.

$\mathbf{z}_\beta$ Unitary Joint β auxiliary axis vector.

$\mathbf{z}_{i-1}$ Unitary Joint i axis vector.

c_x Optical center pixel x-coordinate.

c_y Optical center pixel y-coordinate.

CN Condition Number.

cp_i i-th Control Point distributed along the robot structure.

$D_{filter,i}$ Depth map after the Advanced Scene Segmentation algorithm is applied.

D_i Depth map with the raw data image taken by the camera or vision sensors.

f_{ctrl} Frequency used by the ROS Control based controller to send new commands to the robot.

f_{perc} Frequency of the set of sensors required to take a whole cycle of information of the robot environment.

f_{proc} Frequency of the robot environment processing nodes.

f_{s_x} Focal length of the camera in the x-axis.

f_{s_y} Focal length of the camera in the y-axis.

h Decoupling height.

l Decoupling length.

n Degrees of Freedom (DoFs of the cobot).

$r\beta$ Decoupling angle.

r_1 Constant distance between the elbow joint and the wrist in the kinematic decoupled model.

r_2 Constant distance between the wrist and the end effector in the kinematic decoupled model.

x_c Optical frame position of the camera x-coordinate.

x_d Depth space position representation of point d x-coordinate.

y_c Optical frame position of the camera y-coordinate.

y_d Depth space position representation of point d y-coordinate.

z_c Optical frame position of the camera z-coordinate.

z_d Depth space position representation of point d z-coordinate.

$\mathbf{f}(\mathbf{R}_B^0(\mathbf{q}))$ Compact form to express the orientation as Euler roll, pitch and yaw angles.

$\mathbf{I}_n$ Unitary diagonal matrix of n-th dimension.

$\mathbf{J}$ Geometric Jacobian matrix.

$\mathbf{J}^\dagger$ Right-Pseudoinverse (geometric) Jacobian matrix.

$\mathbf{J}^T$ Transpose of the Geometric Jacobian matrix.

$\mathbf{J}_a$ Analytical Jacobian matrix.

$\mathbf{o}_B^0$ Oreintation of the decoupling point according to the world refernce system.

$\mathbf{o}_K^j$ Oreintation of the point $\mathbf{K}$ according to the j-th reference system.

$\mathbf{q}$ Joint positions vector.

$\mathbf{x}_e$ Cartesian position and orientation vector of the end effector.

Glossary

A

AI Artificial Intelligence. 12, 49

APF Artificial Potential Field. v, vii, xvi, 3, 5, 8, 14–16, 20–22, 45, 47, 48, 50, 51, 53, 55–57, 59–61, 66, 67, 69–71, 77, 78, 80, 88, 89, 92, 93, 98, 99, 103, 107, 109, 115–119, 123, 124, 127–130

Artificial Potential Field is a technique first employed by Oussama Khatib on mobile robots that creates repulsive and attractive virtual forces to command the robot and make them achieve their goals. The aim of the repulsive virtual force is to repel from colliding with obstacles in the robot surrounding, while the attractive virtual forces involves the movement towards the goal of the robot. One of the main problems of these control techniques is the well-known local minima problem where the robot can get stuck due to an equilibrium between the repulsive and attractive forces.. 3

C

CN Condition Number. 16, 17

Cobot cobot. v, xv, 5, 7–10, 15–17, 20, 21, 23, 25, 26, 31–33, 35–41, 45, 46, 59, 61, 62, 71, 76, 80, 81, 93, 99, 103, 106, 121, 124, 127, 129, 130

Control Point a Control Point (cp) is a virtually placed reference along the robot structure from which the distances between the robot arm and the obstacle should be measured.. 51–53

D

d-APF dual Artificial Potential Field. v, vii, xv, xvi, 5, 21–25, 45, 59, 60, 63, 64, 66, 68–71, 88, 89, 92, 93, 95, 98, 99, 103, 106, 107, 109, 115–119, 121–125, 127–131

DC Direct Current. 26

decoupling point is the point in the structure of the robot where the motion of the wrist becomes decoupled from the motion of the arm, allowing the positioning of the end effector just by positioning the wrist in the desired place.. 32

DH Denavit Hartenberg. 17, 31, 39, 81, 82, 94, 95

DLS Damped Least Square. v, vii, xv, xvi, 5, 16, 17, 20, 22, 45, 47–51, 53–57, 59–62, 70, 71, 80, 88, 89, 92, 93, 98, 99, 103, 107, 109, 116–119, 123, 124, 127–129

DMP Dynamic Movement Primitive. 15

DoF Degree of Freedom. xv, 18, 19, 21, 26, 31, 35–37, 39–41, 64, 65, 98, 105, 127

F

FK Forward Kinematic. 15, 23, 31, 33–36, 39, 41, 80, 83, 94

G

GMR Gaussian Mixture Regression. 14

H

Hardware Interface a hardware interface provides a standardized way for software components to interact with hardware devices such as sensors and actuators. Hardware interfaces define a set of functions or messages that nodes can use to communicate with hardware devices, abstracting away the low-level details of how the device works and providing a consistent interface for different hardware components. By using a hardware interface, developers can write software that can work with different hardware devices without having to write custom code for each one, making it easier to develop and maintain complex robotic systems.. 71–73, 76–78, 87–89, 97, 98

HMM Hidden Markov Model. 14

HRC Human-Robot Collaboration. 8, 11

HRCp Human-Robot Cooperation. 11

HRCx Human-Robot Coexistence. 11

I

IK Inverse Kinematic. 8, 15, 16, 20, 21, 23, 31, 32, 34, 36–41, 54–56, 59, 62, 71, 80, 83–85, 94, 95, 127

industrial collaborative scenario is an industrial manufacturing shared environment between operators and robots (preferibable collaborative robots) where at least one operator and a robot developed a tasks together working simultaneously.. xv, 5, 7–11, 20, 23, 32, 60

J

joint space is defined by a vector whose components are the translational and angular displacements of each joint of a robotic link. The common linear PID does not include any component of the robot dynamics into its control law whenever it is used in joint space or task-space.. 14, 34, 55, 56, 62, 68, 69

K

KF Kalman Filter. 14

M

Manipulability manipulability. 16, 17, 31

N

node in ROS, a node is a process that performs a specific task and communicates with other nodes through a communication system called the "ROS communication graph".. 25, 79, 80

non-spherical wrist also known as non-concatened wrist, refers to a type of robot which their last three joints intersect two by two, but not all in the same spot.. v, 5, 7–9, 15, 16, 19–21, 23, 25, 31–33, 35–41, 45, 61, 62, 71, 81, 93, 94, 96, 99, 103, 106, 118, 124, 127, 128, 130, 131

null space it is known as null space the set of configurations where the robot present a null Jacobian determinant. In other words, all the configurations that correspond to a singular configuration of the robot compound the null space.. 15, 16, 20

O

obstacle in this context is employed to describe moving bodies in the surroundings of the cobot, which generally corresponds with the human operator.. 20, 90, 98

ODE Ordinary Differential Equation. 17, 18, 31

OS Operating System. 26, 27, 71, 74, 124

OTG Online Trajectory Generator. 14, 63

Q

quasi-spherical wrist it is denoted as quasi-spherical wrist in this Ph.D. a non-spherical wrist from a robot structure that kinematically behaves as if it is an spherical one.. v, 33, 127

R

ROS Robot Operating System. v, vii, xv–xvii, 24, 25, 56, 71–80, 87–91, 93–95, 97–99, 103, 106, 115, 117, 122, 124, 128, 131

ROS Parameter Server is a shared, centralized storage system that allows nodes to store and retrieve parameters (i.e., key-value pairs) at runtime. The Parameter Server can be used to store configuration settings, calibration data, and other information that nodes need to operate effectively, and can also be accessed by multiple nodes at the same time. This makes it a convenient and useful tool for managing and sharing information in a ROS system.. 46, 51, 79, 80, 88, 90, 117

S

scene in this context, the scene is composed of every static element surrounding the robot. They are usually static elements such as the table where the robot is installed and the sensors that are stuck to a position, among other devices and elements.. 14, 22, 24, 46, 53, 90

singular configuration in this document, the use of singular configuration, singularity and singular region are used indistinctly. A singular configuration is a robot configuration where the determinant of their Jacobian matrix becomes null (or enough close to zero), making the robot to compute excessive velocities of forces for small movements. In general, it can be defined as $\Psi = \{\mathbf{q} \in \mathbb{R}^n : |\mathbf{J}(\mathbf{q})| \leq s_0\}$, where s_0 is a close to zero threshold defined by the user.. 17, 20, 23, 31, 38, 40, 41, 59, 63, 85, 104, 127, 128

Singularity singularity. 59, 128

spherical wrist it is refer to a type of robot which their last three joint axis intersect on a common point. In this manner, the reference system of those joints can be placed coincident in the same spot of the robot structure while developing the kinematic model. Thus, the forward and inverse kinematic models are simplified.. 15, 21, 23, 32, 33, 62

SVD Singular Value Decomposition. 17, 18, 31, 56, 104

T

task space also known as Cartesian space, is defined by the position and orientation of the end effector of a robot. Joint space is defined by a vector whose components are the translational and angular displacements of each joint of a robotic link.. 14, 16, 18, 34, 39, 54–56, 68, 69, 106

TCP Tool Center Point. 32–34, 39, 53, 54, 69, 82, 106

W

wDLS weighted Damped Least Square. 56, 57, 59, 123, 128, 129

workpiece in this context, a workpiece is any load the robot might interact with. They are usually static components of the robot's surroundings.. 34, 49, 90, 119, 131

wrist spherification technique developed in this Ph.D. where a point along a non-spherical (non-concatenated) wrist cobot is selected to keep a constant distance between the last arm joint, the decoupled point, and the end effector. The technique itselft is explained in Chapter 4.. v, 33, 45